Oracle® Communications

Cloud Native Core Console Installation and Upgrade Guide

Release 22.3.2
F59336-05
January 2023

Oracle Communications Cloud Native Core Console Installation and Upgrade Guide, Release 22.3.2

F59336-05

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 provides information about the acronyms and the terminology used in the document:

Table Acronyms

Acronym	Definition
AD	Active Directory
ASM	Aspen Service Mesh
BSF	Binding Support Function
CNCC	Cloud Native Core Console
CNE	Cloud Native Environment
OCCNE	Oracle Communications Cloud Native Environment
CS	Common Service
CRUD Operations	CREATE, READ, UPDATE, DELETE
ECDSA	Elliptic Curve Digital Signature Algorithm
EIR	Equipment Identity Register
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity Access Management
KPI	Key Performance Indicator
M-CNCC	Manager CNC Console or M-CNCC (also known as mCncc) is a CNCC instance which manages local OCCNE common service(s) and remote Agent CNC Console (s) (A-CNCC). M-CNCC has two components: M-CNCC IAM and M-CNCC Core.
M-CNCC IAM	Manager CNC Console IAM or M-CNCC IAM (also known as mCncc Iam) is an IAM component of M-CNCC. M-CNCC IAM contains M-CNCC IAM Ingress Gateway and M-CNCC IAM back-end microservices.
M-CNCC Core	Manager CNC Console Core or M-CNCC Core (also known as mCncc Core) is a core component of M-CNCC that provides GUI and API access portal for accessing NF and OCCNE common services. M-CNCC Core contains M-CNCC Core Ingress Gateway and M-CNCC Core back-end microservices.
A-CNCC	Agent CNC Console is a CNCC Core instance which manages local NF(s) and local OCCNE common services(s). A-CNCC is managed by M-CNCC. A-CNCC contains A-CNCC Core Ingress Gateway and A-CNCC Core back-end microservices. A-CNCC has no IAM component. A-CNCC is also known as A-CNCC Core or aCncc Core.
A-CNCC Kubernetes cluster	Kubernetes cluster hosting A-CNCC
M-CNCC Kubernetes cluster	Kubernetes cluster hosting M-CNCC

Table (Cont.) Acronyms

Acronym	Definition
mTLS	Mutual Transport Layer Security
Instance	OCNF or OCCNE common service managed by either M-CNCC Core or A-CNCC Core.
Site	Kubernetes Cluster
CS	OCCNE Common Services like Grafana, Kibana, Jaeger, Prometheus, Alertmanager and so on.
MC	Multi Cluster. In multi cluster, a single CNCC can manage NF instances that accessess different Kubernetes clusters.
MO	Mananged Objects
MOS	My Oracle Support
LDAP	Lightweight Directory Access Protocol
LDAPS	Lightweight Directory Access Protocol (Over SSL)
NRF	Network Repository Function
OCNF	Oracle Communications Network Function
OSDC	Oracle Software Delivery Cloud
OSO	Operations Services Overlay
REST API	Representational State Transfer Application Programming Interface
SCP	Service Communication Proxy
SAML	Security Assertion Markup Language
SEPP	Security Edge Protection Proxy
TLS	Transport Layer Security
UDR	Unified Data Repository
UE	User Equipment
URI	Subscriber Location Function

What's New in This Guide

This section introduces the documentation updates for Release 22.3.x in Oracle Communications Cloud Native Core Console Installation and Upgrade Guide.

Release 22.3.2 -F59336-05, January 2023

- Updated the caution in the Deploying CNC Console section to include the rollback scenario.
- Updated the Compatibility Matrix section to support the latest version of NFs.

Release 22.3.1 -F59336-04, November 2022

No updates are made in this document.

Release 22.3.1 -F59336-02, October 2022

- Updated the CNC Console Upgrade and Rollback section to support single helm chart.
- Updated the Compatibility Matrix section to support the latest version of NFs.

Release 22.3.0 -F59336-01, August 2022

- Updated the Installing CNC Console section to support Single helm chart.
- Updated the Configuring CNC Console to support ASM and OSO section to support single helm chart.
- Updated the CNC Console Debug tools section to support single helm chart.
- Updated the CNC Console Upgrade and Rollback section to support single helm chart.
- Updated the Performing Helm Test section to support the latest version of helm test.
- Updated the Compatibility Matrix section to support the latest versions of NFs.
- Updated the CNC Console Resource Requirement section.
- Updated the CNC Console Microservices to Port Mapping section.
- Updated the A-CNCC Core and M-CNCC Core Instances Configuration Examples.
- Updated the CNC Console Global Configurations section with the following helm parameters to support helm test and Single helm chart deployment:

  Single helm chart parameters:

  – global.cncc-iam.enabled

  Helm test parameters:

  – global.helmTestServiceAccountName
  – global.test.resources
  – global.test.complianceEnable
- The following static nodeport parameters are depreciated from the custom_values.yaml:

- staticNodePortEnabled
- staticHttpNodePort
- staticHttpsNodePort

1 Introduction

This document provides information on how to install and upgrade Oracle Communications Cloud Native Core Console.

Overview

The Cloud Native Core Console (CNC Console) is a single screen solution to configure and manage Network Functions (NFs). The CNC Console has the following two modules:

- CNC Console Core (CNCC Core): CNCC Core acts as GUI or API portal for NFs and OCCNE common services. CNCC Core module includes CNC Console and its integration with other Cloud native core network functions. The CNCC provides user interface that can be used to configure parameters for the following CNC network functions:
  - Binding Support Function (BSF)
  - Service Communication Proxy (SCP)
  - Network Repository Function (NRF)
  - Cloud Native Core Policy
  - Security Edge Protection Proxy (SEPP)
  - Unified Data Repository (UDR)
  - Network Slice Selection Function (NSSF)
  - CNE Common Services
- CNC Console Identity Access Management (CNCC IAM): CNCC IAM acts as local identity provider and broker for external identity provider. CNCC IAM module includes the required authentication and authorization procedures such as creating and assigning roles to users.

Continuous Delivery Control Server (CDCS)

CNC Console can be deployed using Continuous Delivery Control Server (CDCS) or Command Line Interface (CLI) procedures as described in Installing CNC Console. CDCS provides continuous delivery functionality for multi-site Cloud Native Core (CNC) installations. For more information about CDCS, see *Oracle Communications Cloud Native Core CD Control Server User Guide*.

CDCS is a centralized server that automates CNC Console deployment processes such as installation, upgrade, and rollback CNC Console. CLI provides an interface to run various commands required to install, upgrade, and roll back CNC Console.

CNC Console installation comprises of prerequisites, pre deployment , installation, and post installation tasks. You must perform CNC Console installation tasks in the same sequence as outlined in the following table:

Task	Subtasks	Reference	Applicable for CDCS	Applicable for CLI
Prerequisites: This section describes how to set up the installation environment.		Prerequisites	Yes	Yes
	Software Requirements	Software Requirements	Yes	Yes
	Environment Setup Requirements	Environment Setup Requirements	Yes	Yes
	Resource Requirements	Resource Requirements	Yes	Yes
Downloading CNC Console		Downloading CNC Console package	See Oracle Communications CD Control Server Installation and Upgrade Guide	Yes
CNC Console Predeployment Configuration		CNC Console Predeployment Configuration	Yes	Yes
	Verifying and Creating CNC Console Namespace	Verifying and Creating CNC Console Namespace	Yes	Yes
	Configuring Database	Configuring Database	Yes	Yes
	Installing CNC Console			
CNC Console Predeployment Configurations		CNC Console Predeployment Configurations	Yes	Yes
Global Configurations		Global Configurations	Yes	Yes
	CNC Console Configuration for Service Account	CNC Console Configuration for Service Account	Yes	Yes
	Configuring ASM and OSO in M-CNCC IAM	Configuring ASM and OSO in M-CNCC IAM	Yes	Yes
CNC Console IAM Predeployment Configuration		CNC Console IAM Predeployment Configuration	Yes	Yes
	Configuring M-CNCC IAM Database	Configuring M-CNCC IAM Database	Yes	Yes
	Configuring Secret for Default or Admin User in M-CNCC IAM	Configuring Secret for Default or Admin User in M-CNCC IAM	Yes	Yes

Task	Subtasks	Reference	Applicable for CDCS	Applicable for CLI
	Configuring Secret to Enable HTTPS in M-CNCC IAM	Configuring Secret to Enable HTTPS in M-CNCC IAM	Yes	Yes
	Configuring LDAPS in M-CNCC IAM	Configuring LDAPS in M-CNCC IAM	Yes	Yes
M-CNCC Core Predeployment Configuration		M-CNCC Core Predeployment Configuration	Yes	Yes
	Configuring MySQL in M-CNCC Core	Configuring MySQL in M-CNCC Core	Yes	Yes
	Configuring Secret to Enable HTTPS in M-CNCC Core	Configuring Secret to Enable HTTPS in M-CNCC Core	Yes	Yes
A-CNCC Core Predeployment Configuration		A-CNCC Core Predeployment Configuration	Yes	Yes
	Configuring A-CNCC Core Database	Configuring A-CNCC Core Database	Yes	Yes
	Configuring Secret to Enable HTTPS in A-CNCC Core	Configuring Secret to Enable HTTPS in A-CNCC Core	Yes	Yes
	Configuring A-CNCC Core mTLS	Configuring A-CNCC Core mTLS	Yes	Yes
Deploying CNC Console		Deploying CNC Console	See Oracle Communications CD Control Server Installation and Upgrade Guide	Yes
	Verifying CNC Console Installation	Verifying CNC Console Installation	Yes	Yes
Customizing CNC Console		Customizing CNC Console	Yes	Yes
Accessing CNC Console		Accessing CNC Console	Yes	Yes
Upgrading CNC Console		Upgrading CNC Console	See Oracle Communications CD Control Server Installation and Upgrade Guide	Yes
Uninstalling CNC Console		Uninstalling CNC Console	Yes	Yes

Task	Subtasks	Reference	Applicable for CDCS	Applicable for CLI
CNC Console IAM Post Installation Steps		CNC Console IAM Post Installation Steps	Yes	Yes
Performing Helm Test		Performing Helm Test	Yes	Yes
Configuring CNC Console to support ASM and OSO		Configuring CNC Console to support ASM and OSO	Yes	Yes
CNC Console Debug Tools		CNC Console Debug Tools	Yes	Yes

CNC Console Compatibility Matrix

The following table provides the list of network functions that are compatible with CNC Console 22.3.x:

Release 22.3.2

Table 1-1 Compatibility Matrix

Network Functions	Compatible Versions
BSF	22.3.x
NRF	22.3.x
NSSF	22.3.x
Policy	22.3.x
SCP	22.3.x
SEPP	22.3.x
UDR	22.3.x

Table 1-2 Compatibility Matrix

Components	Compatible Versions
CNE	22.1.x, 22.2.x, 22.3.x
cnDBTier	22.1.x, 22.2.x, 22.3.x
CDCS	22.3.x
OSO	1.6.2, 1.10.x, 22.3.x
ASM	1.4.6-am9, 1.6.14-am4, 1.9.8-am1

CNC Console Deployment Modes

Introduction

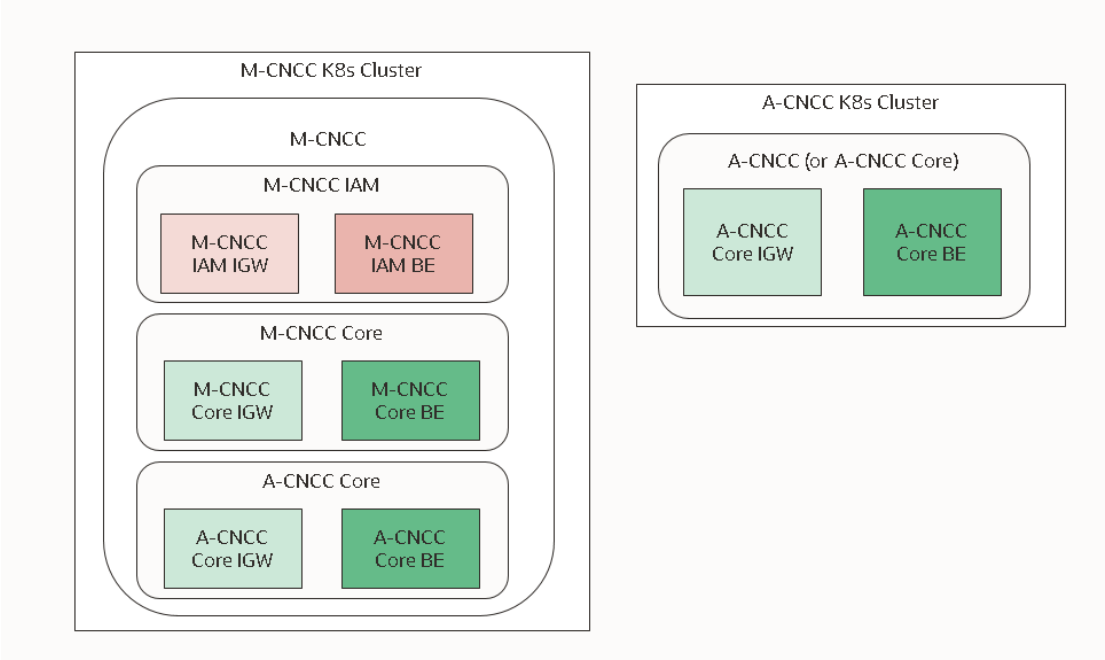
The CNC Console supports both single and multiple cluster deployments. In a single cluster deployment, the CNC Console manages NFs and OCCNE common services deployed in the local Kubernetes clusters. In a multiple cluster deployment, the CNC

Console manages NFs and OCCNE common services deployed in the remote Kubernetes clusters. This section explains the Console component overview, terminology used and Console Single Cluster and Multi Cluster deployment details.

CNC Console Component Overview

The following diagram represents the component overview of CNC Console.

Figure 1-1 CNC Console Component Overview



The CNC Console has following 2 components:

- M-CNCC
- A-CNCC

M-CNCC

Manager CNC Console or M-CNCC is a CNCC instance which manages multiple A-CNCC and local instances.

M-CNCC has two components M-CNCC IAM and M-CNCC Core.

M-CNCC IAM
Manager CNC Console IAM or M-CNCC IAM is an IAM component of M-CNCC. M-CNCC IAM contains M-CNCC IAM Ingress Gateway (CNCC IAM IGW) and M-CNCC IAM Back End (M-CNCC IAM BE) microservices.

M-CNCC Core
Manager CNC Console Core or M-CNCC Core is a core component of M-CNCC which provide GUI and API access portal for accessing NF and OCCNE common service. M-CNCC Core contains M-CNCC Core Ingress Gateway (CNCC Core IGW) and M-CNCC Core Back End (M-CNCC Core BE) microservices.

A-CNCC

A-CNCC Core

Agent CNC Console or A-CNCC Core is a CNCC Core instance which manages local NF(s) and local OCCNE common services(s). It is managed by M-CNCC.

A-CNCC Core contains A-CNCC Core Ingress Gateway and A-CNCC Core Back End microservices.

A-CNCC Core has no IAM component

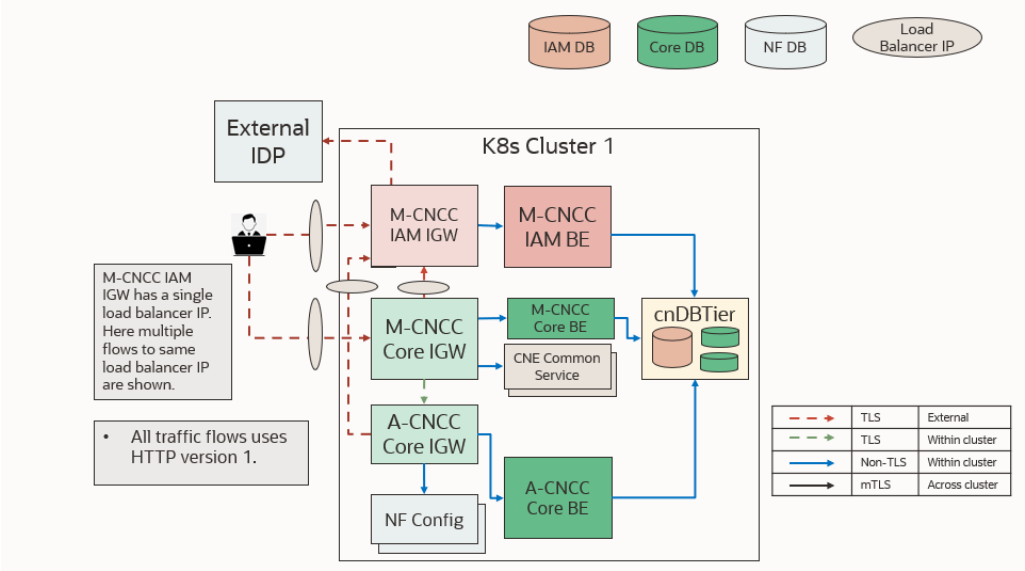
A-CNCC Core is an optional component needed to manage NF instances.

CNC Console Single Cluster Deployment

In a single cluster deployment, CNC Console can manage NFs and OCCNE common services deployed in the local kubernetes cluster.

The following diagram represents the CNCC single cluster deployment:

Figure 1-2 single cluster deployment

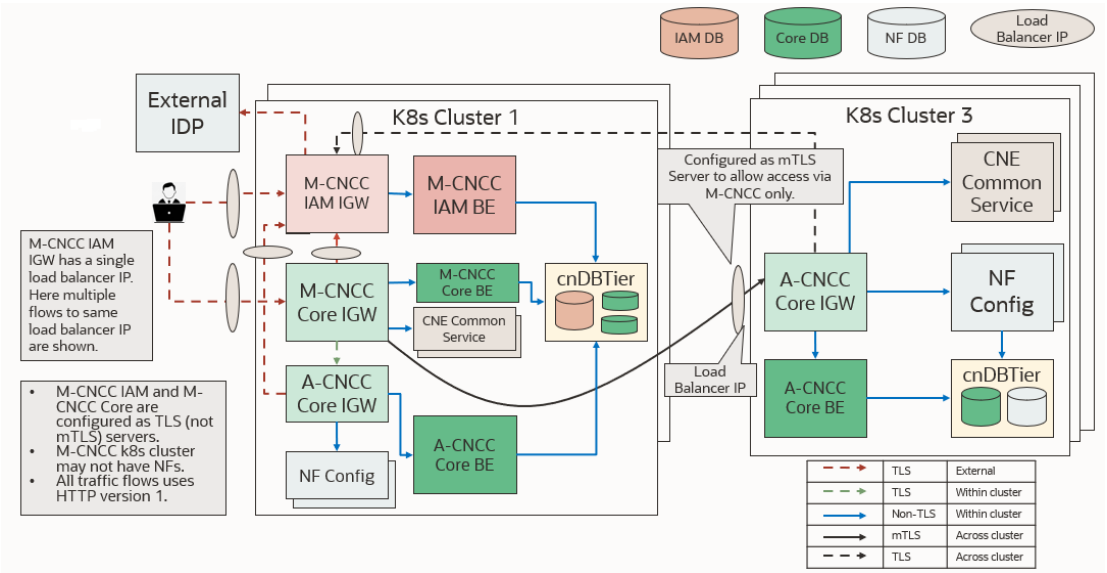


CNC Console Multi Cluster Deployment

In a multi cluster deployment, CNC Console can manage NFs and OCCNE common services deployed in the remote kubernetes cluster(s). CNC Console instance called A-CNCC is needed on remote kubernetes clusters for this deployment. CNC Console instance providing the API access through GUI portal and managing the A-CNCC(s) is called M-CNCC.

The following diagram represents the CNC Console multi cluster deployment:

Figure 1-3 multi cluster deployment



Note:

- For a single cluster deployment both manager (CNCC IAM, M-CNCC Core) and agent(A-CNCC) to be deployed on the same cluster.
- For a multi cluster deployment, if manager cluster has a local NF deployment then both manager(CNCC IAM, M-CNCC Core) and agent(A-CNCC) to be deployed on the same cluster.
- In case manager cluster does not have a local NF deployment, then only manager (CNCC IAM, M-CNCC Core) is to be deployed and agent (A-CNCC) to be deployed on a cluster where NFs are present on the cluster.
- The manager manages CNE or OSO common services if present in a cluster.
  - Manager in a cluster is preferred over Agent in the same cluster to manage the CNE common services.
  - Agent in a cluster can manage CNE common service in absence of a Manager in the same cluster.
- Agent is needed only when NFs are present on the cluster.

CNC Console Deployment Support Matrix

The following table provides details on support of Console Deployment features for various network functions:

The following table provides details on support of Console Deployment features for various network functions:

Table 1-3 CNC Console Deployment Support Matrix

Deployment	Single Instance or Multiple Instance of NF Support	Policy	BSF	SCP	UDR	NRF	SEPP	NSSF
Single-Cluster	Single Instance per cluster	YES	YES	YES	YES	YES	YES	YES
	Multiple Instance per cluster	YES	YES	YES	YES	YES	NO	NO
	Upgrade Order (Source Release 22.1.x/ Older and Target Release 22.3.x) (CNCC or NF to be upgraded first?)	NF	NF	NF	NF	NF	NF	NF
	Upgrade Order (Source Release 22.2.x/ 22.3.0 and Target Release 22.3.x) (CNCC or NF to be upgraded first?)	CNCC	CNCC	CNCC	CNCC	CNCC	CNCC	CNCC
Multi-Cluster	Single Instance per cluster	YES	YES	YES	YES	YES	YES	YES
	Multiple Instance per cluster	YES	YES	YES	YES	YES	NO	NO

Table 1-3 (Cont.) CNC Console Deployment Support Matrix

	Upgrade Order (Source Release 1.9.x and Target Release 22.3.x) (CNCC or NF to be upgraded first?)	NO	NO	NO	NO	NO	NO	NO
	Upgrade Order (Source Release 22.1.x and Target Release 22.3.x) (CNCC or NF to be upgraded first?)	CNCC	NO	NO	NO	NO	NO	NO
	Upgrade Order (Source Release 22.2.x/ 22.3.0 and Target Release 22.3.x) (CNCC or NF to be upgraded first?)	CNCC	CNCC	CNCC	CNCC	CNCC	CNCC	CNCC

Note:

- Single Console Instance supports single instances of all NFs. Example: A single instance of Console can handle single instance of POLICY, SCP, NRF etc.
- Multiple Instance of NF Support means a whether a single instance of Console can handle multiple instances of a single NF type within a Kubernetes cluster. Example: Whether a single Console can handle multiple instances of POLICY or SCP within a Kubernetes cluster.

Note:

From Release 22.2.0 onwards CNC Console has a consistent deployment model for Single cluster and multi cluster deployment. Single cluster deployment has Manager (M-CNCC IAM (CNCC IAM), M-CNCC Core) and Agent (A-CNCC Core) deployed in a same Kubernetes cluster.

Configuration Workflow

This section explains the configuration workflow for the following scenarios:

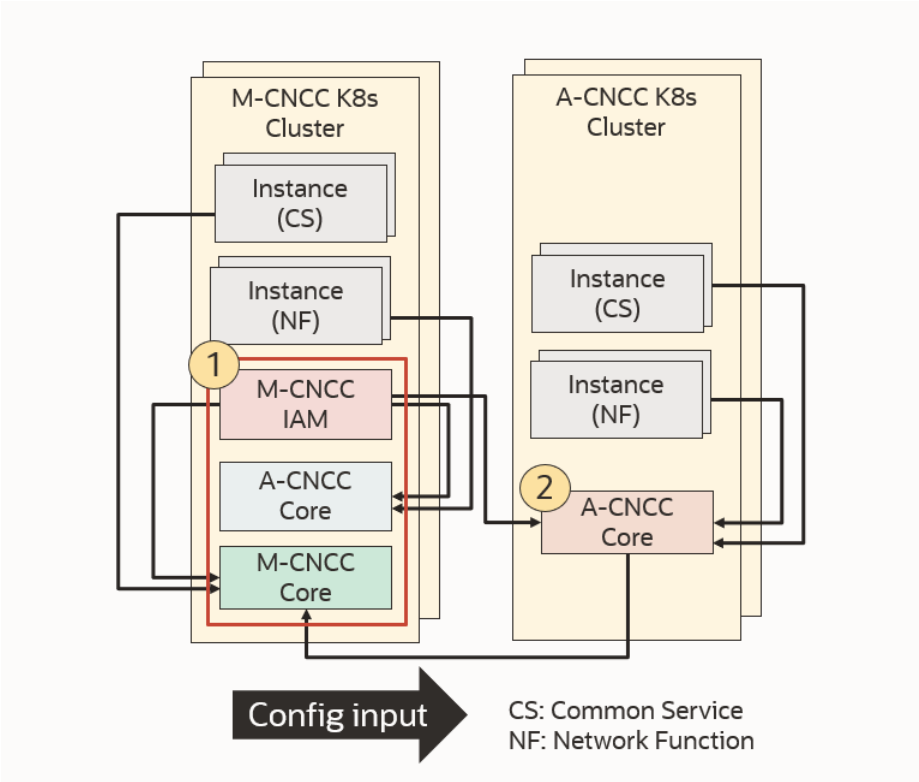
- Fresh Installation
- Add a new M-CNCC
- Add a new A-CNCC
- Remove M-CNCC from A-CNCC
- Remove A-CNCC from M-CNCC

Fresh Installation

This section explains how to do a fresh installation.

The following diagram represents the fresh installation.

Figure 1-4 Fresh Installation of M-CNCC and A-CNCC



Procedure:

1.Install M-CNCC IAM, A-CNCC Core, M-CNCC Core on M-CNCC Kubernetes clusters

- Configuration Input: M-CNCC IAM(s), A-CNCC Core(s) and instances.
- A-CNCC Core on M-CNCC k8s cluster is optional and needed only if there are NF instances on the M-CNCC k8s cluster.

> **Note:**
>
> CS Instances are managed directly by M-CNCC Core and does not need A-CNCC Core.

2.Install A-CNCC Core on A-CNCC k8s clusters

- Configuration Input: M-CNCC IAMs and local instances.

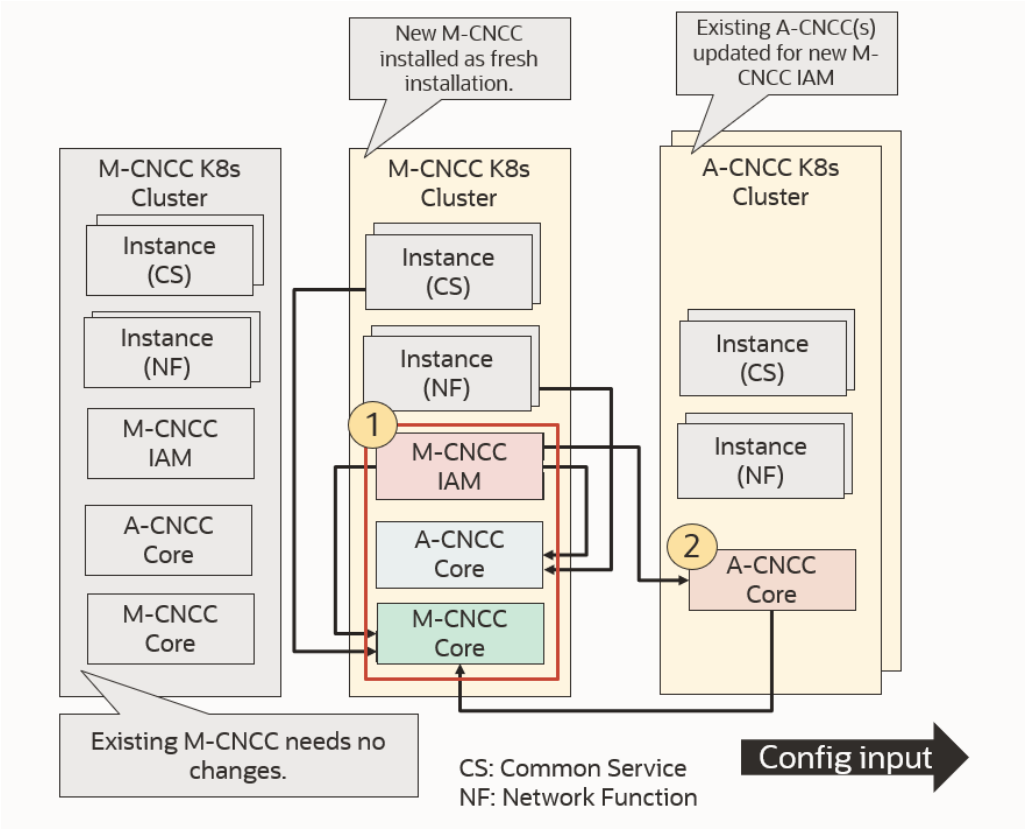
See Installing CNC Console section for installation procedure.

Add a New M-CNCC

This section explains how to add a new M-CNCC.

The following diagram represents the addition of a new M-CNCC:

Figure 1-5 Add a new M-CNCC



Procedure:

1. Install M-CNCC IAM, A-CNCC Core, M-CNCC Core on M-CNCC Kubernetes cluster

- Configuration Input: M-CNCC IAM(s), A-CNCC Core(s) and instances.
- A-CNCC Core on M-CNCC k8s cluster is optional and needed only if there are NF instances on the M-CNCC k8s cluster.

> **Note:**
>
> CS Instances are managed directly by M-CNCC Core and does not need A-CNCC Core.

2. Update A-CNCC Core(s) on A-CNCC k8s cluster(s)

- Configuration Update: Newly added M-CNCC IAM.

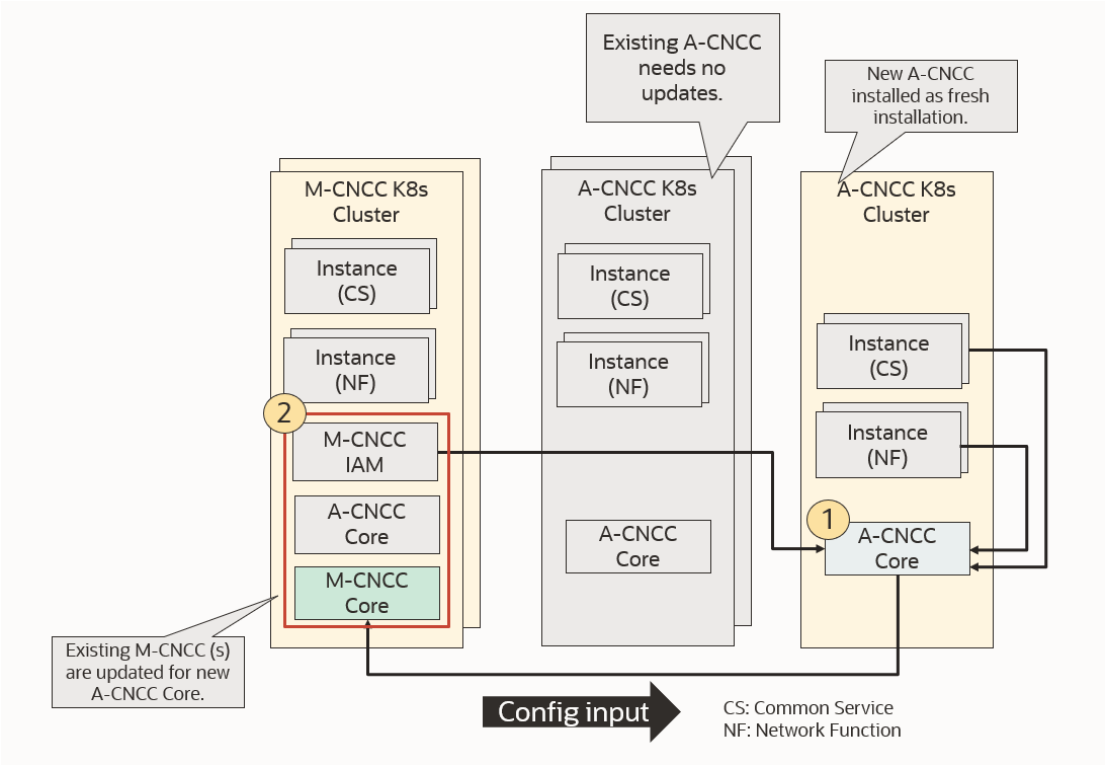
See Installing CNC Console section for installation procedure.

Add a New A-CNCC

This section explains how to add a new A-CNCC.

The following diagram represents the addition of a new A-CNCC:

Figure 1-6 Add a new A-CNCC



Procedure

1.Install A-CNCC Core on new A-CNCC Kubernetes cluster.

- Configuration Input: M-CNCC IAMs and local instances.

2.Update M-CNCC Core on existing M-CNCC Kubernetes clusters.

- Configuration Update: Newly added A-CNCC.

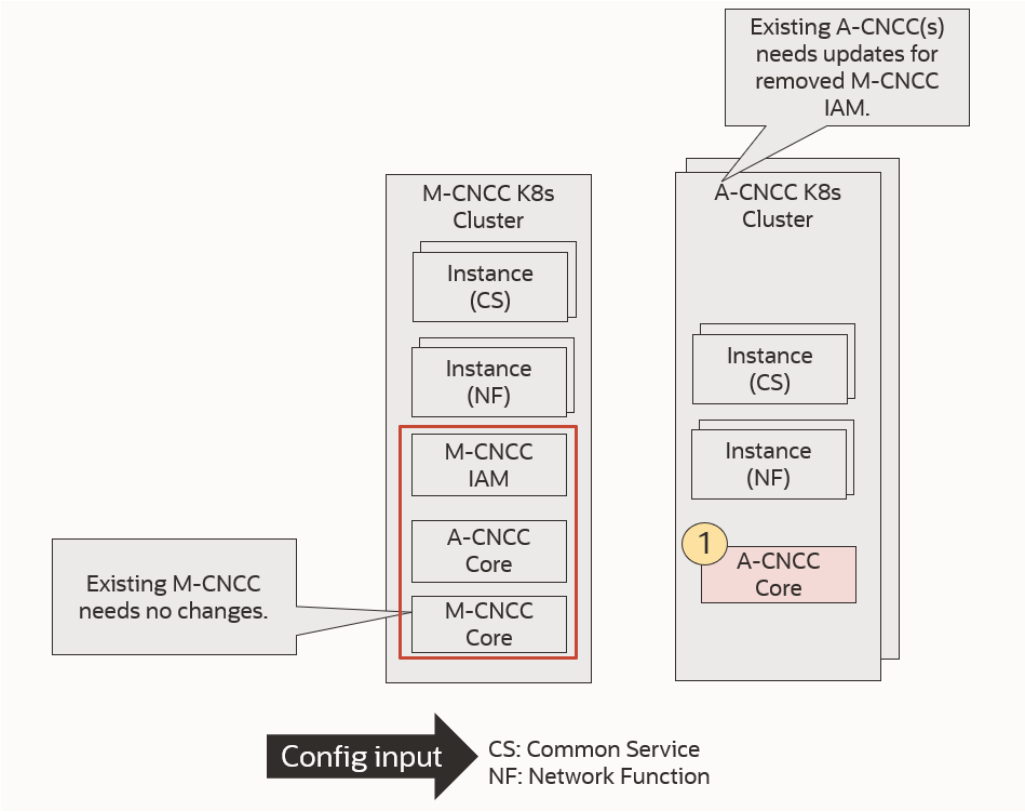
See Installing CNC Console section for installation procedure.

Remove M-CNCC from A-CNCC

This section explains how to remove M-CNCC from A-CNCC.

The following diagram represents the removal of M-CNCC from A-CNCC:

Figure 1-7 Remove M-CNCC from A-CNCC



Procedure

1.Update A-CNCC (one or more)

- Configuration Update: Edit A-CNCC configuration to remove M-CNCC IAM

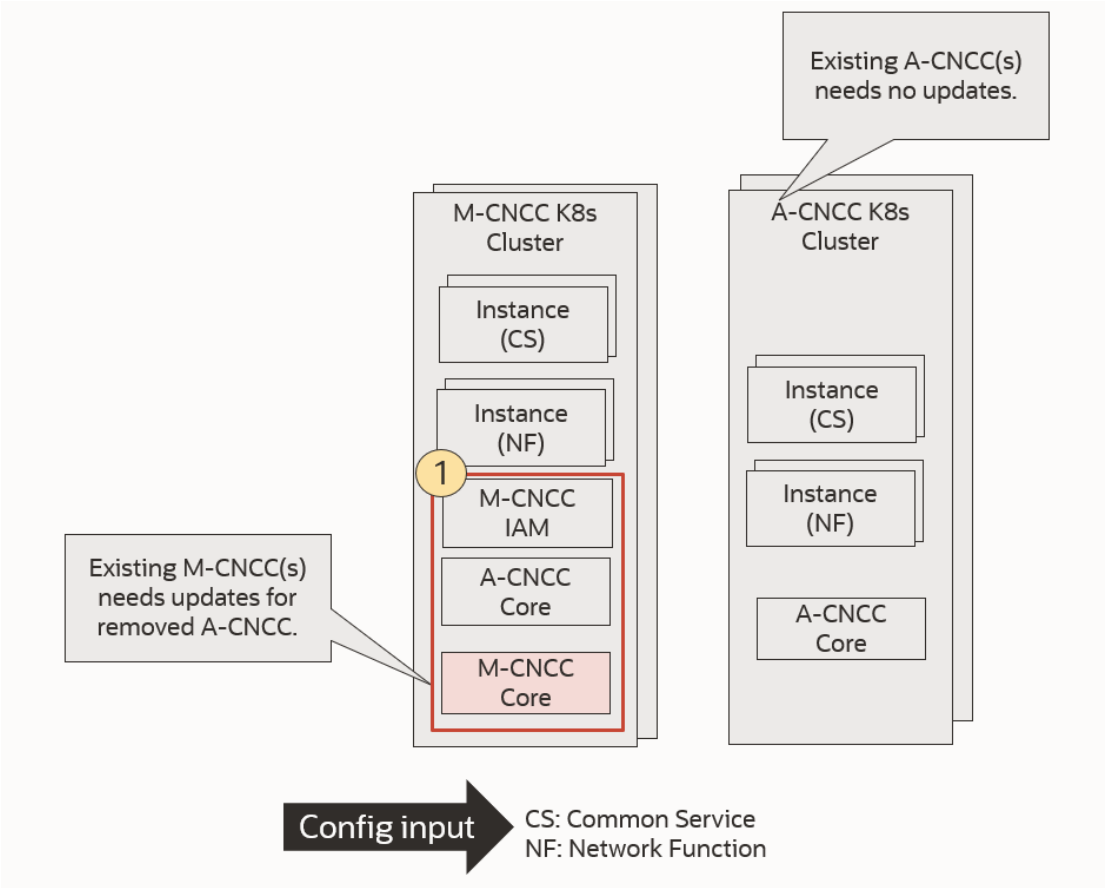
See Installing CNC Console section for installation procedure.

Remove A-CNCC from M-CNCC

This section explains how to remove A-CNCC from M-CNCC.

The following diagram represents the removal of A-CNCC from M-CNCC:

Figure 1-8 Remove A-CNCC from M-CNCC



Procedure

1.Update M-CNCC (one or more)

- Configuration Update: Edit M-CNCC Core configuration to remove A-CNCC

See Installing CNC Console section for installation procedure.

CNC Console Configuration Maximum Limits

The Ffollowing table covers the maximum limit defined in CNC Console configuration.

Table 1-4 CNC Console Configuration Maximum Limits

Attribute Name	Max Limit
Max length of cnccId	40
Max length of instanceId	80
Max number of A-CNCC (aCnccs)	36
Max number of instances	288
Max number of M-CNCC (mCnccs)	3

CnDBTier Usage Guidelines

This section explains the guidelines for cnDBTier usage.

- For managing multiple instances, console must have dedicated cnDBTier.
- For managing single instance, console can share cnDBTier with NF.

Deployment	cnDBTier Usage	Comments
Single Cluster Single Instance	Console and NF have a single shared cnDBTier	Manager IAM, Manager Core, and Agent Core on same k8s cluster use shared cnDBTier.
Single Cluster Multi Instance	Dedicated cnDBTier for Console	Manager IAM, Manager Core, and Agent Core on same k8s cluster use single Console cnDBTier.
Multi Cluster Single Instance	Dedicated cnDBTier for Console per Kubernetes cluster	• Manager IAM, Manager Core and Agent Core on same Kubernetes cluster use single Console cnDBTier. • Agent Core on a remote Kubernetes cluster uses dedicated single Console cnDBTier.
Multi Cluster Multi Instance	Dedicated cnDBTier for Console per Kubernetes cluster	• Manager IAM, Manager Core and Agent Core on same Kubernetes cluster use single Console cnDBTier • Agent Core on a remote Kubernetes cluster uses dedicated single Console cnDBTier

Note:

- Time synchronization is required between kubernetes nodes across cluster for functioning of CNCC security procedures.
- Ensure NTP sync before proceeding with M-CNCC IAM, M-CNCC Core, and A-CNCC Core installation.

Reference

Refer to the following documents for more information:

- *Oracle Communications Cloud Native Core Service Communication Proxy User Guide*
- *Oracle Communications Cloud Native Core Network Repository Function User Guide*
- *Oracle Communications Cloud Native Core Policy User Guide*
- *Oracle Communications Cloud Native Core Unified Data Repository User Guide*
- *Oracle Communications Cloud Native Core Binding Support Function User Guide*

- *Oracle Communications Cloud Native Core Security Edge Protection Proxy User Guide*
- *Oracle Communications Cloud Native Core Network Slice Selection Function User Guide*
- *Oracle Communications Cloud Native Core Network Repository Function Installation and Upgrade Guide*
- *Oracle Communications Cloud Native Core Service Communication Proxy Installation Guide*
- *Oracle Communications Cloud Native Core Unified Data Repository Installation and Upgrade Guide*
- *Oracle Communications Cloud Native Core Binding Support Function Installation Guide*
- *Oracle Communications Cloud Native Core Policy Installation Guide*
- *Oracle Communications Cloud Native Core Security Edge Protection Proxy Installation Guide*
- *Oracle Communications Cloud Native Core Network Slice Selection Function Installation Guide*

2
CNC Console Installation Preparation

Prerequisites

Before installing Oracle Communications CNC Console, make sure that the following requirements are met:

Software Requirements

This section lists the minimum software requirements to install Oracle Communications CNC Console.

The CNC Console software includes:

- CNC Console Helm charts
- CNC Console docker images

Install the following software before installing CNC Console:

Table 2-1 Pre-installed Software

Software	Version
Kubernetes	1.23.x, 1.22.x, 1.21.x
HELM	v3.1.2, v3.5.0, v3.6.3, v3.8.0
Podman	v2.2.1, v3.2.3, v3.3.1

Oracle Communications Cloud Native Environment Requirement

CNC Console supports Oracle Communications Cloud Native Environment (OCCNE). To check the OCCNE version, run the following command:

```
echo $OCCNE_VERSION
```

To check the current helms and Kubernetes version installed in the OCCNE, run the following commands:

```
kubectl version
```

```
helm version
```

Following are the common services that must be deployed as per the requirement:

Table 2-2 Common Services

Software	Chart Version	Required For
elasticsearch	7.9.3	Logging
elastic-curator	5.5.4	Logging
elastic-exporter	1.1.0	Logging
elastic-master	7.9.3	Logging
logs	3.1.0	Logging
kibana	7.9.3	Logging
grafana	7.5.11	Metrics
prometheus	2.32.1	Metrics
prometheus-kube-state-metrics	1.9.7	Metrics
prometheus-node-exporter	1.0.1	Metrics
metallb	0.12.1	External IP
metrics-server	0.3.6	Metric Server
tracer	1.12.0	Tracing

Note:

The common services are available if the CNC Console is deployed in the Oracle Communications Cloud Native Environment (OCCNE). If you are deploying CNC Console in any other environment, the common services must be installed before installing the CNC Console.
To check the installed software items, run the following command:

```
helm3 ls -A
```

Environment Setup Requirements

This section provides information on environment setup requirements for installing CNC Console.

Network Access

The Kubernetes cluster hosts must have network access to:

- Local docker image repository, where the CNC Console images are available.
  To check if the Kubernetes cluster hosts have network access to 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>
  ```

  where:

  docker-repo is the IP address or host name of the repository.

image-name is the docker image name.

image-tag is the tag the image used for the CNC Console pod.

- Local helm repository, where the CNC Console 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
  ```

> **Note:**
>
> All the kubectl and helm related commands that are used in this document must be run on a system depending on the infrastructure of the deployment. It could be a client machine such as a VM, server, local desktop, and so on.

Client Machine Requirement

Client machine must meet the following requirements:

- Network access to the helm repository and docker image repository.
- Helm repository must be configured on the client.
- Network access to the Kubernetes cluster.
- Necessary environment settings to run the `kubectl` and `docker` commands. The environment should have privileges to create a namespace in the Kubernetes cluster.
- Helm client must be installed. The environment should be configured so that the `helm install` command deploys the software in the Kubernetes cluster.

Server or Space Requirement

For information on the server or space requirements, see the *Oracle Communications Cloud Native Environment (OCCNE) Installation Guide*.

OSO Requirement

CNC Console supports OSO for common operation services (Prometheus and components such as alertmanager, pushgateway) for Kubernetes cluster which does not have these common services. For more information on installation procedure, see *Oracle Communications OSO Installation Guide*.

cnDBTier Requirement

CNC Console supports cnDBTier in a vCNE environment. cnDBTier must be up and active in case of containerized CNE. For more information on installation procedure, see *Oracle Communications cnDBTier Installation guide.*

CNC Console Resource Requirement

This section includes information about CNC Console Resource Requirement.

CNC Console Deployment Resource Usage
Resource usage for CNC Console Single Cluster and Multi Cluster deployment is listed in the following tables.

CNC Console Single Cluster Deployment Resource Usage
Single Cluster Deployment will include M-CNCC IAM, M-CNCC Core and A-CNCC Core components.

CNC Console Common Resource is a common resource needed for manager or agent deployment.

Table 2-3 CNC Console Single Cluster Deployment Resource Usage

Component	Limits		Requests	
	CPU	Memory (Gi)	CPU	Memory (Gi)
M-CNCC IAM	7.5	7.5	3.8	3.8
M-CNCC Core	7	7	3.5	3.5
A-CNCC Core	7	7	3.5	3.5
CNCC Common Resource	3	4	1.5	2
Total	24.5	25.5	12.3	12.8

Formula

Total Resource = M-CNCC IAM Resource + M-CNCC Core Resource + A-CNCC Core Resource + CNCC Common Resource

CNC Console Multi Cluster Deployment Resource Usage
Multi Cluster Deployment will include M-CNCC IAM and M-CNCC Core components in Manager cluster. A-CNCC Core component shall be deployed in Manager cluster if there is a local NF.

A-CNCC Core is needed in each Agent cluster for managing local NF. CNCC Common Resource is a common resource needed for manager or agent deployment.

Table 2-4 CNC Console Multi Cluster Deployment Resource Usage

Component	Limits		Requests	
	CPU	Memory (Gi)	CPU	Memory (Gi)
M-CNCC IAM	7.5	7.5	3.8	3.8
M-CNCC Core	7	7	3.5	3.5
A-CNCC Core	7	7	3.5	3.5
CNCC Common Resource	3	4	1.5	2
*No Of Agents In Other Clusters	2			
Total	37.5	40.5	18.8	20.3

* Assumed number of Agents (A-CNCC Core deployments) for the calculation

Formula

Total Resource = M-CNCC IAM Resource + M-CNCC Core Resource + Common Resources + (No Of Agents In Other Clusters * (CNCC Common Resource + A-CNCC Core Resource))

CNC Console Manager Only Deployment

The following table shows resource requirement for manager only deployment. In this case, agent will be deployed in separate cluster.

Component	Limits		Requests	
	CPU	Memory (Gi)	CPU	Memory (Gi)
M-CNCC IAM	7.5	7.5	3.8	3.8
M-CNCC Core	7	7	3.5	3.5
A-CNCC Core	0	0	0	0
CNCC Common Resource	3	4	1.5	2
Total	17.5	18.5	8.8	9.3

CNC Console Agent Only Deployment
The following table shows resource requirement for agent only deployment, in this case manager will be deployed in separate cluster.

Table 2-5 CNC Console Agent Only Deployment

Component	Limits		Requests	
	CPU	Memory (Gi)	CPU	Memory (Gi)
M-CNCC IAM	0	0	0	0
M-CNCC Core	0	0	0	0
A-CNCC Core	7	7	3.5	3.5
CNCC Common Resource	3	4	1.5	2
Total	10	11	5	5.5

CNC Console Manager with Agent Deployment
The following table shows resource requirement for manager with agent deployment, in this case agent will be deployed along with manager to manage local NF.

This manager can manage agents deployed in other clusters.

Table 2-6 CNC Console Manager with Agent Deployment

Component	Limits		Requests	
	CPU	Memory (Gi)	CPU	Memory (Gi)
M-CNCC IAM	7.5	7.5	3.8	3.8
M-CNCC Core	7	7	3.5	3.5
A-CNCC Core	7	7	3.5	3.5
CNCC Common Resource	3	4	1.5	2
Total	24.5	25.5	12.3	12.8

CNC Console Component wise Resource Usage

Table 2-7 CNCC Common Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
debug_tools	tools	1	2	0.5	1	Applicable when debug_tool is enabled
hookJobResources		2	2	1	1	Common Hook Resource
helm test	cncc-test					Uses hookJobResources
Total		3	4	1.5	2	

Table 2-8 M-CNCC IAM Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
cncc-iam-ingress-gateway	ingress-gateway	2	2	1	1	
	init-service	1	1	0.5	0.5	Applicable when HTTPS is enabled
	update-service	1	1	0.5	0.5	Applicable when HTTPS is enabled
	common_config_hook					common_config_hook not used in IAM
cncc-iam-kc-http	kc	2	2	1	1	
	init-service	1	1	0.5	0.5	Optional, used for enabling LDAPS
	healthcheck	0.5	0.5	0.3	0.3	
	cnnc-iam--pre-install					Uses hookJobResources
	cnnc-iam-pre-upgrade					Uses hookJobResources
	cnnc-iam-post-install					Uses hookJobResources

Table 2-8 (Cont.) M-CNCC IAM Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
	cnnc-iam-post-upgrade					Uses hookJobResources
Total		7.5	7.5	3.8	3.8	

Table 2-9 M-CNCC Core Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
cncc-mcore-ingress-gateway	ingress-gateway	2	2	1	1	
	init-service	1	1	0.5	0.5	Applicable when HTTPS is enabled
	update-service	1	1	0.5	0.5	Applicable when HTTPS is enabled
	common_config_hook	1	1	0.5	0.5	Common Configuration Hook container creates databases which are used by Common Configuration Client
cncc-mcore-cmservice	cmservice	2	2	1	1	
	validation-hook					Uses common hookJobResources
Total		7	7	3.5	3.5	

Table 2-10 A-CNCC Core Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
cncc-acore-ingress-gateway	ingress-gateway	2	2	1	1	

Table 2-10 (Cont.) A-CNCC Core Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
	init-service	1	1	0.5	0.5	Applicable when HTTPS is enabled
	update-service	1	1	0.5	0.5	Applicable when HTTPS is enabled
	common_config_hook	1	1	0.5	0.5	Common Configuration Hook container creates databases which are used by Common Configuration Client
cncc-acore-cmservice	cmservice	2	2	1	1	
	validation-hook					Uses common hookJobResources
Total		7	7	3.5	3.5	

CNC Console Microservices Resource Requirement

Table 2-11 CNC Console Microservices Resource Requirement

Microservice Name	CPU Per Pod	Memory Per Pod (GB)	Pod count	CPU All Pods - Maximum	Memory All Pods - Maximum (GB)
	Maximum	Maximum	Maximum		
#cncc-iam-kc-http	&$!2.5	&$!2.5	1	2.5	2.5
#cncc-iam-ingress-gateway	^$2	^$2	1	2	2
#cncc-core-cmservice	$2	$2	1	2	2
#cncc-core-ingress-gateway	^$2	^$2	1	2	2
Total				8.5	8.5

- #: <helm release name>→ will be prefixed in each Microservice name. Example: if helm release name is "cncc-iam", then ingress-gateway Microservice name will be "cncc-iam-ingress-gateway"

- ^: CPU Limit/Request Per Pod and Memory Limit/Request Per Pod needs to added as additional resources for init-service and update-service container if TLS needs to be enabled.

  Init-service container's and Common Configuration Client Hook's resources are not counted because the container gets terminated after initialization completes.

Container Name	CPU Request and Limit Per Container	Memory Request and Limit Per Container	Kubernetes Init Container (Job)
init-service	1 cpu	1 gb	Yes
update-service	1 cpu	1 gb	No
common_config_hook	1 cpu	1 gb	No

  – Update Container service

  Ingress Gateway: For periodically refreshing CNCC Private Key/ Certificate and CA Root Certificate for TLS

  – Init Container service

    Ingress Gateway: To get CNCC Private Key or Certificate and CA Root Certificate for TLS during start up

  – Common Configuration Hook

    CNCC Core Common configuration hook container creates databases which are used by Common Configuration Client

- &: Helm Hooks Jobs - These jobs are pre and post jobs which are invoked during installation, upgrade, rollback, and uninstallation of the deployment. These are short-lived jobs which get terminated after the work is done. So, they are not part of Active deployment Resource but need to be considered only during installation, upgrade, rollback, and uninstallation procedures.

Container Type	CPU Request and Limit Per Container	Memory Request and Limit Per Container
Helm Hooks	Request - 1 cpu, Limit - 2 cpu	Request - 1 gb, Limit - 2 gb

! : Healthcheck Container Service- For monitoring health of the db, extra container is added to the kc pod. This is part of active deployment so additional resource of 0.5 is considered as part of kc pod.

Container Type	CPU Request and Limit Per Container	Memory Request and Limit Per Container
Health Check	Request - 0.3 cpu, Limit - 0.5 cpu	Request - 0.3 gb, Limit - 0.5 gb

- Helm Test Job - This job is run on demand when helm test command is executed. It executes the helm test and stops after completion. These are short-lived jobs which gets terminated after the work is done. So, they are not part of active deployment Resource, but needs to be considered only during helm test procedures.

Container Type	CPU Request and Limit Per Container	Memory Request and Limit Per Container
Helm Test	Request - 1 cpu, Limit - 2 cpu	Request - 1 gb, Limit - 2 gb

$ - Troubleshooting Tool Container - If Troubleshooting Tool Container Injection is enabled during CNCC deployment/upgrade, this container will be injected to each CNCC pod (or selected pod, depends on option chosen during deployment/upgrade). These containers will stay till pod/deployment exists.

Container Name	CPU Request and Limit Per Container	Memory Request and Limit Per Container	ephemeral-storage Request and Limit Per Container
ocdebug-tools	Request - 0.5 cpu, Limit - 1 cpu	Request - 1 gb, Limit - 2 gb	Request - 2 gb, Limit - 4 gb

Downloading CNC Console Package

Perform the following procedure to download the CNC Console release package from My Oracle Support:

1. Log in to MOS using the appropriate credentials.
2. Click the **Patches & Updates** tab to locate the patch.
3. In the **Patch Search** console, 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.
5. Select Oracle Communications Cloud Native Core <release number> in **Release** field.
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**. File Download window appears.
9. Click the <p******_<release number>_Tekelec>.zip file.
10. Extract the release package zip file.
    Package is named as follows:

    ```
    occncc_pkg_<marketing-release-number>.tgz
    ```

    Example:

    ```
    occncc_pkg_22_3_2_0_0.tgz
    ```

    > **Note:**
    >
    > The user must have their own repository for storing the CNC Console images and repository which must be accessible from the Kubernetes cluster.

11. Untar the CNCC package file:

Untar the CNCC package file to the specific repository:

```
tar -xvf occncc_pkg_<marketing-release-number>.tgz
```

The package file consists of the following:

a. CNCC Docker Images File: `occncc-images-<tag>.tar`

b. Helm Chart of CNCC: The tar ball contains Helm Chart and templates *occncc-<tag>.tgz*

c. Readme txt File `Readme.txt`

Example :

List of contents in occncc_pkg_22_3_2_0_0.tgz :

occncc_pkg_22_3_2_0_0.tgz

|_ _ _ _ _ _ occncc-22.3.2 tgz

|_ _ _ _ _ _ occncc-images-22.3.2 tar

|_ _ _ _ _ _ Readme.txt

12. Verify the checksums of tarballs mentioned in `Readme.txt`

13. Run the following command to push the Docker images to docker repository:

```
docker load --input <image_file_name.tar>
Example
 docker load --input occncc-images-22.3.2.tar

Sample Output:
Loaded image: occncc/apigw-configurationinit:22.3.2
Loaded image: occncc/apigw-configurationupdate:22.3.2
Loaded image: occncc/apigw-common-config-hook:22.3.2
Loaded image: occncc/cncc-apigateway:22.3.2
Loaded image: occncc/cncc-cmservice:22.3.2
Loaded image: occncc/cncc-iam:22.3.2
Loaded image: occncc/debug_tools:22.3.2
Loaded image: occncc/nf-test:22.3.2
Loaded image: occncc/cncc-iam/hook:22.3.2
Loaded image: occncc/cncc-iam/healthcheck:22.3.2
Loaded image: occncc/cncc-core/validationhook:22.3.2
```

14. 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>
```

15. Run the following command to check whether all the images are loaded:

```
docker images
```

16. Run the following command to push the helm charts to helm repository:

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

    Example:

    ```
    helm cm-push --force occncc-22.3.2.tgz ocspf-helm-repo
    ```

17. Run the following command to download the CNCC custom templates package file from MOS:

    ```
    occncc_custom_configtemplates_<marketing-release-number>.zip
    ```

    Example:

    ```
    occncc_custom_configtemplates_22_3_2_0_0.zip
    ```

18. Unzip the cncc custom templates package as follows:

    ```
    unzip occncc_custom_configtemplates_<marketing-release-number>.zip
    ```

    The package file consists of the following:

    ```
    M-CNCC IAM custom values file : occncc_custom_values_<version>.yaml
    CNCC IAM Schema file for rollback to previous version :
    occncc_rollback_iam_schema_<version>.sql
    CNCC Metric Dashboard file : occncc_metric_dashboard_<version>.json
    CNCC Metric Dashboard file for CNE supporting Prometheus HA (CNE
    1.9.x onwards): occncc_metric_dashboard_promha_<version>.json
    CNCC Alert Rules file: occncc_alertrules_<version>.yaml
    CNCC Alert Rules file for CNE supporting Prometheus HA (CNE 1.9.x
    onwards): occncc_metric_dashboard_promha_<version>.json
    CNCC MIB files: occncc_mib_<version>.mib,
    occncc_mib_tc_<version>.mib
    ```

    Example:

    ```
    unzip occncc_custom_configtemplates_<marketing-release-number>.zip
    ```

    ```
      Archive:  occncc_custom_configtemplates_22_3_2_0_0.zip
       creating: occncc_custom_configtemplates_22_3_2_0_0/
      inflating: occncc_custom_configtemplates_22_3_2_0_0/
    occncc_mib_tc_22.3.2.mib
      inflating: occncc_custom_configtemplates_22_3_2_0_0/
    occncc_custom_values_22.3.2.yaml
      inflating: occncc_custom_configtemplates_22_3_2_0_0/
    occncc_mib_22.3.2.mib
    ```

```
  inflating: occncc_custom_configtemplates_22_3_2_0_0/
occncc_alerting_rules_promha_22.3.2.yaml
  inflating: occncc_custom_configtemplates_22_3_2_0_0/
occncc_metric_dashboard_promha_22.3.2.json
  inflating: occncc_custom_configtemplates_22_3_2_0_0/
occncc_alertrules_22.3.2.yaml
  inflating: occncc_custom_configtemplates_22_3_2_0_0/
occncc_metric_dashboard_22.3.2.json
  inflating: occncc_custom_configtemplates_22_3_2_0_0/
occncc_rollback_iam_schema_22.3.2.sql
```

3
Installing CNC Console

This section describes the prerequisites and installation procedure for the CNC Console.

CNC Console (cncc) helm chart is a common helm chart to be used for deploying Manager CNCC IAM (M-CNCC IAM), Manager CNCC Core (M-CNCC Core), and Agent CNCC Core (A-CNCC Core).

The package `occncc_custom_configtemplates_<marketing-release-number>.zip` consists of `occncc_custom_values_<version>.yaml` file and can be used for deployment of M-CNCC IAM, M-CNCC Core and A-CNCC Core.

This section covers installation instructions for M-CNCC IAM, M-CNCC Core and A-CNCC Core deployment.

Note:

- For a single cluster deployment both manager (M-CNCC IAM, M-CNCC Core) and agent(A-CNCC Core) to be deployed on the same cluster.
- For a multi cluster deployment if manager cluster has a local NF deployment then both manager(M-CNCC IAM, M-CNCC Core) and agent(A-CNCC Core) to be deployed on the same cluster. In case manager cluster do not have a local NF deployment then only manager(M-CNCC IAM, M-CNCC Core) to be deployed and agent(A-CNCC Core) to be deployed on a cluster where NFs are present on the cluster.
- Manager manages CNE or OSO common service if present in a cluster.
  - Manager in a cluster is preferred over Agent in the same cluster to manage the CNE common services.
  - Agent in a cluster can manage CNE common service in absence of a Manager in the same cluster.
- Agent is needed only when NFs are present on the cluster.

CNC Console Predeployment Configurations

Following are the predeployment configuration procedures:

Verifying and Creating CNC Console Namespace

This section explains how to verify or create new namespace in the system.

To create a namespace, perform the steps described in the procedure below:

> **Note:**
>
> To install CNC Console in NF specific namespace, replace cncc namespace with custom namespace.

To verify if the required namespace already exists in the system, run the following command:

```
$ kubectl get namespaces
```

If the namespace exists, you may continue with the next steps of installation.

If the required namespace is not available, create a namespace using the following command:

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

Example:

```
$ kubectl create namespace cncc
```

Sample output:

```
 namespace/cncc created
```

Naming Convention for Namespaces

While choosing the name of the namespace where you wish to deploy CNC Console, make sure the following requirements are met:

- starts and ends with an alphanumeric character
- Contains a maximum of "63" characters
- contains only alphanumeric characters or '-'

> **Note:**
>
> It is recommended to avoid using prefix `kube-` when creating namespace. This is required as the prefix is reserved for Kubernetes system namespaces.

> **Note:**
>
> For the information about extra privileges required to enable Debug tools, see the steps mentioned in the section CNCConsole Debug Tools.

Configuring Database

This section explains how database administrator can configure the databases.

> **Caution:**
>
> Verify the value of the following parameters, before deploying CNC Console on a cnDBTier 3 site georedundency setup:
>
> ndb:
>
> MaxNoOfOrderedIndexes: 1024
>
> MaxNoOfTables: 1024
>
> NoOfFragmentLogFiles: 512
>
> If the values are not matching, see *Oracle communications cnDBTier Installation Guide* for details on how to update these parameters.

Configuring the Database User

This section explains how to create or verify the existing cncc user.

1. Log in to the server or machine which has permission to access the SQL nodes of the NDB cluster.
2. Connect to the SQL node of the NDB cluster or connect to the cnDbTier.
   Run the following command to connect to cnDbTier:

   ```
   $ kubectl -n <cndbtier_namespace> exec -it <cndbtier_sql_pod_name> -c
   <cndbtier_sql_container_name> -- bash
   ```

   Example:

   ```
   $ kubectl -n cndbtier  exec -it ndbmysqld-0 -c mysqlndbcluster -- bash
   ```

3. Log in to the MySQL prompt using root permission or user, which has permission to create users with permissions.

   ```
   mysql -h 127.0.0.1 -u root -p
   ```

   > **Note:**
   >
   > Provide MySQL password, when prompted.

4. Check if CNCC user already exists by running the following command:

```
$ SELECT User FROM mysql.user;
```

5. If the user does not exist, create a cncc-user by running the following command:

```
$ CREATE USER '<CNCC User Name>'@'%' IDENTIFIED BY '<CNCC
Password>';
```

6. Run the following command to grant NDB_STORED_USER permission to the cncc-user:

```
$ GRANT NDB_STORED_USER ON *.* TO '<username>'@'%' WITH GRANT
OPTION;
```

Example:

```
$ GRANT NDB_STORED_USER ON *.* TO 'cnccusr'@'%' WITH GRANT OPTION;
```

Configuring MySQL Secret

This section explains how to create and verify the kubernetes secret for MySQL.

1. Run the following command to create the kubernetes secret for MySQL:

```
kubectl create secret generic <database secret name> --from-
literal=dbUserNameKey=<CNCC MySQL database username> --from-
literal=dbPasswordKey='<CNCC MySQL database passsword>' -n
<Namespace of MySQL secret>
```

2. Verify the secret created using following command:

```
kubectl describe secret <database secret name> -n <Namespace of
MYSQL secret>
```

Example:

```
$ kubectl create secret generic cncc-db-secret --from-
literal=dbUserNameKey=cnccusr --from-literal=dbPasswordKey='<db
password>' -n cncc
$ kubectl describe secret cncc-db-secret -n cncc
```

Global Configurations

CNC Console Deployment Configurations

This section explains about the cncc global configurations which are common for M-CNCC IAM, M-CNCC Core, and A-CNCC Core deployment.

Table 3-1 CNC Console Deployment Configurations

Deployment	isMultiClusterDeployment enabled	cncc-iam enabled	mcncc-core enabled	acncc-core enabled
Single Cluster	false	true	true	true
Multi Cluster(manager only)	true	true	true	false
Multi Cluster(managing local NF's)	true	true	true	true
Multi cluster(agent only)	true	false	false	true

Example: Configuration in *occncc_custom_values_<version>.yaml* file.

In case of single cluster deployment(*global.isMultiClusterDeployment: false*) the user must configure cncc-iam, mcncc-core and acncc-core flag to true.

```
global:

  isMultiClusterDeployment: false

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true
```

CNC Console Configuration for Service Account

Global Service Account Configuration

CNC Console provides option to configure custom service account. Skip this, if already created as part of ASM configuration.

Sample CNCC service account yaml file

```
## Service account yaml file for cncc-sa
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cncc-sa
  namespace: cncc
  annotations: {}
---
apiVersion: rbac.authorization.k8s.io/v1
```

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

Configure service account for ingress-gateway and keycloak in *occncc_custom_values_<version>.yaml* file as follows:

For Ingress Gateway and Keycloak provide custom service account under **global.serviceAccountName.**

```
global:

  serviceAccountName: &serviceAccountName cncc-sa

cncc-iam:
  kc:
    keycloak:
      serviceAccount:
        # The name of the service account to use.
        name: *serviceAccountName
```

Helm Test Service Account Configuration

Helm Test Service Account is an optional field. It should be added only if helm test kubernetes resource logging feature is enabled.

Custom service account can be provided for helm test in `global.helmTestServiceAccountName`:

```
global:

  # ********    Sub-Section Start: Common Global Parameters        ********
  # ************************************************************************

  helmTestServiceAccountName: cncc-helmtest-serviceaccount
```

Sample helm test service account : cncc-helmtest-serviceaccount.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cncc-helmtest-serviceaccount
  namespace: cncc
  annotations: {}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: cncc-helmtest-role
  namespace: cncc
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
  - services
  - configmaps
  - pods
  - secrets
  - endpoints
  - persistentvolumeclaims
  - serviceaccounts
  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: cncc-helmtest-rolebinding
  namespace: cncc
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: cncc-helmtest-role
subjects:
- kind: ServiceAccount
  name: cncc-helmtest-serviceaccount
  namespace: cncc
```

> **Note:**
>
> If user doesn't want to create the separate service account for helm test then also logging of resources and complaint check could be done using global service account . For that, required resources should be added to the global service account. For more details on helmTestserviceAccount, see Helm Test section.

Configuring ASM and OSO in CNC Console

For M-CNCC IAM ASM and OSO Configurations, see CNC Console Configuration to support ASM and OSO.

M-CNCC IAM Predeployment Configuration

The following are the predeployment configuration procedures of M-CNCC IAM:

Configuring M-CNCC IAM Database

This section explains how to create M-CNCC IAM Database and grant permissions to the CNC Console user for relevant operations on the Database.

> **Note:**
>
> **CNCC Database Naming Convention**
>
> As the CNCC instances cannot share the same database, user must provide a unique name to the CNCC DB in the cnDBTier either limited to a site or spanning across sites.
>
> The recommended format for database name is as follows:
>
> ```
> <database-name>_<site-name>_<cluster>
> ```
>
> Example:
>
> ```
> cnccdb_site1_cluster1
> ```
>
> While choosing the name of the database, make sure the following requirements are met:
>
> - starts and ends with an alphanumeric character
> - contains a maximum of 63 characters
> - contains only alphanumeric characters or '-'
>
> In this guide, the commands use "cnccdb" as sample database name. If a custom database name is used, user must use it in place of cnccdb.

1. Log in to the server or machine with permission to access the SQL nodes of NDB cluster.
2. Connect to the SQL node of the NDB cluster or connect to the cnDBTier.

Run the following command to connect to the cnDBTier:

```
$ kubectl -n <cndbtier_namespace> exec -it <cndbtier_sql_pod_name> -
c <cndbtier_sql_container_name> -- bash
```

Example:

```
$ kubectl -n cndbtier  exec -it ndbmysqld-0 -c mysqlndbcluster --
bash
```

3. Run the following command to log in to the MySQL prompt as a user with root permissions, which has permission to create users with permissions:

```
mysql -h 127.0.0.1 -u root -p
```

> **Note:**
>
> Enter the above command, and use the MYSQL password.

4. Check if CNCC database already exists:
   a. Run the following command to check if database exists:

      ```
      show databases;
      ```

   b. Run the following command to drop the existing cnccdb database:

      ```
      DROP DATABASE <CNCC IAM Database>
      ```

      Example:

      ```
      DROP DATABASE cnccdb;
      ```

   c. Run the following command to create the Database:

      ```
      $ CREATE DATABASE IF NOT EXISTS <CNCC IAM Database>;
      ```

   d. Run the following command to grant permission to cncc-user:

```
$ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES,
REFERENCES, INDEX, CREATE TEMPORARY TABLES, DELETE, UPDATE, EXECUTE
ON <M-CNCC IAM Database>.* TO'<CNCC IAM DB User Name>'@'%';
```

Example to demonstrate cnccdb creation, and granting permissions to cncc user:

```
# Command to check if database exists:
$ show databases;
# Drop the existing cnccdb database:
```

```
$ DROP DATABASE cnccdb;
# Database creation for CNCC:
$ CREATE DATABASE IF NOT EXISTS cnccdb;
#Granting permission to user:
$ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, REFERENCES, INDEX,
CREATE TEMPORARY TABLES, DELETE, UPDATE, EXECUTE ON cnccdb .*
TO'cnccusr'@'%';
```

Configuring Secret for Default or Admin User in M-CNCC IAM

This section describes how to create kubernetes secret for default (admin) user.

1. Run the following command to create the Kubernetes secret for admin user:

   ```
   $ kubectl create secret generic <secret-name> --from-
   literal=iamAdminPasswordKey='<password>'--namespace <namespace>
   ```

   Example:

   ```
   $ kubectl create secret generic cncc-iam-secret --from-
   literal=iamAdminPasswordKey='password' --namespace cncc
   ```

2. Run the following command to verify the secret creation:

   ```
   $ kubectl describe secret <secret name> -n <namespace>
   ```

   Example:

   ```
   $ kubectl describe secret cncc-iam-secret -n cncc
   ```

Configuring Secret to Enable HTTPS in M-CNCC IAM

This section describes how create and configure secrets to enable HTTPS. This section must be run before Configuring Secret to Enable HTTPS CNCC Core Ingress Gateway.

> **Note:**
>
> The passwords for TrustStore and KeyStore are stored in respective password files.

Prerequisites to create kubernetes secret for HTTPS:

- ECDSA private key and CA signed certificate of CNCC (if initialAlgorithm is ES256)
- RSA private key and CA signed certificate of CNCC (if initialAlgorithm is RSA256)
- TrustStore password file
- KeyStore password file
- CA certificate/ CA Bundle

CA Bundle creation

When combining multiple CA certificates into a single certificate, add a delimiter after each certificate.

Delimiter: "--------"

Sample CA Bundle

```
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
--------
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
--------
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
```

Perform the following procedure to create the secrets to enable HTTPS after required certificates and password files are generated:

1. Run the following command to create secret:

```
$ kubectl create secret generic <secret-name> --from-
file=<ssl_ecdsa_private_key.pem> --from-
file=<rsa_private_key_pkcs1.pem> --from-file=<ssl_truststore.txt> --
from-file=<ssl_keystore.txt> --from-file=<caroot.cer> --from-
file=<ssl_rsa_certificate.crt> --from-
file=<ssl_ecdsa_certificate.crt> -n <Namespace of CNCC IAM Ingress
Gateway secret>
```

Example:

```
$ kubectl create secret generic cncc-iam-ingress-secret --from-
file=ssl_ecdsa_private_key.pem  --from-
file=rsa_private_key_pkcs1.pem --from-file=ssl_truststore.txt --
from-file=ssl_keystore.txt --from-file=caroot.cer --from-
file=ssl_rsa_certificate.crt --from-file=ssl_ecdsa_certificate.crt -
n cncc
```

> **Note:**
>
> Note down the command used during the creation of Kubernetes secret as this command is used for future updates.

2. On successfully running the above command, the following message is displayed:
   secret/cncc-iam-ingress-secret created
3. Run the following command to verify the secret creation:

```
$ kubectl describe secret cncc-iam-ingress-secret -n cncc
```

Perform the following procedure to update existing secrets to enable HTTPS, if they already exist:

1. Run the following command to create secret:

```
$ kubectl create secret generic <secret-name> --from-
file=<ssl_ecdsa_private_key.pem> --from-file=<rsa_private_key_pkcs1.pem>
--from-file=<ssl_truststore.txt> --from-file=<ssl_keystore.txt> --from-
file=<caroot.cer> --from-file=<ssl_rsa_certificate.crt> --from-
file=<ssl_ecdsa_certificate.crt> --dry-run -o yaml -n <Namespace of CNCC
IAM Ingress Gateway secret> | kubectl replace -f --n <Namespace of CNCC
IAM Ingress Gateway secret>
```

   Example:

```
$ kubectl create secret generic cncc-iam-ingress-secret --from-
file=ssl_ecdsa_private_key.pem --from-file=rsa_private_key_pkcs1.pem --
from-file=ssl_truststore.txt --from-file=ssl_keystore.txt --from-
file=caroot.cer --from-file=ssl_rsa_certificate.crt --from-
file=ssl_ecdsa_certificate.crt --dry-run -o yaml -n cncc | kubectl
replace -f --n cncc
```

2. On successfully running the above command, the following message is displayed:
   secret/cncc-iam-ingress-secret replaced

Dynamic Reloading of Certificates of M-CNCC IAM

Perform the following procedure for configuring CNC Console IAM to Support Dynamic Reloading of Certificates of M-CNCC IAM:

CNC Console supports Dynamic Reload of Certificates that are used to establish both TLS and mTLS connections.

To enable dynamic reloading of certificates, the following flags must be enabled in the *occncc_custom_values_<version>.yaml* file.

```
cncc-iam:
 global:
   ingressGwCertReloadEnabled:
              &iamIGwCertReloadEnabled true
```

> **Note:**
>
> The new certificate must be created with the existing secret and certificate name.

The procedure for dynamic reloading of certificates as follows:

1. Delete the existing certificates with which existing secure connections were established.
2. Create the new certificate as per the requirement. The certificate must be created with the same name as the existing certificate.
3. The Ingress Gateway pods automatically pick up the new certificates and the changes will be reflected in the browser.

> **Note:**
>
> **Naming Update of Certificates and Secrets**
>
> If the name of the secret or the certificate is changed then the corresponding changes must be made in the *occncc_custom_values_<version>.yaml* file and either a reinstall or a helm upgrade must be done.

Configuring LDAPS in M-CNCC IAM

This section explains the procedure to enable LDAPS in M-CNCC IAM.

When you configure a secured connection URL to your LDAP store (Example: `ldaps://myhost.com:636'), CNCC IAM uses SSL for communication with the LDAP server. The truststore must be properly configured on the CNCC IAM server side; otherwise CNCC IAM cannot trust the SSL connection to LDAP.

For enabling LDAPS update kc section of *occncc_custom_values_<version>.yaml* as below:

```
cncc-iam
               kc:
                 ldaps:
                     enabled: true
```

M-CNCC IAM Secret configuration for enabling LDAPS

> **Note:**
>
> The passwords for TrustStore and KeyStore are stored in respective password files.

To create Kubernetes secret for LDAPS, following files are required:

- TrustStore password file
- CA certificate or CA Bundle

Creating CA Bundle

When combining multiple CA certificates into a single certificate, add a delimiter after each certificate.

Delimiter: "--------"

Sample CA Bundle

```
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
--------
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
```

> **Note:**
>
> Creation process for private keys, certificates and passwords is on discretion of user.

Perform the following procedure to create the secrets to enable LDAPS after required certificates and password files are generated and update details in kc section:

> **Note:**
>
> The value of *ssl_truststore.txt* and *ssl_truststore-password-key* value must be same.

1. Run the following command to create secret:

   ```
   kubectl create secret generic <secret-name> --from-file=<caroot.cer> --
   from-file=ssl_truststore.txt --from-literal=ssl_truststore-password-
   key=<password> --namespace cncc
   ```

   > **Note:**
   >
   > Note down the command used during the creation of Kubernetes secret as this command is used for future updates.

   Example:

   ```
   $ kubectl create secret generic cncc-iam-kc-root-ca --from-
   file=caroot.cer --from-file=ssl_truststore.txt --from-
   literal=ssl_truststore-password-key=<password> --namespace cncc
   ```

Run the following to display the sample ssl_truststore.txt:

```
echo <password> > ssl_truststore.txt
```

2. On successfully running the above command, the following message is displayed: *secret/cncc-iam-kc-root-ca created*
3. Run the following command to verify the secret creation:

```
$ kubectl describe secret cncc-iam-kc-root-ca -n cncc
```

M-CNCC IAM Service Account configuration for enabling LDAPS

This section describes the customizations that you should make in custom-value.yaml file to configure Kubernetes service account. M-CNCC IAM provides option to configure custom service account.

> **Note:**
>
> Skip this section if service account is already configured as part of HTTPS or ASM configuration.

Configure service account for ingress-gateway and keycloak in *custom-cncc_values_<version>.yaml* as follows:

1. For ingress-gateway provide custom service account under *global.serviceAccountName.*

```
global:

  serviceAccountName: &serviceAccountName cncc-sa

cncc-iam:
  kc:
    keycloak:
      serviceAccount:
        # The name of the service account to use.
        name: *serviceAccountName
```

For CNCC IAM LDAP related configurations, see *Integrating CNC Console LDAP Server with CNC Console IAM* section in Oracle Communications Cloud Native Core Console User Guide.

Sample Service account section in *custom-cncc_values_<version>.yaml*:

```
## Service account yaml file for cncc-sa
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cncc-sa
  namespace: cncc
  annotations: {}
```

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

M-CNCC Core Predeployment Configuration

Following are the predeployment configuration procedures:

Configuring MySQL in M-CNCC Core

This section explains how to create M-CNCC Core database (*mcncccommonconfig*) and grant permissions to the M-CNCC Core database user for relevant operations on the *mcncccommonconfig* Database.

Note:

CNCC Common Config DB Naming Convention

As the CNCC instances cannot share the same database, operator has to provide a unique name to the M-CNCC Common Config DB in the cnDBTier either limited to a site or spanning across sites.

Format for database name for M-CNCC Core:

```
<database-name>_<site-name>_<cluster>
```

Example database name:

```
mcncccommonconfig_site1_cluster1
```

While choosing the name of the namespace, make sure the following requirements are met:

- starts and ends with an alphanumeric character
- Contains a maximum of 63 characters
- contains only alphanumeric characters or '-'

In this installation guide, the commands use "mcncccommonconfig" as sample db name. The sample db name "mcncccommonconfig" must be replaced to the name chosen as per naming conventions defined by this note.

1. Log in to the server or machine which has permission to access the SQL nodes of the NDB cluster.
2. Connect to the SQL node of the NDB cluster or connect to the cnDBTier. Run the following command to connect to the cnDBTier:

```
$ kubectl -n <cndbtier_namespace> exec -it <cndbtier_sql_pod_name> -
c <cndbtier_sql_container_name> -- bash
```

Example:

```
$  kubectl -n occne-cndbtier  exec -it ndbmysqld-0 -c
mysqlndbcluster -- bash
```

3. Log in to the MySQL prompt using root permission or as a user with permission to create new users (with permissions).

```
$ mysql -h 127.0.0.1 -u root -p
```

> **Note:**
>
> After running the command mentioned above, user must enter MySQL password.

4. Run the following command to check if Database *mcncccommonconfig* exists:

```
$ show databases;
```

5. Run the following command to create a mcncccommonconfig, if the mcncccommonconfig does not exist:

```
$ CREATE DATABASE IF NOT EXISTS <M-CNCC Common Config Database>;
```

6. Run the following command to grant permission to cncc-user:

```
$ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE
TEMPORARY TABLES, DELETE, UPDATE,
EXECUTE ON <M-CNCC Core Common Config Database>.* TO'<CNCC User
Name>'@'%';
```

Example to demonstrate mcncccommonconfig creation, and granting permissions to cncc user:

```
# Command to check if database exists:
$ show databases;
# Database creation for CNCC mcncccommonconfig if do not exists
$ CREATE DATABASE IF NOT EXISTS mcncccommonconfig;
# Granting permission to user:
$ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE TEMPORARY
TABLES, DELETE, UPDATE, EXECUTE ON mcncccommonconfig .* TO'cnccusr'@'%';
```

Configuring Secret to Enable HTTPS in M-CNCC Core

This section describes how to create secret configuration for enabling HTTPS. This section must be run before enabling HTTPS in CNCC Core Ingress Gateway.

> **Note:**
>
> The passwords for TrustStore and KeyStore are stored in respective password files.

To create kubernetes secret for HTTPS, following files are required:

- ECDSA private key and CA signed certificate of CNCC (if initialAlgorithm is ES256)
- RSA private key and CA signed certificate of CNCC (if initialAlgorithm is RSA256)
- TrustStore password file
- KeyStore password file
- CA certificate or CA Bundle

> **Note:**
>
> For single cluster deployment, both M-CNCC Core and A-CNCC Core can use same configurations as both resides in same cluster namespace.

> **Note:**
>
> For multi cluster deployment configurations need to be created on both the clusters separately.

CA Bundle creation

When combining multiple CA certificates into a single certificate, add a delimiter after each certificate.

Delimiter: "--------"

Sample CA Bundle

```
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
--------
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
--------
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
```

> **Note:**
>
> Creation process for private keys, certificates and passwords is on discretion of user.

This section explains how to create the secrets for enabling HTTPS after required certificates and password files are generated:

1. Run the following command to create secret:

```
$ kubectl create secret generic <secret-name> --from-
file=<ssl_ecdsa_private_key.pem> --from-
```

```
file=<rsa_private_key_pkcs1.pem> --from-file=<ssl_truststore.txt> --from-
file=<ssl_keystore.txt> --from-file=<caroot.cer> --from-
file=<ssl_rsa_certificate.crt> --from-file=<ssl_ecdsa_certificate.crt> -n
<Namespace of CNCC Core Ingress Gateway secret>
```

Example:

```
kubectl create secret generic cncc-core-ingress-secret --from-
file=ssl_ecdsa_private_key.pem --from-file=rsa_private_key_pkcs1.pem --
from-file=ssl_truststore.txt --from-file=ssl_keystore.txt --from-
file=caroot.cer --from-file=ssl_rsa_certificate.crt --from-
file=ssl_ecdsa_certificate.crt -n cncc
```

On successfully running the above command, the following message will be displayed:
secret/cncc-core-ingress-secret created

> **Note:**
>
> Note down the command used during the creation of kubernetes secret, this command is used for updates in future.

2. Run the following command to verify the secret creation:

```
$ kubectl describe secret cncc-core-ingress-secret -n cncc
```

Perform the following procedure to update existing secrets to enable HTTPS:

1. Run the following command to create secret:

```
$ kubectl create secret generic <secret-name> --from-
file=<ssl_ecdsa_private_key.pem> --from-file=<rsa_private_key_pkcs1.pem>
--from-file=<ssl_truststore.txt> --from-file=<ssl_keystore.txt> --from-
file=<caroot.cer> --from-file=<ssl_rsa_certificate.crt> --from-
file=<ssl_ecdsa_certificate.crt> --dry-run -o yaml -n <Namespace of M-
CNCC Core Ingress Gateway secret> | kubectl replace -f - -n <Namespace of
CNCC Core Ingress Gateway secret>
```

Example:

```
$ kubectl create secret generic cncc-core-ingress-secret --from-
file=ssl_ecdsa_private_key.pem  --from-file=rsa_private_key_pkcs1.pem --
from-file=ssl_truststore.txt --from-file=ssl_keystore.txt --from-
file=caroot.cer --from-file=ssl_rsa_certificate.crt --from-
file=ssl_ecdsa_certificate.crt --dry-run -o yaml -n cncc | kubectl
replace -f - -n cncc
```

2. On successfully running the above command, the following message will be displayed:
secret/cncc-core-ingress-secret replaced

Dynamic Reloading of Certificates of M-CNCC Core

CNC Console supports dynamic reloading of certificates that are used to establish both TLS and mTLS connections. To enable dynamic reloading of certificates, the following flags must be enabled in the *occncc_custom_values_<version>.yaml* file.

```
mcncc-core:
  global:

        ingressGwCertReloadEnabled: & mcoreIGwCertReloadEnabled true
```

> **Note:**
>
> Here the new certificate must be created with the existing secret and certificate name.

The procedure for dynamic reloading of certificates as follows:

1. Delete the existing certificates with which existing secure connections were established.
2. Create the new certificate as per the requirement. The certificate must be created with the same name as the existing certificate.
3. The IGW pods automatically pick up the new certificates and the changes will be reflected in the browser.

> **Note:**
>
> **Naming update of Certificates and Secrets**
>
> If the name of the secret or the certificate is changed then the corresponding changes must be made in the *occncc_custom_values_<version>.yaml* file and either a reinstall or a helm upgrade must be done.

A-CNCC Core Predeployment Configuration

Following are the predeployment configuration procedures:

Configuring A-CNCC Core Database

This section explains how to create A-CNCC Core database (*acncccommonconfig*) and grant permissions to the A-CNCC Core database user for relevant operations on *acncccommonconfig* Database.

Note:

CNCC Common Config DB Naming Convention

As the CNCC instances cannot share the same database, operator has to provide a unique name to the CNCC Common Config DB in the cnDBTier either limited to a site or spanning across sites.

Format for database name:

```
<database-name>_<site-name>_<cluster>
```

Example database name for A-CNCC Core:

```
acncccommonconfig_site1_cluster1
```

While choosing the name of the namespace, make sure the following requirements are met:

- starts and ends with an alphanumeric character
- Contains a maximum of "63" characters
- contains only alphanumeric characters or '-'

In this installation guide, the commands use "acncccommonconfig" as sample db name. The sample db name "cncccommonconfig" must be replaced to the name chosen as per naming conventions defined by this note.

1. Log in to the server or machine which has permission to access the SQL nodes of the NDB cluster.
2. Connect to the SQL node of the NDB cluster or connect to the cnDBTier.
   Run the following command to connect to the cnDBTier:

   ```
   $ kubectl -n <cndbtier_namespace> exec -it <cndbtier_sql_pod_name> -c
   <cndbtier_sql_container_name> -- bash
   ```

   Example:

   ```
   $ kubectl -n occne-cndbtier  exec -it ndbmysqld-0 -c mysqlndbcluster --
   bash
   ```

3. Log in to the MySQL prompt using root permission or as a user with permissions to create users (with specific permissions).

   ```
   $  mysql -h 127.0.0.1 -u root -p
   ```

> **Note:**
>
> After running the command mentioned above, user must enter MySQL password.

4. Run the following command to check if Database *acncccommonconfig* exists:

```
$ show databases;
```

5. Run the following command to create a acncccommonconfig, if the acncccommonconfig does not exist:

```
$ CREATE DATABASE IF NOT EXISTS <A-CNCC Common Config Database>;
```

6. Run the following command to grant permission to cncc-user:

```
$ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE
TEMPORARY TABLES, DELETE, UPDATE,
EXECUTE ON <A-CNCC Common Config Database>.* TO'<CNCC User
Name>'@'%';
```

Example to demonstrate acncccommonconfig creation, and granting permissions to cncc user:

```
# Command to check if database exists:
$ show databases;
# Database creation for CNCC acncccommonconfig if do not exists
$ CREATE DATABASE IF NOT EXISTS acncccommonconfig;
# Granting permission to user:
$ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE
TEMPORARY TABLES, DELETE, UPDATE, EXECUTE ON acncccommonconfig .*
TO'cnccusr'@'%';
```

Configuring Secret to Enable HTTPS in A-CNCC Core

This section describes how to create secret configuration for enabling HTTPS. This section must be run before enabling HTTPS in A-CNCC Core Ingress Gateway.

> **Note:**
>
> The passwords for TrustStore and KeyStore are stored in respective password files.

To create kubernetes secret for HTTPS, following files are required:

- ECDSA private key and CA signed certificate of CNCC (if initialAlgorithm is ES256)
- RSA private key and CA signed certificate of CNCC (if initialAlgorithm is RSA256)
- TrustStore password file

- KeyStore password file
- CA certificate or CA Bundle

> **Note:**
>
> For single cluster deployment, both M-CNCC Core and A-CNCC can use same configurations as both reside in same cluster namespace.

> **Note:**
>
> For multi cluster deployment configurations need to be created on both the clusters separately.

CA Bundle creation

When combining multiple CA certificates into a single certificate, add a delimiter after each certificate.

Delimiter: "--------"

Sample CA Bundle

```
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
--------
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
--------
-----BEGIN CERTIFICATE-----
MIID4TCC...
...
...jtUl/zQ==
-----END CERTIFICATE-----
```

> **Note:**
>
> Creation process for private keys, certificates and passwords is on discretion of user.

This section explains how to create the secrets for enabling HTTPS after required certificates and password files are generated:

1. Run the following command to create secret:

```
$ kubectl create secret generic <secret-name> --from-
file=<ssl_ecdsa_private_key.pem> --from-
file=<rsa_private_key_pkcs1.pem> --from-file=<ssl_truststore.txt> --
from-file=<ssl_keystore.txt> --from-file=<caroot.cer> --from-
file=<ssl_rsa_certificate.crt> --from-
file=<ssl_ecdsa_certificate.crt> -n <Namespace of CNCC Core Ingress
Gateway secret>
```

Example:

```
kubectl create secret generic cncc-core-ingress-secret --from-
file=ssl_ecdsa_private_key.pem --from-
file=rsa_private_key_pkcs1.pem --from-file=ssl_truststore.txt --
from-file=ssl_keystore.txt --from-file=caroot.cer --from-
file=ssl_rsa_certificate.crt --from-file=ssl_ecdsa_certificate.crt -
n cncc
```

> **Note:**
>
> Note down the command used during the creation of Kubernetes secret, this command is used for updating the secrets in future.

On successfully running the above command, the following message will be displayed:
secret/cncc-core-ingress-secret created

2. Run the following command to verify the secret creation:
```
$ kubectl describe secret cncc-core-ingress-secret -n cncc
```

Perform the following procedure to update existing secrets to enable HTTPS:

1. Run the following command to create secret:

```
$ kubectl create secret generic <secret-name> --from-
file=<ssl_ecdsa_private_key.pem> --from-
file=<rsa_private_key_pkcs1.pem> --from-file=<ssl_truststore.txt> --
from-file=<ssl_keystore.txt> --from-file=<caroot.cer> --from-
file=<ssl_rsa_certificate.crt> --from-
file=<ssl_ecdsa_certificate.crt> --dry-run -o yaml -n <Namespace of
CNCC Core Ingress Gateway secret> | kubectl replace -f - -n
<Namespace of CNCC Core Ingress Gateway secret>
```

Example:

```
$ kubectl create secret generic cncc-core-ingress-secret --from-
file=ssl_ecdsa_private_key.pem  --from-
file=rsa_private_key_pkcs1.pem --from-file=ssl_truststore.txt --
from-file=ssl_keystore.txt --from-file=caroot.cer --from-
```

```
fileS=ssl_rsa_certificate.crt --from-file=ssl_ecdsa_certificate.crt --dry-
run -o yaml -n cncc | kubectl replace -f - -n cncc
```

2. On successfully running the above command, the following message will be displayed: *secret/cncc-core-ingress-secret replaced*

Dynamic Reloading of Certificates of A-CNCC Core

CNC Console supports dynamic reloading of certificates that are used to establish both TLS and mTLS connections. To enable dynamic reloading of certificates, the following flags must be enabled in the *occncc_custom_values_<version>.yaml* file.

```
acncc-core:
  global:
     ingressGwCertReloadEnabled: &acoreIGwCertReloadEnabled true
```

> **Note:**
>
> Here the new certificate must be created with the existing secret and certificate name.

The procedure for dynamic reloading of certificates as follows:

1. Delete the existing certificates with which existing secure connections were established.
2. Create the new certificate as per the requirement. The certificate must be created with the same name as the existing certificate.
3. The IGW pods automatically pick up the new certificates and the changes will be reflected in the browser.

> **Note:**
>
> **Naming update of Certificates and Secrets**
>
> If the name of the secret or the certificate is changed then the corresponding changes must be made in the *occncc_custom_values_<version>.yaml* file and either a reinstall or a helm upgrade must be done.

Configuring A-CNCC Core mTLS

This section describes the A-CNCC Core Configuration for enabling mTLS.

mTLS Configuration at A-CNCC Core

mTLS must be enabled at the ingress-gateway SSL configuration section of the A-CNCC Core *occncc_custom_values_<version>.yaml* file. The parameter **scheme** must be set to https in the *occncc_custom_values_<version>.yaml* file.

Sample TLS Configuration section of A-CNCC Core:

```
acncc-core:
  global
    # CNCC https enabled
    httpsEnabled: &httpsEnabled true
    # Server Configuration for http and https support
    enableIncomingHttp: &enableIncomingHttp false
    enableIncomingHttps: &enableIncomingHttps true
    # Enables server with MTLS
    needClientAuth: &needClientAuth true
```

> **Note:**
>
> While enabling mTLS, **needClientAuth** must be set to true in A-CNCC Core.

Deploying CNC Console

This section provides installation procedures to install CNC Console using Command Line Interface (CLI). To install CNC Console using CDCS, see *Oracle Communications Cloud Native Core CD Control Server User Guide.*

Perform the following procedure to deploy CNC Console:

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

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

   Example:

   ```
    helm search repo cncc -l


   NAME              CHART VERSION  APP VERSION  DESCRIPTION
   ocspf-helm-repo/cncc      22.3.2         22.3.2     A Helm chart
   for CNC Console
   ```

2. Prepare the `occncc_custom_values_<version>.yaml`file with the required parameter information. For configuring custom-cncc_values.yaml file, see CNC Console Configuration Parameters section.

Note:

The annotation *metallb.universe.tf/address-pool: signaling/oam* is required in global section if MetalLB in CNE 1.8.x onwards is used.

```
customExtension:
  lbServices:
    labels: {}
    annotations:
     # The annotation metallb.universe.tf/address-pool:
signaling/oam is required if MetalLB in CNE 1.8.x is used
      metallb.universe.tf/address-pool: oam
      service.beta.kubernetes.io/oci-load-balancer-internal:
"true"
```

Note:

The annotation `oracle.com.cnc/app-protocols: '{"http2-tls":"TCP"}'` is required in global section of *custom-cncc-iam_values.yaml* file when CNC Console is deployed with HTTPs enabled in CNE.

```
customExtension:
  lbServices:
    labels: {}
    annotations:
      oracle.com.cnc/app-protocols: '{"http2-tls":"TCP"}'
```

Following configurations are needed in *occncc_custom_values_<version>.yaml* file. This includes configuring the following based on the deployment:

a. Update unique CNCC ID per cluster (global.self.cnccId)
b. Provide the details of M-CNCC IAMs (mCnccIams)
c. Provide the details of A-CNCC (aCnccs)
d. Provide the details of Agent Instances (instances)
e. In case of M-CNCC Core, additionally provide the details of M-CNCC Cores and M-CNCC Instances (instances)

Note:

- For instances configuration details see, CNC Console Instances Configurations section and section CNC Console Instances Configuration Options section.
- There are multiple M-CNCC, A-CNCC, NF Instances, and OCCNE Instances. So, be cautious while updating *occncc_custom_values_<version>.yaml* file.
- In case of M-CNCC Core*, cmservice.envSystemName* in *occncc_custom_values_<version>.yaml* file can be used to display cluster name.
  Example: envSystemName: CNCC - Site Name
- Routes creation happens automatically, there is no need to provide routes in the ingress-gateway section; only instances details have to be provided in global section.

  A-CNCC and M-CNCC Core uses same cncc helm chart, only deployment configurations may differ.
- See CNC Console Deployment Configuration Workflow section for details on deployment specific configuration updates CNC Console Deployment Configuration Workflow.

3. The following are the Sample configuration section for CNCC:
   a. **Single Cluster Deployment**
   Sample configuration section for CNCC in case of single cluster deployment

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.bumblebee
      port: 80
```

```
instances:
  - id: Cluster1-grafana
    type: CS
    owner: Cluster1
    fqdn: occne-kube-prom-stack-grafana.occne-infra.svc.bumblebee
    apiPrefix: /bumblebee/grafana
  - id: Cluster1-kibana
    type: CS
    owner: Cluster1
    fqdn: occne-kibana.occne-infra.svc.bumblebee
    apiPrefix: /bumblebee/kibana
  - id: Cluster1-scp-instance1
    type: SCP
    owner: Cluster1
    fqdn: ocscp-scpc-configuration.scp.svc.bumblebee
    port: 80
```

b. Multi Cluster Deployment

Sample configuration section for manager only deployment set cncc-iam, mcncc-core to "true" and acncc-core to "false".

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: false

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id: Cluster1-grafana
      type: CS
      owner: Cluster1
      fqdn: occne-kube-prom-stack-grafana.occne-infra.svc.bumblebee
      apiPrefix: /bumblebee/grafana
    - id: Cluster1-kibana
      type: CS
      owner: Cluster1
      fqdn: occne-kibana.occne-infra.svc.bumblebee
      apiPrefix: /bumblebee/kibana
```

```
        - id: Cluster2-kibana
          type: CS
          owner: Cluster2
          fqdn: occne-kibana.occne-infra.svc.jazz
          apiPrefix: /jazz/kibana
        - id: Cluster2-scp-instance2
          type: SCP
          owner: Cluster2
          fqdn: ocscp-scpc-configuration.scp.svc.jazz
          port: 80
```

Sample configuration section for manager managing local NFs. User must configure cncc-iam, mcncc-core and acncc-core flag to "true".

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.bumblebee
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id: Cluster1-grafana
      type: CS
      owner: Cluster1
      fqdn: occne-kube-prom-stack-grafana.occne-
infra.svc.bumblebee
      apiPrefix: /bumblebee/grafana
    - id: Cluster1-kibana
      type: CS
      owner: Cluster1
      fqdn: occne-kibana.occne-infra.svc.bumblebee
      apiPrefix: /bumblebee/kibana
    - id: Cluster1-scp-instance1
      type: SCP
```

```
      owner: Cluster1
      fqdn: ocscp-scpc-configuration.scp.svc.bumblebee
      port: 80
    - id: Cluster2-scp-instance2
      type: SCP
      owner: Cluster2
      fqdn: ocscp-scpc-configuration.scp.svc.jazz
      port: 80
```

Sample configuration section for agent only deployment. User must configure cncc-iam and mcncc-core to "false" and acncc-core to "true".

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id: Cluster2-kibana
      type: CS
      owner: Cluster2
      fqdn: occne-kibana.occne-infra.svc.jazz
      apiPrefix: /jazz/kibana
    - id: Cluster2-scp-instance2
      type: SCP
      owner: Cluster2
      fqdn: ocscp-scpc-configuration.scp.svc.jazz
      port: 80
```

Note:

- In the above examples, mCnccIams port is assumed as "80". The mCnccIams port configuration must be added only if port value is other than "80".
- Instance id must be globally unique as it will be used for routing, here is the recommendation for id naming
- `id: <owner>-<instance name>`
  Example:

  ```
  id: Cluster2-scp-instance2
  ```
- `aCnccs.id`:
  - is mandatory as its needed for site authorization
  - `aCnccs.id` value must be same as `self.cnccId`
  - `aCnccs.role` and `mCnccCores.role` are optional, which can be used for overriding site role name.
- For HTTPS enabled deployment scheme and port has to be updated to "https" and "https port value" in `mCnccIams` and `aCnccs`. For sample configuration, see CNCC Core InstancesConfiguration examples listed in appendix.

4. Deploy M-CNCC IAM using repository and helm tar.

Caution:

The CNCC helm install command may take longer than usual time to run because the kubernetes job runs through install helm hook. Helm deployment status is shown as **DONE** after the applicable hook is run.

Caution:

Pod restarts maybe observed at M-CNCC Core ingress-gateway during fresh installation, upgrade, or rollback. This is because M-CNCC Core ingress-gateway internally check if CNCC IAM KC pod is up or not via CNCC IAM ingress-gateway. Once CNCC IAM KC pod is up then you should see M-CNCC Core ingress-gateway in running state.

To verify the deployment status, open a new terminal and run the following command:

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

Example:

```
kubectl get pods -n cncc -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.

> **Note:**
>
> If helm purge do not clean the deployment and Kubernetes objects completely, then follow CNC Console IAM Clean up section.

Update DB details in *occncc_custom_values_<version>.yaml* file.

```
dbVendor: mysql
dbName: cnccdb
dbHost: mysql-sds.default.svc.cluster.local
dbPort: 3306
```

> **Note:**
>
> Database must be created first and that Database name must be mentioned as dbName.

a. Run the following command for installing using helm repository:

```
helm install <release_name> <helm-repo> -f
occncc_custom_values_22.3.2.yaml --namespace <namespace_name> --
version <helm_version>
```

Where:

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

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

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

Example:

```
helm install cncc ocscp-helm-repo/cncc -f
occncc_custom_values_22.3.2.yaml --namespace cncc --version 22.3.2
```

b. Run the following command for Installing using helm tar:

```
helm install cncc -f occncc_custom_values_22.3.2.yaml --namespace
<namespace> <chartpath>./<chart>.tgz
```

Example:

```
helm install cncc -f occncc_custom_values_22.3.2.yaml --namespace
cncc occncc-22.3.2.tgz
```

5. Run the following commands to upgrade the CNCC Configuration:

> **Note:**
>
> For details about CNC Console Deployment Configuration workflow, see CNC Console Deployment Configuration Workflow section.

You have to do the following steps for upgrade:

- Prepare the *occncc_custom_values_<version>.yaml* file for upgrade
- Upgrade CNCC

a. Run the following command for upgrading using helm repository:

```
$ helm3 upgrade <release_name> <helm_chart> -f
<occncc_custom_values_<version>.yaml> --namespace <namespace-
name>
```

Example:

```
$ helm3 upgrade cncc ocspf-helm-repo/cncc -f
occncc_custom_values_22.3.2.yaml --namespace cncc
```

b. Run the following command for upgrading using helm tar:

```
helm  upgrade cncc -f occncc_custom_values_<version>.yaml --
namespace <namespace> <chartpath>./<chart>.tgz
```

Example:

```
helm  upgrade cncc -f occncc_custom_values_22.3.2.yaml --
namespace cncc occncc-22.3.2s.tgz
```

6. Run the following command to check the deployment status:

```
helm status <release_name>
```

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

   ```
   kubectl -n <namespace_name> get services
   ```

   Example:

   ```
   $ kubectl -n cncc get services
   ```

   Example:

   ```
   NAME                          TYPE           CLUSTER-IP      EXTERNAL-
   IP      PORT(S)                       AGE
   cncc-acore-cmservice          ClusterIP      10.233.41.221
   <none>           8442/TCP                     24h
   cncc-acore-igw-cache          ClusterIP      None
   <none>           8000/TCP                     24h
   cncc-acore-ingress-gateway    ClusterIP      10.233.47.246
   <none>           80/TCP                       24h
   cncc-iam-igw-cache            ClusterIP      None
   <none>           8000/TCP                     24h
   cncc-iam-ingress-gateway      LoadBalancer   10.233.3.123
   10.75.224.188    80:30185/TCP                 24h
   cncc-iam-kc-headless          ClusterIP      None
   <none>           8285/TCP                     24h
   cncc-iam-kc-http              ClusterIP      10.233.42.16
   <none>           8285/TCP,8443/TCP,9990/TCP   24h
   cncc-mcore-cmservice          ClusterIP      10.233.9.144
   <none>           8442/TCP                     24h
   cncc-mcore-igw-cache          ClusterIP      None
   <none>           8000/TCP                     24h
   cncc-mcore-ingress-gateway    LoadBalancer   10.233.44.167
   10.75.224.189    80:30175/TCP                 24h
   ```

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

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

   Example:

   ```
   $ kubectl -n cncc get pods
   ```

   Example:

   ```
   NAME                                          READY   STATUS
   RESTARTS        AGE
   cncc-acore-cmservice-8699559488-jxwwt         1/1     Running
   0               24h
   cncc-acore-ingress-gateway-c685bf678-bgmdf    1/1     Running
   0               24h
   cncc-iam-ingress-gateway-6776df55cd-xzx2m     1/1     Running
   0               24h
   ```

```
cncc-iam-kc-0                                  2/2     Running
0              24h
cncc-mcore-cmservice-587749d58d-8lnd7          1/1     Running
0              24h
cncc-mcore-ingress-gateway-59758876b-zc7d7     1/1     Running
0              24h
```

Caution:

Do not exit from helm install command manually. After running the helm install command, it will take a while to install all the services. This is because during installation Kubernetes jobs are initiated by helm hooks. For verifying the installation status, see Verifying CNC Console IAM Installation.

Note:

timeout duration (optional): If not specified, the default value will be 5m (5 minutes) in Helm3. Specifies the time to wait for any individual Kubernetes operation (like Jobs for hooks). The default value is 5m0s. If the helm install command fails at any point to create a Kubernetes object, it will internally call the purge to delete after timeout value (default: 300s). Here timeout value is not for overall install, but it is for automatic purge on installation failure.

Verifying CNC Console Installation

This section describes how to verify if CNC Console is installed successfully.

1. To check the installation status, run the following command:

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

   For example:

   ```
   helm ls -n cncc
   ```

   You should see the status as **DEPLOYED** if the deployment is successful.

2. To get the status of jobs and pods, run the following command:

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

   For example:

   ```
   kubectl get pod -n cncc
   ```

   You should see the status as **Running** and **Ready** for all the pods if the deployment is successful.

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

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

For example:

```
kubectl get services -n cncc
```

Note:

Take a backup of the following files that are required during disaster recovery:

- Updated *occncc_custom_values_<version>.yaml* file
- Updated helm charts
- secrets, certificates, and keys that are used during installation

Note:

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 *Oracle Communications Cloud Native Core Console Troubleshooting Guide.*

CNC Console Microservices

This section provides the details of CNC Console microservices.

M-CNCC IAM Microservices

M-CNCC IAM has the microservices which are responsible for Identity Access Management:

The following is an example of services CNC Console IAM offers:

Table 3-2 CNC Console IAM Microservices

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
cncc-iam-kc-headless	ClusterIP	None	<none>	8285/TCP	9m13s
cncc-iam-kc-http	ClusterIP	10.233.25.75	<none>	8285/TCP	9m13s
cncc-iam-ingress-gateway	LoadBalancer	10.233.7.236	10.75.182.72	8080:30346/TCP	9m13s
cncc-iam-cncc-iam-igw-cache	ClusterIP	None	<none>	8000/TCP	9m13s

> **Note:**
>
> `cncc-iam-cncc-iam-igw-cache` is an ingress gateway coherence service which is enabled by default by ingress-gateway chart. CNC Console IAM does not use this feature.

M-CNCC Core Microservices

M-CNCC Core has two microservices:

1. **cncc-mcore-ingress-gateway**: cncc-mcore-ingress-gateway is responsible to redirect the request to either producer NF or CNCC Core GUI.
2. **cncc-mcore_cmservice**: cncc-mcore_cmservice is responsible for displaying CNCC Core GUI.

Following is an example of services M-CNCC Core offers:

Table 3-3 M-CNCC Core Microservices

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
cncc-mcore-cmservice	ClusterIP	10.108.119.182	<none>	8442/TCP	24h
cncc-mcore-igw-cache	ClusterIP	None	<none>	8000/TCP	24h
cncc-mcore-ingress-gateway	LoadBalancer	10.103.6.17	<pending>	8080:31417/TCP	24h

A-CNCC Core Microservices

A-CNCC Core has the following microservices:

1. **cncc-acore-ingress-gateway**: cncc-acore-ingress-gateway is responsible to redirect the request to either producer NF or CNCC Core GUI.
2. **cncc-acore_cmservice**: cncc-acore_cmservice is responsible for displaying CNCC Core GUI.

Following is an example of services A-CNCC Core offers:

Table 3-4 A-CNCC Core Microservices

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
cncc-acore-cmservice	ClusterIP	10.106.93.165	<none>	8442/TCP	24h
cncc-acore-cncc-core-igw-cache	ClusterIP	None	<none>	8000/TCP	24h
cncc-acore-ingress-gateway	ClusterIP	10.105.11.175	<none>	8080/TCP	24h

> **Note:**
>
> `cncc-acore-igw-cache` is an ingress gateway coherence service which is enabled by default by ingress-gateway helm chart, CNC Console Core does not use this feature.

4
Customizing CNC Console

This section describes about the CNC Console customization.

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

To customize the custom yaml files, perform the following steps:

1. Unzip `Custom_Templates` file available in the extracted documentation release package. For more information on how to download the package from MOS, see Downloading CNC Console package section.
   The following files are used to customize the deployment parameters during installation:
   - M-CNCC IAM custom values file : `occncc_custom_values_<version>.yaml`
   - CNCC IAM Schema file for rollback to previous version : `occncc_rollback_iam_schema_<version>.sql`
   - CNCC Metric Dashboard file : `occncc_metric_dashboard_<version>.json`
   - CNCC Metric Dashboard file for CNE supporting Prometheus HA (CNE 1.9.x onwards): `occncc_metric_dashboard_promha_<version>.json`
   - CNCC Alert Rules file: `occncc_alertrules_<version>.yaml`
   - CNCC Alert Rules file for CNE supporting Prometheus HA (CNE 1.9.x onwards): `occncc_metric_dashboard_promha_<version>.json`
   - CNCC MIB files: `occncc_mib_<version>.mib, occncc_mib_tc_<version>.mib`
2. Customize the appropriate custom value yaml 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.

Global Configuration Options

This section includes information about the configuration parameters of the Global Configuration section of CNC Console Core:

Table 4-1 Global Configuration Options

Parameter	Description	Details
`global.dockerRegistry`	This is a mandatory parameter. Here user provides the registry that contains cncc-core images. It comprises of the following: <registry-url>:<registry-port> Example: ocspf-registry.us.oracle.com:5000	**DataType:**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:** ocspf-registry.us.oracle.com:5000
`global.clusterDomain`	This is a mandatory parameter. Cluster Domain where cncc-core will be deployed example: cluster.local Note: To check cluster domain use the following command: `kubectl -n kube-system get configmap kubeadm-config -o yaml | grep -i dnsDomain`	**DataType:**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.
`global.serviceAccountName`	This is an optional parameter. Name of service account. If this field is kept empty then a default service account 'cncc-core-service-account' is created. If any value is provided then a service account has to be created manually. `kubectl create serviceaccount <name> -n <namespace>`	**DataType:**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 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.customExtension.allResources.labels`	This is an optional parameter. This can be used to add custom label(s) to all k8s resources that will be created by Ingress Gateway helm chart.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway kubernetes resources
`global.customExtension.allResources.annotations`	This is an optional parameter. This can be used to add custom annotation(s) to all k8s resources that will be created by Ingress Gateway helm chart.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway kubernetes resources
`global.customExtension.nonlbStatefulsets.labels`	This is an optional parameter. This can be used to add custom label(s) to all Statefulsets that will be created by Ingress-Gateway helm chart which are associated to a Service which if of Load Balancer Type.	**DataType:**String **Range:**Custom Annotations that needs to be added to Ingress Gateway Deployments that are associated to a Service which is of Load Balancer type
`global.customExtension.nonlbStatefulsets.annotations`	This is an optional parameter. This can be used to add custom annotation(s) to all Statefulsets that will be created by Ingress-Gateway helm chart which are associated to a Service which if of Load Balancer Type.	**DataType:**String **Range:**Custom Annotations that needs to be added to Ingress Gateway Deployments that are associated to a Service which is of Load Balancer type
`global.customExtension.lbServices.labels`	This is an optional parameter. This can be used to add custom label(s) to all Load Balancer Type Services that will be created by Ingress Gateway helm chart.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway kubernetes resources
`global.customExtension.lbServices.annotations`	This is an optional parameter. This can be used to add custom annotation(s) to all Load Balancer Type Services that will be created by Ingress Gateway helm chart.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway kubernetes resources

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.customExtension.lbDeployments.labels`	This is an optional parameter. This can be used to add custom label(s) to all Deployments that will be created by Ingress Gateway helm chart which are associated to a Service which if of Load Balancer Type.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway kubernetes resources
`global.customExtension.lbDeployments.annotations`	This is an optional parameter. This can be used to add custom annotation(s) to all Deployments that will be created by Ingress Gateway helm chart which are associated to a Service which if of Load Balancer Type.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway 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 Ingress Gateway helm chart.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway kubernetes resources
`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 Ingress Gateway helm chart.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway kubernetes resources
`global.customExtension.nonlbDeployments.labels`	This can be used to add custom label(s) to all Deployments that will be created by Ingress Gateway helm chart which are associated to a Service which if not of Load Balancer Type.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway kubernetes resources

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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 Ingress Gateway helm chart which are associated to a Service which if not of Load Balancer Type.	**DataType:**String **Range:**Custom Labels that needs to be added to all the Ingress Gateway kubernetes resources
`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.	**DataType:**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.nfName`	This is a mandatory parameter. Name of deployment for which helm test is done	**DataType:**String **Range:**NF Name
`global.test.image.name`	This is a mandatory parameter. Image name for the helm test container image	**DataType:**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:**
`global.test.image.tag`	This is a mandatory parameter. Image version tag for helm test	**DataType:**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.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent
`global.test.config.logLevel`	This is a mandatory parameter. Log level for helm test pod	**DataType:**String **Range:**WARN, DEBUG, INFO, etc.

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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. Unit:seconds	**DataType:**String **Range:** 1-300
`global.test.resources`	This is a mandatory parameter. which ever kubernetes resource are mentioned, will be logged in helm test.	**DataType:**<List[String]> **Range:** It takes resources and its version in the form of <resource_name>/<max_version_supportedbyNF> - horizontalpodautoscalers/v1 - deployments/v1 - configmaps/v1 - prometheusrules/v1 - serviceaccounts/v1 - poddisruptionbudgets/v1 - roles/v1 - statefulsets/v1 - persistentvolumeclaims/v1 - services/v1 - rolebindings/v1
`global.test.complianceEnable`	This is a mandatory parameter. It will enable or disable helm test resource logging	**DataType:**Boolean **Range:**True or False **Default Value:** True
`global.validationHook.image.name`	This is a mandatory parameter. Image name for validation hook	**DataType:**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.validationHook.image.tag`	This is a mandatory parameter. Image version tag for validation hook	**DataType:**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.validationHook.image.imagePullPolicy`	This is an optional parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.validationHook.config.logLevel.root`	This is a mandatory parameter. Log level for validation hook pod	**DataType:**String **Range:**WARN, DEBUG, INFO, etc.
`global.ephemeralStorage.requests.containersLogStorage`	This is a mandatory parameter. Set value for Ephemeral Storage Requests	**DataType:**Integer **Range:** It can take values in integer and that values are used in MBs.
`global.ephemeralStorage.requests.containersCriticalStorage`	This is a mandatory parameter. Set value for Ephemeral Storage Requests	**DataType:**Integer **Range:** It can take values in integer and that values are used in MBs. **Default Value:**
`global.ephemeralStorage.limits.containersLogStorage`	This is a mandatory parameter. Set value for Ephemeral Storage Limits	**DataType:**Integer **Range:** It can take values in integer and that values are used in MBs.
`global.ephemeralStorage.limits.containersCriticalStorage`	Set value for Ephemeral Storage Limits	**DataType:**Integer **Range:** It can take values in integer and that values are used in MBs.
`global.hookJobResources.limits.cpu`	This is a mandatory parameter. It limits the number of CPUs to be used by the helm test pod.	**DataType:**String **Range:**Valid floating point value between 0 and 1
`global.hookJobResources.limits.memory`	This is a mandatory parameter. It limits the number of memory to be used by the helm test pod.	**DataType:**Integer **Range:** Valid Integer value followed by Mi/Gi etc.
`global.hookJobResources.limits.logStorage`	This is a mandatory parameter. It limits the logStorage (ephemeral storage) to be used by the helm test pod.	**DataType:**Integer **Range:** Values will be set by global.ephemeralStorage.limits.containersLogStorage **Default Value:**
`global.hookJobResources.limits.criticalStorage`	This is a mandatory parameter. It limits the criticalStorage (ephemeral storage) to be used by the helm test pod.	**DataType:**Integer **Range:** Values will be set by global.ephemeralStorage.limits.containersCrititcalStorage
`global.hookJobResources.requests.cpu`	This is a mandatory parameter. The minimum amount of CPUs required	**DataType:**String **Range:**Valid floating point value between 0 and 1
`global.hookJobResources.requests.memory`	This is a mandatory parameter. The minimum amount of CPUs required	**DataType:**Integer **Range:** Valid Integer value followed by Mi/Gi etc.

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.hookJobResources.requests.logStorage`	This is a mandatory parameter. The minimum amount of logStorage (ephemeral storage)	**DataType:**Integer **Range:** Values will be set by global.ephemeralStorage.requests.containerLogStorage **Default Value:**
`global.hookJobResources.requests.criticalStorage`	This is a mandatory parameter. The minimum amount of criticalStorage (ephemeral storage)	**DataType:**Integer **Range:** Values will be set by global.ephemeralStorage.requests.containerCrititcalStorage **Default Value:**
`global.k8sResource.container.prefix`	This is an optional parameter. This value will be used to prefix to all the container names of Ingress-Gateway	**DataType:**Integer **Range:** Value that will be prefixed to all the container names of Ingress-Gateway. **Default Value:**
`global.k8sResource.container.suffix`	This is an optional parameter. This value will be used to suffix to all the container names of Ingress-Gateway.	**DataType:**Integer **Range:** Value that will be prefixed to all the container names of Ingress-Gateway. **Default Value:**
`global.extraContainers`	This is a mandatory parameter. To enable or disable the debug tools container	**DataType:**enum **Range:** DISABLED, ENABLED **Default Value:**

`global.extraContainersTpl.command`	This is a mandatory parameter.
	String array used for container command.
		DataType: List[String]
		Range:
		Example:
		`drop:`
		`-/bin/sleep`
		`-infinity`

`global.extraContainersTpl.image`	This is a mandatory parameter. Docker image name	**DataType:**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.extraContainersTpl.imagePullPolicy`	This is a mandatory parameter. Image Pull Policy	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.extraContainersTpl.name`	This is a mandatory parameter. Name of the container	**DataType:**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.extraContainersTpl.resources.limits.ephemeral-storage`	This is a mandatory parameter. The maximum amount of Ephemeral Storage allowed	**DataType:**Integer **Range:** Valid Integer value followed by Mi/Gi etc. **Default Value:**4Gi
`global.extraContainersTpl.resources.limits.cpu`	The maximum amount of cpu allowed	**DataType:**String **Range:**Valid floating point value between 0 and 1
`global.extraContainersTpl.resources.limits.memory`	This is a mandatory parameter. The maximum amount of memory allowed	**DataType:**Integer **Range:** Valid Integer value followed by Mi/Gi etc. **Default Value:**2Gi
`global.extraContainersTpl.resources.requests.ephemeral-storage`	This is a mandatory parameter. The minimum amount of Ephemeral Storage required	**DataType:**Integer **Range:** Valid Integer value followed by Mi/Gi etc. **Default Value:**2Gi
`global.extraContainersTpl.resources.requests.cpu`	This is a mandatory parameter. The minimum amount of cpu limits required	**DataType:**String **Range:**Valid floating point value between 0 and 1 **Default Value:**
`global.extraContainersTpl.resources.requests.memory`	This is a mandatory parameter. The minimum amount of memory required	**DataType:**Integer **Range:** Valid Integer value followed by Mi/Gi etc. **Default Value:**1Gi
`global.extraContainersTpl.securityContext.allowPrivilegeEscalation`	This is a mandatory parameter. AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This bool directly controls if the no_new_privs flag will be set on the container process	**DataType:**Boolean **Range:**True or False **Default Value:** True

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details

`global.extraContainersTpl.securityContext.capabilities.drop`	This is a mandatory parameter.
	Removed capabilities
		DataType: List[String]
		Range:
		Example:
		`drop:`
		`-ALL`
		Default Value:

`global.extraContainersTpl.securityContext.capabilities.add`	This is a mandatory parameter.
	Added capabilities
		DataType: List[String]
		Range:
		Example:
		`add:`
		`- NET_RAW`
		`- NET_ADMIN`
		Default Value:

`global.extraContainersTpl.securityContext.readOnlyRootFilesystem`	This is a mandatory parameter. Whether this container has a read-only root filesystem. Default is false.	**DataType:**Boolean **Range:**True or False **Default Value:** False
`global.serviceMeshCheck`	This is an optional parameter. This flag needs to set it "true" if Service Mesh would be present where CNCC will be deployed	**DataType:**Boolean **Range:**True or False **Default Value:** False
`global.serviceMeshHttpsEnabled`	This is an optional parameter. If Service Mesh is deployed with TLS/MTLS disabled then set this flag to false	**DataType:**Boolean **Range:**True or False **Default Value:** True
`global.istioSidecarQuitUrl`	This is an optional parameter. This needs to be set with correct url format http://127.0.0.1:<Istio sidecar proxy admin port>/quitquitquit" if Service Mesh would be present where CNCC will be deloyed	**DataType:**String **Range:**Valid url

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.istioSidecarReadyUrl`	This is an optional parameter. This needs to be set with correct url format http://127.0.0.1:<Istio sidecar proxy admin port>/ready" if Service Mesh would be present where CNCC will be deloyed	**DataType:**String **Range:**Valid url **Default Value:**
`global.dbHost`	This is a mandatory parameter. It the hostname for persistence db Example: mysql.default.svc.cluster.local	**DataType:**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:**
`global.dbPort`	This is a mandatory parameter. It is the db port for cncc-core Example: 3306	**DataType:**Integer **Range:**0-65535. **Default Value:**
`global.secretName`	It specifies an existing secret to be used for mysql username and password. Example: secretName: &mySqlSecretNameRef cncc-db-secret	**DataType:**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:**
`global.cmServiceHttpPort`	This is a mandatory parameter. It specifies the port number which makes cmservice visible to other services running within the same K8s cluster and the also used by common config client for db creation.	**DataType:**Integer **Range:**0-65535. **Default Value:**
`global.isMultiClusterDeployment`	This is an optional parameter. This flag indicates single cluster deployment or multi cluster deployment. By default single cluster deployment is configured, flag must be set to true for multi cluster deployment.	**DataType:**Boolean **Range:**True or False **Default Value:** False

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.cncc-iam.enabled`	This is a mandatory parameter. This flag indicates manager IAM installation. By default it is set to true for single cluster deployment	**DataType:**Boolean **Range:**True or False **Default Value:**
`global.mcncc-core.enabled`	This is a mandatory parameter. This flag indicates manger core installation. By default it is set to true for single cluster deployment Note: Set this flag to false while installing agent core in case of multi cluster deployment	**DataType:**Boolean **Range:**True or False **Default Value:** True
`global.acncc-core.enabled`	This is a mandatory parameter. This flag indicates agent core installation. By default it is set to true for single cluster deployment Note: Set this flag to false while installing manager core in case of multi cluster deployment	**DataType:**Boolean **Range:**True or False **Default Value:** True
`global.self.cnccId`	This is a mandatory parameter. It is the ID of deployment in Multi Cluster Multi Instance Cluster	**DataType:**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:**

`global.mCnccIams[]`	This is a mandatory parameter.
	It is the list of Manager CNCC-IAMsFor further Information Refer CNCConsole 22.2.0 Multi Cluster and Multi Instance Configurations
		DataType: List[String]
		Range:
		Example:
		`drop:`
		`-all`
		Default Value:

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.mCnccCores[]`	This is a mandatory parameter. It is the list of Manager CNCC-Cores. For more Information see, CNCConsole Multi Cluster and Multi Instance Configurations. Note: This parameter is not applicable for A-CNCC-Core deployment	**DataType:**<List> **Range:** **Default Value:**
`global.aCnccs[]`	This is a mandatory parameter. For more Information see, CNCConsole Multi Cluster and Multi Instance Configurations.	**DataType:**<List> **Range:** Example: **Default Value:**
`global.instances[]`	This is a mandatory parameter. It is the list of NF Instances and Common Services added on various Agents and Managers. For more Information see, CNCConsole Multi Cluster and Multi Instance Configurations.	**DataType:**Integer **Range:**0-65535. **Default Value:** 60
`global.publicHttpSignalingPort`	This is an optional parameter. It is the port on which Ingress Gateway service is exposed # If httpsEnabled is false, this Port would be HTTP/2.0 Port (unsecured) publicHttpSignalingPort: 80	**DataType:**Integer **Range:**0-65535. **Default Value:**
`global.publicHttpsSignallingPort`	This is an optional parameter. It is the port on which Ingress Gateway service is exposed # If httpsEnabled is true, this Port would be HTTPS/2.0 Port (secured SSL)	**DataType:**Integer **Range:**0-65535. **Default Value:**

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.metalLbIpAllocationEnabled`	This is an optional parameter. This field enables or disables IP Address allocation from Metallb Pool	**DataType:**Boolean **Range:**True or False **Default Value:** True
`global.metalLbIpAllocationAnnotation`	It is the address Pool Annotation for Metallb	**DataType:**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:** metallb.universe.tf/ address-pool: signaling"
`global.staticIpAddressEnabled`	This is an optional parameter. If Static load balancer IP needs to be set, then set staticIpAddressEnabled flag to true and provide value for staticIpAddress else random IP will be assigned by the metalLB from its IP Pool	**DataType:**Boolean **Range:**True or False **Default Value:** False
`global.staticIpAddress`	This is an optional parameter. If Static load balancer IP needs to be set, then set staticIpAddressEnabled flag to true and provide value for staticIpAddress else random IP will be assigned by the metalLB from its IP Pool	**DataType:**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:**
`global.nodeSelector.nodeKey`	This is an optional parameter. global node selector key	**DataType:**String **Range:**
`global.nodeSelector.nodeValue`	This is an optional parameter. global node value key	**DataType:**String **Range:**
`global.logStorage`	This is a mandatory parameter. Set value for Ephemeral Storage Limits for logStorage	**DataType:**Integer **Range:** It can take values in integer and that values are used in MBs. **Default Value:**
`global.criticalStorage`	This is a mandatory parameter. Set value for Ephemeral Storage Limits for criticalStorage	**DataType:**Integer **Range:** It can take values in integer and that values are used in MBs. **Default Value:**

Table 4-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.ephemeralStorageLimit`	This is a mandatory parameter. Limits value for Ephemeral Storage.	**DataType:**Integer **Range:** It can take values in integer and that values are used in MBs. **Default Value:**

CNC Console IAM Configuration Parameters

This section includes information about the configuration parameters of the CNC Console IAM:

Global Parameters

This section includes information about the global parameters of the CNC Console IAM:

Table 4-2 Global Parameters

Parameter	Description	Details
`cncc-iam.global.iamServiceHttpPort`	This is a mandatory parameter. This should be same as kc.keycloak.service.httpPort	**DataType:**Integer **Range**: 0-65535 **Default Value:**
`cncc-iam.global.hook.image.name`	This is a mandatory parameter. Image name for the helm hook	**DataType:**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:** cncc/cncc-iam/hook
`cncc-iam.global.hook.image.tag`	This is a mandatory parameter. Image version tag for helm hook	**DataType:**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:**<current version>
`cncc-iam.global.hook.image.imagePullPolicy`	This is an optional parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:** IfNotPresent, Always, Never **Default Value:**IfNotPresent
`cncc-iam.global.hook.config.logLevel.root`	This is a mandatory parameter. Root log level for helm hook pod	**DataType:**String **Range:**It can take values like: WARN, DEBUG, INFO, etc. **Default Value:** INFO

Table 4-2 (Cont.) Global Parameters

Parameter	Description	Details
cncc-iam.global.publicHttpSignalingPort	This is a mandatory parameter. If https is enabled, this Port would be HTTP/1.0 Port (unsecured) If https is disabled, this Port would be HTTPS/1.0 Port (secured SSL)	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters. **Default Value:** 8080
cncc-iam.global.publicHttpsSignallingPort	This is a mandatory parameter. If https is enabled, this Port would be HTTP/1.0 Port (unsecured) If https is disabled, this Port would be HTTPS/1.0 Port (secured SSL)	**DataType:**String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters. **Default Value:** 8443
cncc-iam.global.staticIpAddressEnabled	This is an optional parameter. If Static load balancer IP needs to be set, then set staticIpAddressEnabled flag to true and provide value for staticIpAddress Else random IP will be assigned by the metalLB from its IP Pool	**DataType:**boolean **Range:** true or false **Default Value:** false
cncc-iam.global.staticIpAddress	This is an optional parameter. It is Static Ip and applicable only when ingress-gateway.cncc-iam.global.staticNodePortEnabled is true.	**DataType:**String **Range:** Valid IP address **Default Value:**10.75.212.60
cncc-iam.global.dbName	This is a mandatory parameter. It is the name of the database used for cncc-iam. User should create DB with the same name as provided here before deploying CNCC-IAM	**DataType:**String **Range:**Valid String **Default Value:**
cncc-iam.global.httpsEnabled	This is a mandatory parameter. To enable or disable https support	**DataType:**Boolean **Range:**True or False **Default Value:** False
cncc-iam.global.enableIncomingHttp	This is a mandatory parameter. Server Configuration for http and https support	**DataType:**Boolean **Range:**True or False **Default Value:** False

Table 4-2 (Cont.) Global Parameters

Parameter	Description	Details
cncc-iam.global.enableIncomingHttps	This is a mandatory parameter. Server Configuration for http and https support	**DataType:**Boolean **Range:**True or False **Default Value:** False
cncc-iam.global.ingressGwCertReloadEnabled	This is a mandatory parameter. If enabled, then certificates can be updated during run-time without up/restart of the application	**DataType:**Boolean **Range:**True or False **Default Value:** True

IAM Backend Parameters

This section includes information about the IAM backend parameters of the CNC Console IAM:

Table 4-3 IAM Backend Parameters

Parameter	Description	Details
cncc-iam.kc.ldaps.enabled	This is a mandatory parameter. The flag to enable or disable LDAPS feature. .	**DataType:**Boolean **Range:**True or False **Default Value:** False
cncc-iam.kc.ldaps.initContainersImage.name	This is a mandatory parameter. Image Name to be used for LDAPS.	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:**cncc/apigw-configurationinit
cncc-iam.kc.ldaps.initContainersImage.tag	This is a mandatory parameter. Image version tag for LDAPS.	**DataType:**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:**<Current Version>
cncc-iam.kc.ldaps.initContainersImage.pullPolicy	This is an optional parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent

Table 4-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
cncc-iam.kc.ldaps.service.customExtension.labels	This is an optional parameter. This can be used to add custom label(s) that are specific to service and will be created by cncc-iam helm chart.	**DataType:**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:**NA
cncc-iam.kc.ldaps.service.customExtension.annotations	This is an optional parameter. This can be used to add custom label(s) that are specific to service and will be created by cncc-iam helm chart.	**DataType:**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:**NA
cncc-iam.kc.ldaps.service.ssl.tlsVersion	This is a mandatory parameter. TLS Version.	**DataType:**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:** TLSv1.2
cncc-iam.kc.ldaps.service.ssl.caBundle.k8SecretName	This is a mandatory parameter. Name of the caBundle secret. Example: cncc-iam-kc-root-ca	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**cncc-iam-kc-root-ca
cncc-iam.kc.ldaps.service.ssl.caBundle.k8NameSpace	This is a mandatory parameter. Namespace of caBundle. Example: cncc.	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**cncc
cncc-iam.kc.ldaps.service.ssl.caBundle.fileName	This is a mandatory parameter. rsa caBundle file name. Example: caroot.cer	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**caroot.cer

Table 4-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
cncc-iam.kc.ldaps.service.ssl.trustStorePassword.k8SecretName	This is a mandatory parameter. Name of the caBundle secret. Example: cncc-iam-kc-root-ca	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**cncc-iam-kc-root-ca
cncc-iam.kc.ldaps.service.ssl.trustStorePassword.k8NameSpace	This is a mandatory parameter. Namespace of caBundle. Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**cncc
cncc-iam.kc.ldaps.service.ssl.trustStorePassword.fileName	This is a mandatory parameter. rsa caBundle file name. Example: ssl_truststore.txt	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**ssl_truststore.txt
cncc-iam.kc.ldaps.service.ssl.initialAlgorithm	This is a mandatory parameter. Name of the initial algorithm.	**DataType:**String **Range:** **Default Value:**RS256
cncc-iam.kc.ldaps.resources.limits.initServiceCpu	This is a mandatory parameter. Init Container CPU Limit.	**DataType:**String **Range:** **Default Value:**1Gi
cncc-iam.kc.ldaps.resources.limits.initServiceMemory	This is a mandatory parameter. Init Container Memory Limit.	**DataType:**String **Range:** **Default Value:**1Gi
cncc-iam.kc.ldaps.resources.requests.initServiceCpu	This is a mandatory parameter. Init Container CPU Limit.	**DataType:**String **Range:** **Default Value:**0.5Gi
cncc-iam.kc.ldaps.resources.requests.initServiceMemory	This is a mandatory parameter. Init Container Memory Limit.	**DataType:**String **Range:** **Default Value:**0.5Gi
cncc-iam.kc.ldaps.ports.containerPort	This is a mandatory parameter. ContainerPort represents a network port in a single container.	**DataType:**String **Range:** 0-65535. **Default Value:**8086
cncc-iam.kc.ldaps.logLevel.initContainer	This is a mandatory parameter. Log level for initContainer logs.	**DataType:**String **Range:**WARN, DEBUG, INFO, TRACE **Default Value:**INFO

Table 4-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
cncc-iam.kc.extraContainers	This is a mandatory parameter. To enable or disable debug tools container.	**DataType:**enum **Range:**DISABLED, ENABLED, USE_GLOBAL_VALUE **Default Value:**
cncc-iam.kc.healthcheck.image.name	This is a mandatory parameter. Image name for the helm healthcheck.	**DataType:**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:** cncc/cncc-iam/healthcheck
cncc-iam.kc.healthcheck.image.tag	This is a mandatory parameter. Image version tag for helm healthcheck.	**DataType:**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:** <Current Version>
cncc-iam.kc.healthcheck.image.pullpolicy	This is a mandatory parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent
cncc-iam.kc.healthcheck.logLevel.root	This is a mandatory parameter. Root log level for helm healthcheck container.	**DataType:**String **Range:**WARN, DEBUG, INFO, etc. **Default Value:** INFO
cncc-iam.kc.healthcheck.resources.limits.cpu	This is a mandatory parameter. It limits the number of CPUs to be used by the helm test container.	**DataType:**String **Range:**Valid floating point value between 0 and 1 **Default Value:** 0.5
cncc-iam.kc.healthcheck.resources.limits.memory	This is a mandatory parameter. It limits the number of memory to be used by the helm test container.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi etc. **Default Value:** 0.5Gi
cncc-iam.kc.healthcheck.limits.logStorage	This is a mandatory parameter. It limits the logStorage (Ephemeral storage)	**DataType:**Integer **Range:**Valid Integer value **Default Value:**
cncc-iam.kc.healthcheck.limits.criticalStorage	This is a mandatory parameter. It limits the criticalStorage (Ephemeral storage)	**DataType:**Integer **Range:**Valid Integer value **Default Value:**

Table 4-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
cncc-iam.kc.healthcheck.resources.requests.cpu	This is a mandatory parameter. The minimum amount of CPUs required.	**DataType:**String **Range:**Valid floating point value between 0 and 1 **Default Value:** 0.3
cncc-iam.kc.healthcheck.resources.requests.memory	This is a mandatory parameter. The minimum amount of memory required.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi etc. **Default Value:** 0.3Gi
cncc-iam.kc.healthcheck.limits.logStorage	This is a mandatory parameter. Minimum memory for logStorage (Ephemeral storage)	**DataType:**Integer **Range:**Valid Integer value **Default Value:**
cncc-iam.kc.healthcheck.limits.criticalStorage	This is a mandatory parameter. Minimum memory for criticalStorage (Ephemeral storage)	**DataType:**Integer **Range:**Valid Integer value **Default Value:**
cncc-iam.kc.service.customExtension.labels	This is an optional parameter. This can be used to add custom label(s) that are specific to service and will be created by cncc-iam helm chart.	**DataType:**String **Range:** **Default Value:** NA
cncc-iam.kc.service.customExtension.annotations	This is an optional parameter. This can be used to add custom annotations that are specific to service and will be created by cncc-iam helm chart.	**DataType:**String **Range:** **Default Value:** NA
cncc-iam.kc.keycloak.image.name	This is a mandatory parameter. Image Name to be used for cncc-iam micro service.	**DataType:**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:** cncc/cncc-iam
cncc-iam.kc.keycloak.image.tag	This is a mandatory parameter. Image Tag to be used for cncc-iam micro service.	**DataType:**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. **Default Value:** <Current Version>

Table 4-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
`cncc-iam.kc.keycloak.image.pullpolicy`	This is a mandatory parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent
`cncc-iam.kc.keycloak.serviceAccount.create`	This is an optional parameter. Flag for creating service account.	**DataType:**Boolean **Range:**True or False **Default Value:** False
`cncc-iam.kc.keycloak.serviceAccount.name`	This is an optional parameter. The name of service account. Applicable only if keycloak.serviceAccount.create is set to 'true'. If keycloak.serviceAccount.name is kept as empty, a default service account with name 'cncc-iam' is created by CNCC, otherwise user has to create the service account and provide its name here. `kubectl create serviceaccount <name> -n <namespace>`	**DataType:**String **Range:** values will be set by global.serviceAccountName **Default Value:**
`cncc-iam.kc.keycloak.username`	This is a mandatory parameter .It is the name of cncc-iam user as given by the user. Example: admin	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. **Default Value:**admin
`cncc-iam.kc.keycloak.existingSecret`	This is a mandatory parameter. It specifies an existing secret name to be used for the admin password. Example: cncc-iam-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters **Default Value:** cncc-iam-secret
`cncc-iam.kc.keycloak.existingSecretKey`	This is a mandatory parameter. Applicable only if keycloak.existingSecret is provided. It is the key in the existing secret that stores the password. Example: iamAdminPasswordKey	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters **Default Value:** iamAdminPasswordKey

Table 4-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
cncc-iam.kc.keycloak.service.httpPort	This is an optional parameter. It is the port number which makes cncc-iam service visible to other services running within the same kubernetes cluster.	**DataType:**String **Range:** Values will be set by global.iamServiceHttpPort **Default Value:** 8285
cncc-iam.kc.keycloak.resources.limits.cpu	This is a mandatory parameter. It limits the number of CPUs to be used by the helm test container.	**DataType:**String **Range:**Valid floating point value between 0 and 1 **Default Value:**
cncc-iam.kc.keycloak.resources.limits.memory	This is a mandatory parameter. It limits the number of memory to be used by the helm test container.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi etc. **Default Value:**
cncc-iam.kc.keycloak.resources.requests.cpu	This is a mandatory parameter. The minimum amount of CPUs required.	**DataType:**String **Range:**Valid floating point value between 0 and 1. **Default Value:**
cncc-iam.kc.keycloak.resources.requests.memory	This is a mandatory parameter. The minimum amount of memory required.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi etc. **Default Value:**
cncc-iam.kc.keycloak.persistence.dbVendor	This is a mandatory parameter. It is the database vendor name Example: mysql	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. **Default Value:** mysql
cncc-iam.kc.keycloak.persistence.dbName	This is a mandatory parameter. It is the name of the database used for cncc-iam. User should create DB with the same name as provided here before deploying CNCC-IAM Example: cnccdb	**DataType:**String **Range:** Values will be set by global.dbName **Default Value:** This values gets read from global variable *mySqlDbRef, no changes needed here.
cncc-iam.kc.keycloak.persistence.dbHost	This is a mandatory parameter. It the hostname for persistence db Example: mysql-sds.default.svc.cluster.local.	**DataType:**String **Range:** Values will be set by global.dbHost **Default Value:** This values gets read from global variable *mySqlHostRef, no changes needed here.

Table 4-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
`cncc-iam.kc.keycloak.persistence.dbPort`	This is a mandatory parameter. It is the db port for cncc-iam. Example: 3306	**DataType:**Integer **Range:** Values will be set by global.dbPort **Default Value:** This values gets read from global variable *mySqlPortRef, no changes needed here.
`cncc-iam.kc.keycloak.persistence.existingSecret`	This is a mandatory parameter. It specifies an existing secret to be used for mysql username and password Example: cncc-db-secret	**DataType:**String **Range:** Values will be set by global.secretName **Default Value:** This values gets read from global variable *mySqlSecretNameRef, no changes needed here.
`cncc-iam.kc.keycloak.persistence.existingSecretPasswordKey`	This is a mandatory parameter. It is the key in the existing secret that stores the password Example: dbPasswordKey	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters **Default Value:** dbPasswordKey
`cncc-iam.kc.keycloak.persistence.existingSecretUsernameKey`	This is an optional parameter. It is the key in the existing secret that stores the username Example: `dbUserNameKey`	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters. **Default Value:** dbUserNameKey
`cncc-iam.kc.keycloak.nodeselector`	This is an optional parameter. Node selector key specific to chart (note this will be looked first and then if not present global node key will be picked)	**DataType:**String **Range:** **Default Value:** NA

Ingress Gateway Parameters

This section includes information about the Ingress gateway parameters of the CNC Console IAM:

Table 4-4 Ingress Gateway Parameters

Parameter	Description	Details
cncc-iam.ingress-gateway.extraContainers	This is a mandatory parameter. To enable or disable debug tools container	**DataType:**enum **Range:**DISABLED, ENABLED, USE_GLOBAL_VALUE **Default Value:** USE_GLOBAL_VALUE
cncc-iam.ingress-gateway.prefix	This is a mandatory parameter. Metrics Instance Identifier to uniquely identify both Manager CNCC Core and Agent CNCC IAM metrics	**DataType:**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:**'cncc-iam'
cncc-iam.ingress-gateway.image.name	This is a mandatory parameter. Image Name to be used for "cncc-iam.ingress-gateway" micro service	**DataType:**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:** cncc/cncc-apigateway
cncc-iam.ingress-gateway.image.tag	This is a mandatory parameter. Image Tag to be used for "cncc-iam.ingress-gateway" micro service	**DataType:**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. **Default Value:** <Current Version>
cncc-iam.ingress-gateway.image.pullPolicy	This is a mandatory parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:**IfNotPresent
cncc-iam.ingress-gateway.initContainersImage.name	This is a mandatory parameter. Image Name to be used for init container	**DataType:**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:** cncc/apigw-configurationinit
cncc-iam.ingress-gateway.initContainersImage.tag	This is a mandatory parameter. Image tag to be used for init container	**DataType:**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. **Default Value:**<Current Version>

Table 4-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
cncc-iam.ingress-gateway.initContainersImage.pullPolicy	This is a mandatory parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent
cncc-iam.ingress-gateway.updateContainersImage.name	This is a mandatory parameter. Image Name to be used for update container	**DataType:**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:** cncc/apigw-configurationupdate
cncc-iam.ingress-gateway.updateContainersImage.tag	This is a mandatory parameter. Image tag to be used for update container	**DataType:**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. **Default Value:** <Current Version>
cncc-iam.ingress-gateway.updateContainersImage.pullPolicy	This is a mandatory parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent
cncc-iam.ingress-gateway.service.ssl.tlsVersion	This is a mandatory parameter. TLS Version	**DataType:**String **Range:** **Default Value:** TLSv1.2
cncc-iam.ingress-gateway.service.ssl.privateKey.k8SecretName	This is a mandatory parameter. Name of the privatekey secret Example: cncc-iam-cncc-iam.ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc-iam-cncc-iam.ingress-secret
cncc-iam.ingress-gateway.service.ssl.privateKey.k8NameSpace	This is a mandatory parameter. Namespace of privatekey Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**cncc

Table 4-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
cncc-iam.ingress-gateway.service.ssl.privateKey.rsa.fileName	This is a mandatory parameter. rsa private key file name Example: rsa_private_key_pkcs1.pem	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** rsa_private_key_pkcs1.pem
cncc-iam.ingress-gateway.service.ssl.privateKey.ecdsa.fileName	This is a mandatory parameter. ecdsa private key file name Example: ssl_ecdsa_private_key.pem	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** ssl_ecdsa_private_key.pem
cncc-iam.ingress-gateway.service.ssl.certificate.k8SecretName	This is a mandatory parameter. Name of the certificate secret Example: cncc-iam-cncc-iam.ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc-iam-cncc-iam.ingress-secret
cncc-iam.ingress-gateway.service.ssl.certificate.k8NameSpace	This is a mandatory parameter. Namespace of certificate Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc
cncc-iam.ingress-gateway.service.ssl.certificate.rsa.fileName	This is a mandatory parameter. rsa certificate file name Example: ssl_rsa_certificate.crt	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** ssl_rsa_certificate.crt
cncc-iam.ingress-gateway.service.ssl.certificate.ecdsa.fileName	This is a mandatory parameter. ecdsa certificate file name Example: ssl_ecdsa_certificate.crt	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** ssl_ecdsa_certificate.crt

Table 4-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
cncc-iam.ingress-gateway.service.ssl.caBundle.k8SecretName	This is a mandatory parameter. Name of the caBundle secret Example: cncc-iam-cncc-iam.ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** ncc-iam-cncc-iam.ingress-secret
cncc-iam.ingress-gateway.service.ssl.caBundle.k8NameSpace	This is a mandatory parameter. Namespace of caBundle Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc
cncc-iam.ingress-gateway.service.ssl.caBundle.f**ileName**	This is a mandatory parameter. rsa caBundle file name Example: caroot.cer	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:**caroot.cer
cncc-iam.ingress-gateway.service.ssl.initialAlgorithm	This is a mandatory parameter. This is initial Algorithm Example: RSA256	**DataType:**String **Range:** **Default Value:** RS256
cncc-iam.ingress-gateway.service.customExtension.labels	This is an optional parameter. This can be used to add custom label(s) to cncc-iam.ingress-gateway Service.	**DataType:**String **Range:**Custom Labels that needs to be added to cncc-iam.ingress-gateway specific Service. **Default Value:**{}
cncc-iam.ingress-gateway.service.customExtension.annotations	This is an optional parameter. This can be used to add custom annotation(s) to cncc-iam.ingress-gateway Service.	**DataType:**String **Range:**Custom Annotations that needs to be added to cncc-iam.ingress-gateway specific Services. **Default Value:**{}
cncc-iam.ingress-gateway.deployment.customExtension.labels	This is an optional parameter. This can be used to add custom label(s) to cncc-iam.ingress-gateway Deployment.	**DataType:**String **Range:**Custom Labels that needs to be added to cncc-iam.ingress-gateway specific Deployment. **Default Value:** {}
cncc-iam.ingress-gateway.deployment.customExtension.annotations	This is an optional parameter. This can be used to add custom annotation(s) to cncc-iam.ingress-gateway Deployment.	**DataType:**String **Range:**Custom Annotations that needs to be added to cncc-iam.ingress-gateway specific Deployment. **Default Value:**{}

Table 4-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
cncc-iam.ingress-gateway.ports.containerPort	This is a mandatory parameter. ContainerPort represents a network port in a single container	**DataType:**String **Range:**0-65535. **Default Value:**8081
cncc-iam.ingress-gateway.ports.containersslPort	This is a mandatory parameter. This is container ssl port	**DataType:**String **Range:** **Default Value:**8443
cncc-iam.ingress-gateway.ports.actuatorPort	This is a mandatory parameter. This is actuator Port	**DataType:**String **Range:** **Default Value:** 9090
cncc-iam.ingress-gateway.log.level.root	This is a mandatory parameter. It is the level at which user wants to see the logs. E.g. WARN	**DataType:**String **Range:** WARN, DEBUG, INFO, TRACE etc. **Default Value:** WARN
cncc-iam.ingress-gateway.log.level.cncc-iam.ingress	This is a mandatory parameter. Log level for cncc-iam.ingress logs	**DataType:**String **Range:**WARN, DEBUG, INFO, TRACE etc. **Default Value:** INFO
cncc-iam.ingress-gateway.log.level.cncc.security	This is a mandatory parameter. Log level for cncc security logs	**DataType:**String **Range:** WARN, DEBUG, INFO, TRACE etc. **Default Value:** INFO
cncc-iam.ingress-gateway.log.level.cncc.root	This is a mandatory parameter. Log level for cncc root logs	**DataType:**String **Range:** WARN, DEBUG, INFO, TRACE etc. **Default Value:**WARN
cncc-iam.ingress-gateway.readinessProbe.initialDelaySeconds	This is a mandatory parameter. It tells the kubelet that it should wait the mentioned seconds before performing the first probe	**DataType:**String **Range:**0-65535 **Default Value:** 30
cncc-iam.ingress-gateway.readinessProbe.timeoutSeconds	This is a mandatory parameter. It is the number of seconds after which the probe times out	**DataType:**String **Range:**0-65535 **Default Value:**3
cncc-iam.ingress-gateway.readinessProbe.periodSeconds	This is a mandatory parameter. It specifies that the kubelet should perform a liveness probe every xx seconds	**DataType:**String **Range:** 0-65535 **Default Value:** 10

Table 4-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
cncc-iam.ingress-gateway.readinessProbe.successThreshold	This is a mandatory parameter. Minimum consecutive successes for the probe to be considered successful after having failed	**DataType:**String **Range:**0-65535. **Default Value:** 1
cncc-iam.ingress-gateway.readinessProbe.failureThreshold	This is a mandatory parameter. When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up	**DataType:**String **Range:**0-65535. **Default Value:** 3
cncc-iam.ingress-gateway.livenessProbe.initialDelaySeconds	This is a mandatory parameter. It tells the kubelet that it should wait the mentioned seconds before performing the first probe	**DataType:**String **Range:**0-65535. **Default Value:** 30
cncc-iam.ingress-gateway.livenessProbe.timeoutSeconds	This is a mandatory parameter. It is the number of seconds after which the probe times out	**DataType:**String **Range:**0-65535. **Default Value:** 3
cncc-iam.ingress-gateway.livenessProbe.periodSeconds	This is a mandatory parameter. It specifies that the kubelet should perform a liveness probe every xx seconds	**DataType:**String **Range:**0-65535. **Default Value:** 15
cncc-iam.ingress-gateway.livenessProbe.successThreshold	This is a mandatory parameter. Minimum consecutive successes for the probe to be considered successful after having failed	**DataType:**String **Range:**0-65535. **Default Value:** 1
cncc-iam.ingress-gateway.livenessProbe.failureThreshold	This is a mandatory parameter. When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up	**DataType:**String **Range:**0-65535. **Default Value:** 3
cncc-iam.ingress-gateway.resources.limits.cpu	This is a mandatory parameter. It limits the number of CPUs to be used by the microservice.	**DataType:**String **Range:**Valid floating point value between 0 and 1 **Default Value:**
cncc-iam.ingress-gateway.resources.limits.initServiceCpu	This is a mandatory parameter. Init Container CPU Limit	**DataType:**String **Range:** **Default Value:** 1

Table 4-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
cncc-iam.ingress-gateway.resources.limits.updateServiceCpu	This is a mandatory parameter. Update Container CPU Limit	**DataType:**String **Range:** **Default Value:** 1
cncc-iam.ingress-gateway.resources.limits.memory	This is a mandatory parameter. It limits the memory utilization by the "cncc-cmservice" microservice. By default, it is set to '2'.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi etc **Default Value:** 2Gi
cncc-iam.ingress-gateway.resources.limits.updateServiceMemory	This is a mandatory parameter. Update Container Memory Limit	**DataType:**String **Range:** **Default Value:** 1Gi
cncc-iam.ingress-gateway.resources.limits.initServiceMemory	This is a mandatory parameter. Init Container Memory Limit	**DataType:**String **Range:** **Default Value:** 1Gi
cncc-iam.ingress-gateway.resources.requests.cpu	This is a mandatory parameter. It limits the number of CPUs to be used by the "cncc-cmservice" microservice. By default, it is set to '2'.	**DataType:**String **Range:**Valid floating point value between 0 and 1 **Default Value:** 1
cncc-iam.ingress-gateway.resources.requests.initServiceCpu	This is a mandatory parameter. Init Container CPU Limit	**DataType:**String **Range:** **Default Value:** 1
cncc-iam.ingress-gateway.resources.requests.updateServiceCpu	This is a mandatory parameter. Update Container CPU for requests	**DataType:**String **Range:** **Default Value:** 1
cncc-iam.ingress-gateway.resources.requests.memory	This is a mandatory parameter. It limits the memory utilization by the "cncc-cmservice" microservice. By default, it is set to '2'.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi etc. **Default Value:** 1Gi
cncc-iam.ingress-gateway.resources.requests.updateServiceMemory	This is a mandatory parameter. Update Container Memory for requests	**DataType:**String **Range:** **Default Value:** 1Gi
cncc-iam.ingress-gateway.resources.requests.initServiceMemory	This is a mandatory parameter. Init Container Memory for requests	**DataType:**String **Range:** **Default Value:** 1Gi
cncc-iam.ingress-gateway.resources.target.averageCpuUtil	This is a mandatory parameter. It gives the average CPU utilization percentage.	**DataType:**String **Range:**A value in between 0-100 **Default Value:**80

Table 4-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
`cncc-iam.ingress-gateway.service.ssl.keyStorePassword.k8SecretName`	This is a mandatory parameter. Name of the keyStorePassword secret Example: cncc-iam-cncc-iam.ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:** cncc-iam-cncc-iam.ingress-secret
`cncc-iam.ingress-gateway.service.ssl.keyStorePassword.k8NameSpace`	This is a mandatory parameter. Namespace of keyStorePassword Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc
`cncc-iam.ingress-gateway.service.ssl.keyStorePassword.fileName`	This is a mandatory parameter. File name that has password for keyStore Example: ssl_keystore.txt	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**ssl_keystore.txt
`cncc-iam.ingress-gateway.service.ssl.trustStorePassword.k8SecretName`	This is a mandatory parameter. Name of the trustStorePassword secret Example: cncc-iam-cncc-iam.ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**cncc-iam-cncc-iam.ingress-secret
`cncc-iam.ingress-gateway.service.ssl.trustStorePassword.k8NameSpace`	This is a mandatory parameter. Namespace of trustStorePassword Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:** cncc
`cncc-iam.ingress-gateway.service.ssl.trustStorePassword.fileName`	This is a mandatory parameter. File name that has password for trustStore Example: ssl_truststore.txt	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:** ssl_truststore.txt

Table 4-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
cncc-iam.ingress-gateway.cipherSuites	This is a mandatory parameter (if cncc-iam.ingressgateway.enableIncomingHttps is true). Allowed CipherSuites for TLS1.2	**DataType:**List[String] **Range:** TLS_ECDHE_ ECDSA_WITH_AES_256_ GCM_SHA38 4 TLS_ECDHE_ RSA_WITH_ AES_256_GC M_SHA384 TLS_ECDHE_ RSA_WITH_ CHACHA20_ POLY1305_S HA256 TLS_DHE_RS A_WITH_AES_256_GCM_ SHA384 TLS_ECDHE_ ECDSA_WITH_AES_128_ GCM_SHA25 6 TLS_ECDHE_ RSA_WITH_ AES_128_GC M_SHA256 **Default Value:**
cncc-iam.ingress-gateway.initssl	This is a mandatory parameter. To Initialize SSL related infrastructure in init/update container	**DataType:**String **Range:** Values will be set by global.httpsEnabled **Default Value:** NA
cncc-iam.ingress-gateway.enableIncomingHttp	This is a mandatory parameter. Server Configuration for http and https support.	**DataType:**String **Range:** Values will be set by global.enableIncomingHttp **Default Value:**
cncc-iam.ingress-gateway.enableIncomingHttps	This is a mandatory parameter. Server Configuration for http and https support.	**DataType:**String **Range:** Values will be set by global.enableIncomingHttps **Default Value:**
cncc-iam.ingress-gateway.needClientAuth	This is a mandatory parameter. This must be true if client certificate identity is required in the header x-custom-cncc-iam.ingress-client-identity Note: This parameter will be set to true only in case of ACNCC-CORE deployment	**DataType:**Boolean **Range:** Values will be set by global.needClientAuth **Default Value:**
cncc-iam.ingress-gateway.cncc-iam.ingressGwCertReloadEnabled	This is a mandatory parameter. If enabled, then certificates can be updated during run-time without up/restart of the application	**DataType:**Boolean **Range:** True or False **Default Value:** Values will be set by global.cncc-iam.ingressGwCertReloadEnabled

Table 4-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
`cncc-iam.ingress-gateway.clusterDomain`	This is a mandatory parameter. Cluster Domain where cncc-iam will be deployed.	**DataType:**string **Range:** **Default Value:** Values will be set by global.clusterDomain
`cncc-iam.ingress-gateway.cncc.securitylogEnabled`	This is a mandatory parameter. This flag is to enable/disable security logs for cncc.	**DataType:**Boolean **Range:** True or False **Default Value:** True
`cncc-iam.ingress-gateway.cncc.iam.port`	This is a mandatory parameter. This should be same as kc.keycloak.service.httpPort	**DataType:**Integer **Range:** Values will be set by global.iamServiceHttpPort **Default Value:**

M-CNCC Core and A-CNCC Core Configuration Options

This section includes information about the configuration parameters of the A-CNCC Core and M-CNCC Core.

> **Note:**
>
> The following tables are common for mcncc-core and acncc-core section in *occncc_custom_values_<version>.yaml* file.
>
> The options mentioned in the following tables can be read as *mcncc-core.<field Name>* or *acncc-core.<field Name>*.
>
> Example: The *global.dbName* can be read as *mcncc-core.global.dbName* or *acncc-core.global.dbName*.

Global Parameters

This section includes information about the global parameters of the CNC Console Core:

Table 4-5 Global Parameters

Parameter	Description	Details
`global.publicHttpSignalingPort`	This is an optional parameter. It is the port on which Ingress Gateway service is exposed # If httpsEnabled is false, this Port would be HTTP/2.0 Port (unsecured) publicHttpSignalingPort: 80	**DataType:**Integer **Range:**0-65535. **Default Value:**80
`global.publicHttpsSignallingPort`	This is an optional parameter. It is the port on which Ingress Gateway service is exposed # If httpsEnabled is true, this Port would be HTTPS/2.0 Port (secured SSL)	**DataType:**Integer **Range:**0-65535. **Default Value:**443
`global.metalLbIpAllocationEnabled`	This is an optional parameter. This field enables or disables IP Address allocation from Metallb Pool	**DataType:**Boolean **Range:**True or False **Default Value:** False
`global.staticIpAddress`	This is an optional parameter. If Static load balancer IP needs to be set, then set staticIpAddressEnabled flag to true and provide value for staticIpAddress else random IP will be assigned by the metalLB from its IP Pool	**DataType:**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:**
`global.dbName`	This is a mandatory parameter. It is the name of the database used for M-CNCC OR A-CNCC Core, user should create DB with unique name before deploying M-CNCC OR A-CNCC Core	**DataType:**String **Range:**Valid string **Default Value:** • mcncccommonconfig in case of mcncc-core • mcncccommonconfig in case of mcncc-core
`global.httpsEnabled`	This is an optional parameter. To Initialize SSL related infrastructure in init/update container	**DataType:**Boolean **Range:**True or False **Default Value:** False
`global.enableIncomingHttp`	This is an optional parameter. Server Configuration for http and https support.	**DataType:**Boolean **Range:**True or False **Default Value:** False
`global.enableIncomingHttps`	This is an optional parameter. Server Configuration for http and https support.	**DataType:**Boolean **Range:**True or False **Default Value:** False

Table 4-5 (Cont.) Global Parameters

Parameter	Description	Details
global.needClientAuth	This must be true if client certificate identity is required in the header x-custom-ingress-client-identity Note: This parameter will be set to true only in case of ACNCC-CORE deployment.	**DataType:**Boolean **Range:**True or False **Default Value:** False
global.ingressGwCertReloadEnabled	This is an optional parameter. If enabled, then certificates can be updated during run-time without up/restart of the application.	**DataType:**Boolean **Range:**True or False **Default Value:** False

Core Backend Parameters

This section includes information about the core backend parameters of the CNC Console Core.

Table 4-6 Core Backend Parameters

Attribute Name	Description	Details
cmservice.envLoggingLevelApp	This is an optional parameter. It is the level at which user wants to see the logs. Example: WARN	**DataType:**String **Range:**WARN, DEBUG, INFO, TRACE **Default Value:** WARN
cmservice.image.name	This is a mandatory parameter. Image Name to be used for "cncc-cmservice" micro service	**DataType:**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:** cncc/cncc-cmservice
cmservice.image.tag	This is a mandatory parameter. Image Tag to be used for "cncc-cmservice" micro service	**DataType:**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:** <Current Version>
cmservice.image.pullPolicy	This is a mandatory parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent

Table 4-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
cmservice.persistence.dbName	This is an optional parameter. It is the name of the database used for cncc-core. User should create DB with the same name as provided here before deploying CNCC-Core This values gets read from global variable *mySqlDbRef, no changes needed here Example: cncccommonconfig	**DataType:**String **Range:**Valid String **Default Value:**
cmservice.persistence.dbHost	This is an optional parameter. It the hostname for persistence db This values gets read from global variable *mySqlHostRef, no changes needed here Example: mysql-sds.default.svc.cluster.local	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. **Default Value:**
cmservice.persistence.dbPort	This is an optional parameter. It is the db port for cncc-core This values gets read from global variable *mySqlPortRef, no changes needed here Example: 3306	**DataType:**Integer **Range:**0-65535 **Default Value:**
cmservice.persistence.existingSecret	This is an optional parameter. It specifies an existing secret to be used for mysql username and password This values gets read from global variable *mySqlSecretNameRef, no changes needed here Example: cncc-db-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters **Default Value:**

Table 4-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
cmservice.persistence.existingSecretPasswordKey	This is an optional parameter. It is the key in the existing secret that stores the password Example: dbPasswordKey	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters **Default Value:** dbPasswordKey
cmservice.persistence.existingSecretUsernameKey	This is an optional parameter. It is the key in the existing secret that stores the username Example: dbUserNameKey	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters **Default Value:** dbUserNameKey
cmservice.resources.limits.cpu	This is an optional parameter. It limits the number of CPUs to be used by the "cncc-cmservice" microservice. By default, it is set to '2'.	**DataType:**Float **Range:**Valid floating point value between 0 and 1 **Default Value:**
cmservice.resources.limits.memory	This is an optional parameter. It limits the memory utilization by the "cncc-cmservice" microservice. By default, it is set to '2'.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi **Default Value:** 2Gi
cmservice.resources.limits.logStorage	This is an optional parameter. It limits the logStorage (ephemeral storage) to be used by the helm test pod.	**DataType:**Integer **Range:**Values will be set by global.ephemeralStorage.limits.containerLogStorage **Default Value:** 2Gi
cmservice.resources.limits.criticalStorage	This is an optional parameter. It limits the criticalStorage (ephemeral storage) to be used by the helm test pod.	**DataType:**Integer **Range:**Values will be set by global.ephemeralStorage.limits.containerCrititcalStorage **Default Value:** 2Gi
cmservice.resources.requests.cpu	This is an optional parameter. It provides a given number of CPUs for the "cncc-cmservice" microservice.	**DataType:**Float **Range:**Valid floating point value between 0 and 1 **Default Value:**1

Table 4-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
cmservice.resources.requests.memory	This is an optional parameter. It provides a given amount of memory for the "cncc-cmservice" microservice. By default, it is set to '1.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi **Default Value:** 1
cmservice.resources.requests.logStorage	This is an optional parameter. he minimum amount of logStorage (ephemeral storage)	**DataType:**Integer **Range:**Values will be set by global.ephemeralStorage.requests.containerLogStorage
cmservice.resources.requests.criticalStorage	This is an optional parameter. The minimum amount of criticalStorage (ephemeral storage)	**DataType:**Integer **Range:**Values will be set by global.ephemeralStorage.requests.containerCrititcalStorage
cmservice.extraContainers	This is a mandatory parameter. To enable or disable debug tools container	**DataType:**enum **Range:** DISABLED, ENABLED, USE_GLOBAL_VALUE **Default Value:** USE_GLOBAL_VALUE
cmservice.deployment.customExtension.labels	This is an optional parameter. This can be used to add custom label(s) to all kubernetes resources that will be created by cmservice helm chart.	**DataType:**String **Range:**Custom Labels that needs to be added to all the cmservice kubernetes resources
cmservice.deployment.customExtension.annotations	This is an optional parameter. This can be used to add custom annotation(s) to all kubernetes resources that will be created by cmservice helm chart.	**DataType:**String **Range:**Custom Annotations that needs to be added to all the cmservice kubernetes resources
cmservice.deployment.envSystemName	This is a mandatory parameter. This is the name of product which appears as brand name and can be used to mention site name as well. Example: envSystemName: CNCC	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**
cmservice.deployment.envNFVersion	This is a mandatory parameter. This is the version of product which appears with brand name. Example: envNFVersion: 22.3.0	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** <Current Version>

Table 4-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
cmservice.deployment.cmWindowName	This is the name of the window that appears on the browser tab. Example: cmWindowName: CNCC	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:** CNCC
cmservice.deployment.nodeSelectorEnabled	This is an optional parameter. NodeSelector is the simplest recommended form of node selection constraint. NodeSelector is a field of PodSpec. It specifies a map of key-value pairs. For the pod to be eligible to run on a node, the node must have each of the indicated key-value pairs as labels	**DataType:**Boolean **Range:**True or False **Default Value:**False
cmservice.deployment.nodeSelectorKey	This is an optional parameter. Node Selector Key	**DataType:**Integer **Range:** **Default Value:** zone.
cmservice.deployment.nodeSelectorValue	This is an optional parameter. Node Selector value	**DataType:**Integer **Range:** **Default Value:** app
cmservice.deployment.livenessProbe.initialDelaySeconds	This is an optional parameter. It tells the kubelet that it should wait second before performing the first probe	**DataType:**Integer **Range:**0-65535. **Default Value:** 60
cmservice.deployment.livenessProbe.periodSeconds	This is an optional parameter. It specifies that the kubelet should perform a liveness probe every xx seconds	**DataType:**Integer **Range:**0-65535. **Default Value:** 3
cmservice.deployment.livenessProbe.timeoutSeconds	This is an optional parameter. It is the number of seconds after which the probe times out	**DataType:**Integer **Range:** 0-65535. **Default Value:** 15
cmservice.deployment.livenessProbe.successThreshold	This is an optional parameter. Minimum consecutive successes for the probe to be considered successful after having failed	**DataType:**Integer **Range:**0-65535. **Default Value:** 1

Table 4-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
cmservice.deployment.livenessProbe.failureThreshold	This is an optional parameter. When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up	**DataType:**Integer **Range:**0-65535. **Default Value:** 3
cmservice.deployment.readinessProbe.initialDelaySeconds	This is an optional parameter. It tells the kubelet that it should wait second before performing the first probe	**DataType:**Integer **Range:** 0-65535. **Default Value:** 20
cmservice.deployment.readinessProbe.timeoutSeconds	This is an optional parameter. It is the number of seconds after which the probe times out	**DataType:**Integer **Range:** 0-65535. **Default Value:** 3
cmservice.deployment.readinessProbe.periodSeconds	This is an optional parameter. It specifies that the kubelet should perform a liveness probe every xx seconds	**DataType:**Integer **Range:**0-65535. **Default Value:** 10
cmservice.deployment.readinessProbe.successThreshold	This is an optional parameter. Minimum consecutive successes for the probe to be considered successful after having failed	**DataType:**Integer **Range:**0-65535. **Default Value:** 1
cmservice.deployment.readinessProbe.failureThreshold	This is an optional parameter. When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up	**DataType:**Integer **Range:** 0-65535. **Default Value:** 3
cmservice.deployment.dependenciesLogging[].name	This is an optional parameter. Name of the package that for which log level is to be set. Eg: logging.level.org.springframework	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**
cmservice.deployment.dependenciesLogging[].value	This is an optional parameter. It is the level at which user wants to see the logs. Example: WARN	**DataType:**String **Range:**WARN, DEBUG, INFO, TRACE **Default Value:**

Table 4-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
cmservice.service.customExtension.labels	This is an optional parameter. This can be used to add custom label(s) to all kubernetes resources that will be created by cmservice helm chart.	**DataType:**String **Range:**Custom Labels that needs to be added to all the cmservice kubernetes resources
cmservice.service.customExtension.annotations	This is an optional parameter. This can be used to add custom annotation(s) to all kubernetes resources that will be created by cmservice helm chart.	**DataType:**String **Range:**Custom Annotations that needs to be added to all the cmservice kubernetes resources
cmservice.service.type	This is an optional parameter. It is used to decide where user wants to expose the service from outside the Kubernetes cluster or not.	**DataType:**String **Range:** **Default Value:** ClusterIP
cmservice.servicePorts.cmServiceHttp	This is an optional parameter. It is the port number which makes cmservice visible to other services running within the same kubernetes cluster	**DataType:**Integer **Range:**0-65535
cmservice.containerPorts.monitoringHttp	This is an optional parameter. It is the monitoring container port for cm service.	**DataType:**Integer **Range:**0-65535 **Default Value:**
cmservice.containerPorts.cmServiceHttp	This is an optional parameter. It is the container port for cm service.	**DataType:**Integer **Range:**0-65535 **Default Value:**

Ingress Gateway Parameters

This section includes information about the Ingress Gateway parameters of the CNC Console Core.

Table 4-7 Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.extraContainers	This is a mandatory parameter. To enable or disable debug tools container	**DataType:**enum **Range:**DISABLED, ENABLED, USE_GLOBAL_VALUE **Default Value:** USE_GLOBAL_VALUE
ingress-gateway.prefix	This is a mandatory parameter. Metrics Instance Identifier to uniquely identify CNCC Core metrics	**DataType:**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:** /grafana
ingress-gateway.image.name	This is a mandatory parameter. It is the image name of the Ingress Gateway as provided by the user	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:** cncc/cncc-apigateway
ingress-gateway.image.tag	This is a mandatory parameter. Image Tag to be used for Ingress Gateway.	**DataType:**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 **Default Value:** <Current Version>
ingress-gateway.image.pullPolicy	This is an optional parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent
ingress-gateway.initContainersImage.name	This is a mandatory parameter. Image Name to be used for "cncc-cmservice" microservice	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:** cncc/apigw-configurationinit
ingress-gateway.initContainersImage.tag	This is a mandatory parameter. Image Tag to be used for "cncc-cmservice" micro service	**DataType:**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 **Default Value:** <Current Version>

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.initContainersImage.pullPolicy	This is an optional parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:** IfNotPresent
ingress-gateway.updateContainersImage.name	This is a mandatory parameter. Image Name to be used for "cncc-cmservice" micro service	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:** cncc/apigw-configurationupdate
ingress-gateway.updateContainersImage.tag	This is a mandatory parameter. Image Tag to be used for "cncc-cmservice" micro service	**DataType:**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 **Default Value:** <Current Version>
ingress-gateway.updateContainersImage.pullPolicy	This is an optional parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always, Never **Default Value:**IfNotPresent
ingress-gateway.service.ssl.tlsVersion	This is an optional parameter. It is the TLS version	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:**TLSv1.2
ingress-gateway.service.ssl.privateKey.k8SecretName	This is an optional parameter. Name of the privatekey secret Example: cncc-core-ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:** cncc-core-ingress-secret
ingress-gateway.service.ssl.privateKey.k8NameSpace	This is an optional parameter. Namespace of privatekey Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.service.ssl.privateKey.rsa.fileName	This is an optional parameter. rsa private key file name Example: rsa_private_key_pkcs1.pem	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** rsa_private_key_pkcs1.pem
ingress-gateway.service.ssl.privateKey.ecdsa.fileName	This is an optional parameter. ecdsa private key file name Example: ssl_ecdsa_private_key.pem	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** ssl_ecdsa_private_key.pem
ingress-gateway.service.ssl.certificate.k8SecretName	This is an optional parameter. Name of the certificate secret Example: cncc-core-ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:**cncc-core-ingress-secret
ingress-gateway.service.ssl.certificate.k8NameSpace	This is an optional parameter. Namespace of certificate Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc
ingress-gateway.service.ssl.certificate.rsa.fileName	This is an optional parameter. rsa certificate file name Example: ssl_rsa_certificate.crt	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** ssl_rsa_certificate.crt
ingress-gateway.service.ssl.certificate.ecdsa.fileName	This is an optional parameter. ecdsa certificate file name Example: ssl_ecdsa_certificate.crt	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** ssl_ecdsa_certificate.crt

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.service.ssl.caBundle.k8SecretName	This is an optional parameter. Name of the caBundle secret Example: cncc-core-ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator **Default Value:** cncc-core-ingress-secret
ingress-gateway.service.ssl.caBundle.k8NameSpace	This is an optional parameter. Namespace of caBundle Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc
ingress-gateway.service.ssl.caBundle.fileName	This is an optional parameter. rsa caBundle file name Example: caroot.cer	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** caroot.cer
ingress-gateway.service.ssl.keyStorePassword.k8SecretName	This is an optional parameter. Name of the keyStorePassword secret Example: cncc-core-ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc-core-ingress-secret
ingress-gateway.service.ssl.keyStorePassword.k8NameSpace	This is an optional parameter. Namespace of keyStorePassword Example: cncc	**DataType:**String **Range:**VValid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc
ingress-gateway.service.ssl.keyStorePassword.fileName	This is an optional parameter. File name that has password for keyStore Example: ssl_keystore.txt	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** ssl_keystore.txt
ingress-gateway.service.ssl.trustStorePassword.k8SecretName	This is an optional parameter. Name of the trustStorePassword secret Example: cncc-core-ingress-secret	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc-core-ingress-secret

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.service.ssl.trustStorePassword.k8NameSpace	This is an optional parameter. Namespace of trustStorePassword Example: cncc	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** cncc
ingress-gateway.service.ssl.trustStorePassword.fileName	This is an optional parameter. File name that has password for trustStore Example: ssl_truststore.txt	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** ssl_truststore.txt
ingress-gateway.service.ssl.initialAlgorithm	This is an optional parameter. Default values is RSA256	**DataType:**String **Range:**VValid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. A name component may not start or end with a separator. **Default Value:** RS256
ingress-gateway.service.customExtension.labels	This is an optional parameter. This can be used to add custom label(s) to ingress-gateway Service.	**DataType:**String **Range:**Custom Labels that needs to be added to ingress gateway specific Service. **Default Value:** NA
ingress-gateway.service.customExtension.annotations	This is an optional parameter. This can be used to add custom annotation(s) to ingress-gateway Service.	**DataType:**String **Range:**VCustom Labels that needs to be added to ingress gateway specific Service. **Default Value:** NA
ingress-gateway.deployment.customExtension.labels	This is an optional parameter. This can be used to add custom label(s) to ingress-gateway Deployment.	**DataType:**String **Range:**Custom Labels that needs to be added to ingress gateway specific Service. **Default Value:** NA
ingress-gateway.deployment.customExtension.annotations	This is an optional parameter. This can be used to add custom annotation(s) to ingress-gateway Deployment.	**DataType:**String **Range:**Custom Annotations that needs to be added to ingress gateway specific Deployment. **Default Value:** NA
ingress-gateway.ports.containerPort	This is an optional parameter. It is the http port of the container for the ingress-gateway.	**DataType:**Integer **Range:**0-65535 **Default Value:** 8081

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.ports.containersslPort	This is an optional parameter. It is the https port of the container for the ingress-gateway.	**DataType:**Integer **Range:**0-65535 **Default Value:** 8443
ingress-gateway.ports.actuatorPort	This is an optional parameter. It is the actuator port of the container for the ingress-gateway.	**DataType:**Integer **Range:**0-65535 **Default Value:** 9090
ingress-gateway.log.level.root	This is an optional parameter. It is the level at which user wants to see the logs. Example: WARN	**DataType:**String **Range:**WARN, DEBUG, INFO, TRACE etc. **Default Value:** INFO
ingress-gateway.log.level.ingress	This is an optional parameter. Log level for ingress logs	**DataType:**String **Range:**WARN, DEBUG, INFO, TRACE etc. **Default Value:** INFO
ingress-gateway.log.level.cncc.root	This is a mandatory parameter. Log level for cncc root logs	**DataType:**String **Range:** WARN, DEBUG, INFO, TRACE etc. **Default Value:**WARN
ingress-gateway.log.level.cncc.audit	This is a mandatory parameter. Log level for cncc audit logs	**DataType:**String **Range:** WARN, DEBUG, INFO, TRACE etc. **Default Value:** INFO
ingress-gateway.log.level.cncc.security	This is an optional parameter. Log level for cncc security logs	**DataType:**String **Range:**WARN, DEBUG, INFO, TRACE etc. **Default Value:** INFO
ingress-gateway.readinessProbe.initialDelaySeconds	This is an optional parameter. It tells the kubelet that it should wait second before performing the first probe	**DataType:**Integer **Range:**0-65535. **Default Value:** 30
ingress-gateway.readinessProbe.timeoutSeconds	This is an optional parameter. It is the number of seconds after which the probe times out	**DataType:**Integer **Range:**0-65535. **Default Value:** 3
ingress-gateway.readinessProbe.periodSeconds	This is an optional parameter. It specifies that the kubelet should perform a liveness probe every xx seconds	**DataType:**Integer **Range:**0-65535. **Default Value:** 10
ingress-gateway.readinessProbe.successThreshold	This is an optional parameter. Minimum consecutive successes for the probe to be considered successful after having failed	**DataType:**Integer **Range:**0-65535. **Default Value:** 1

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.readinessProbe.failureThreshold	This is an optional parameter. When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up	**DataType:**Integer **Range:**0-65535. **Default Value:** 3
ingress-gateway.livenessProbe.initialDelaySeconds	This is an optional parameter. It tells the kubelet that it should wait second before performing the first probe	**DataType:**Integer **Range:**0-65535. **Default Value:** 30
ingress-gateway.livenessProbe.timeoutSeconds	This is an optional parameter. It is the number of seconds after which the probe times out	**DataType:**Integer **Range:**0-65535. **Default Value:** 3
ingress-gateway.livenessProbe.periodSeconds	This is an optional parameter. It specifies that the kubelet should perform a liveness probe every xx seconds	**DataType:**Integer **Range:**0-65535. **Default Value:** 15
ingress-gateway.livenessProbe.successThreshold	This is an optional parameter. Minimum consecutive successes for the probe to be considered successful after having failed	**DataType:**Integer **Range:**0-65535. **Default Value:** 1
ingress-gateway.livenessProbe.failureThreshold	This is an optional parameter. When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up	**DataType:**Integer **Range:**0-65535. **Default Value:** 3
ingress-gateway.resources.limits.cpu	This is an optional parameter. It limits the number of CPUs to be used by the microservice.	**DataType:**Float **Range:**Valid floating point value between 0 and 1 **Default Value:**
ingress-gateway.resources.limits.initServiceCpu	This is an optional parameter. Init Container CPU Limit	**DataType:**String **Default Value:**1
ingress-gateway.resources.limits.updateServiceCpu	This is an optional parameter. Update Container CPU Limit	**DataType:**String **Default Value:** 1
ingress-gateway.resources.limits.commonHooksCpu	This is an optional parameter. common Hooks Cpu limit	**DataType:**String **Range:** **Default Value:** 1
ingress-gateway.resources.limits.memory	This is an optional parameter. It limits the memory utilization by the microservice.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi etc. **Default Value:**
ingress-gateway.resources.limits.updateServiceMemory	This is an optional parameter. Update Container Memory Limit	**DataType:**String **Default Value:** 1Gi

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.resources.limits.initServiceMemory	This is an optional parameter. Init Container Memory Limit	**DataType:**String **Range:** **Default Value:** 1Gi
ingress-gateway.resources.limits.commonHooksMemory	This is an optional parameter. common Hook Container Memory Limit	**DataType:**String **Default Value:** 1Gi
ingress-gateway.resources.requests.cpu	This is an optional parameter. It provides a given number of CPUs for the microservice.	**DataType:**Float **Range:**Valid floating point value between 0 and 1 **Default Value:**
ingress-gateway.resources.requests.initServiceCpu	This is an optional parameter. Init Container CPU Limit	**DataType:**String **Range:** **Default Value:**0.5
ingress-gateway.resources.requests.updateServiceCpu	This is an optional parameter. Update Container CPU for requests	**DataType:**String **Default Value:** 0.5
ingress-gateway.resources.requests.commonHooksCpu	This is an optional parameter. Common Hook Container CPU for requests	**DataType:**String **Default Value:** 0.5
ingress-gateway.resources.requests.memory	This is an optional parameter. It provides a given amount of memory for the microservice.	**DataType:**String **Range:**Valid Integer value followed by Mi/Gi etc.
ingress-gateway.resources.requests.updateServiceMemory	This is an optional parameter. Update Container Memory for requests	**DataType:**String **Default Value:** 0.5Gi
ingress-gateway.resources.requests.initServiceMemory	This is an optional parameter. Init Container Memory for requests	**DataType:**String **Range:** **Default Value:** 0.5Gi
ingress-gateway.resources.requests.commonHooksMemory	This is an optional parameter. Common Hook Container Memory for requests	**DataType:**String **Default Value:**0.5Gi
ingress-gateway.resources.target.averageCpuUtil	This is an optional parameter. It gives the average CPU utilization percentage.	**DataType:**String **Range:**A value in between 0-100 **Default Value:**

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.cipherSuites	Allowed CipherSuites for TLS1.2, if ingressgateway.enableIncomingHttps is true	**DataType:**List[String] **Range:** TLS_ECDHE_ ECDSA_WITH_AES_256_ GCM_SHA38 4 TLS_ECDHE_ RSA_WITH_ AES_256_GC M_SHA384 TLS_ECDHE_ RSA_WITH_ CHACHA20_ POLY1305_S HA256 TLS_DHE_RS A_WITH_AES_256_GCM_ SHA384 TLS_ECDHE_ ECDSA_WITH_AES_128_ GCM_SHA25 6 TLS_ECDHE_ RSA_WITH_ AES_128_GC M_SHA256 **Default Value:**
ingress-gateway.initssl	This is an optional parameter. To Initialize SSL related infrastructure in init/update container	**DataType:**Boolean **Range:**True or False **Default Value:** Values will be set by mcncc-core.global.httpsEnabled acncc-core.global.httpsEnabled
ingress-gateway.enableIncomingHttp	This is an optional parameter. Server Configuration for http and https support	**DataType:**Boolean **Range:**True or False **Default Value:** Values will be set by mcncc-core.global.enableIncomingHttp acncc-core.global.enableIncomingHttpq
ingress-gateway.enableIncomingHttps	This is an optional parameter. Server Configuration for http and https support	**DataType:**Boolean **Range:**True or False **Default Value:** Values will be set by mcncc-core.global.enableIncomingHttps acncc-core.global.enableIncomingHttps
ingress-gateway.needClientAuth	This is an optional parameter. This must be true if client certificate identity is required in the header x-custom-ingress-client-identity. Note: This parameter will be set to true only in case of ACNCC-Core deployment	**DataType:**Boolean **Range:**True or False **Default Value:** Values will be set by acncc-core.global.needClientAuth

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.ingressGwCertReloadEnabled	This is a mandatory parameter. If enabled, then certificates can be updated during run-time without up/restart of the application	**DataType:**Boolean **Range:**True or False **Default Value:** Values will be set by mcncc-core.global.ingressGwCertReloadEnabled acncc-core.global.ingressGwCertReloadEnabled
ingress-gateway.nodeSelector.nodeKey	This is an optional parameter. node selector key specific to chart (Note: This will be looked first and then, if not present global node key will be picked)	**DataType:**String **Range:** **Default Value:**
ingress-gateway.nodeSelector.nodeValue	This is an optional parameter. node selector value specific to chart (Note: This will be looked first and then, if not present global node value will be picked)	**DataType:**String **Range:** **Default Value:**
ingress-gateway.commonCfgClient.enabled	This is an optional parameter. If set to true Common Config Client would create tables in cncccommonconfig	**DataType:**Boolean **Range:**True or False **Default Value:**True
ingress-gateway.commonCfgServer.port	This is an optional parameter. It specifies the port number which makes cmservice visible to other services running within the same K8s cluster and the also used by common config client for db creation.	**DataType:**Integer **Default Value:** Values will be set by global.cmServiceHttpPort
ingress-gateway.dbConfig.dbHost	This is an optional parameter. It the hostname for persistence db Example: mysql.default.svc.cluster.local	**DataType:**String **Range:**Valid String **Default Value:** Values will be set by global.dbHost
ingress-gateway.dbConfig.dbPort	This is an optional parameter. It is the db port for cncc-core Example: 3306	**DataType:**Integer **Range:**0-65535. **Default Value:** Values will be set by global.dbPort

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.dbConfig.secretName	This is an optional parameter. It specifies an existing secret to be used for mysql username and password Example: secretName: &mySqlSecretNameRef cncc-db-secret	**DataType:**String **Range:**Valid String **Default Value:** Values will be set by global.secretName
ingress-gateway.dbConfig.dbName	This is an optional parameter. • It is the name of the database used for M-CNCC OR A-CNCC Core, user should create DB with unique name before deploying M-CNCC OR A-CNCC Core Example: – DB Name in case M-CNCC Core mcncccommonconfig – DB Name in case A-CNCC Core acncccommonconfig	**DataType:**String **Range:**Valid String **Default Value:** Values will be set by mcncc-core.global.dbName in case of mcncc-core acncc-core.global.dbName in case of acncc-core
ingress-gateway.dbConfig.dbUNameLiteral	This is an optional parameter. It is the key in the existing secret that stores the password Example: dbPasswordKey	**DataType:**String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters **Default Value:** NA
ingress-gateway.dbConfig.dbPwdLiteral	This is an optional parameter. It is the key in the existing secret that stores the username Example: dbUserNameKey	**DataType:**String **Range:**VValid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It may not start with a period or a dash and may contain a maximum of 128 characters **Default Value:** NA
ingress-gateway.dbHookImage.name	This is an optional parameter. Image Name to be used for "ingress-gateway db hook" micro service	**DataType:**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**Default Value:** cncc/apigw-common-config-hook

Table 4-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
ingress-gateway.dbHookImage.tag	This is an optional parameter. Image Tag to be used for "ingress-gateway db hook"	**DataType:**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 **Default Value:** <Current Version>
ingress-gateway.dbHookImage.pullPolicy	This is an optional parameter. Pull Policy decides from where to pull the image.	**DataType:**String **Range:**IfNotPresent, Always,Never **Default Value:**IfNotPresent
ingress-gateway.cncc.securitylogEnabled	This is an optional parameter. This flag is to enable and disable security logs for cncc.	**DataType:**Boolean **Range:**True or False **Default Value:**True
ingress-gateway.cncc.core.sessionTimeout	This is a mandatory parameter. It takes the timeout value for CNCC Session in seconds. Default: 1800 Minimum: 300 Maximum: 7200	**DataType:**Integer **Range:**0-65535 **Default Value:**1800

CNCC Instances Configurations

This section explains about the CNCC instances configurations.

Note:

- Instance id must be globally unique as it will be used for routing, recommendation for *id name <owner>-<instance name>*
- Type values are case sensitive, supported type values are POLICY, NRF, SCP, NSSF, SEPP,UDR-PROV, UDR-CONFIG, BSF, CS
- apiPrefix is must for type CS (OCCNE Common Services)
- Supported common services are grafana, kibana, jaeger, prometheus, alertmanager

CNC Console Supported OCCNE Common Services with expected apiPrefix format

Table 4-8 CNC Console Supported OCCNE Common Services with expected apiPrefix format

Common Service	apiPrefix format	Example
Grafana	/$OCCNE_CLUSTER/grafana	/mycne-cluster/grafana
Kibana	/$OCCNE_CLUSTER/kibana	/mycne-cluster/kibana
Jaeger	/$OCCNE_CLUSTER/jaeger	/mycne-cluster/jaeger
Prometheus	/$OCCNE_CLUSTER/prometheus	/mycne-cluster/prometheus
Alertmanager	/$OCCNE_CLUSTER/ alertmanager	/mycne-cluster/alertmanager

CNC Console supports common services only if its api prefix is globally unique like defined in above table.

For the CNCC Core Instances Configuration examples of all supported NFs, see CNC Console Instances Configuration ExamplesCNC Console Instances Configuration Examples in appendix.

CNC Console Instances Configuration Options

This section describes the parameters that are configured in the CNCC Instances configuration section in custom values.yaml file.

Attribute Name	SubType	Description	Details
global.self.cnccId	NA	This is a mandatory parameter. ID to uniquely identify the deployment. Its also called as owner or site name. Ex: cnccId: Clsuter1	**DataType:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. Id may not start with a period or a dash and may contain a maximum of 40 characters
global.mCnccIams.[]	id	This is a mandatory parameter. ID to uniquely identify the M-CNCC IAM	**DataType:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. Id may not start with a period or a dash and may contain a maximum of 40 characters
	fqdn	This is a mandatory parameter (if ip is not provided). M-CNCC IAM URI FQDN	**DataType:** String
	ip	This is a mandatory parameter (if ip is not provided). M-CNCC IAM URI IP	**DataType:** String

Attribute Name	SubType	Description	Details
	port	This is an optional parameter. M-CNCC IAM URI port	**DataType:** String **Range:** It can take value in the range: 0-65535 Default value is 80
	scheme	This is an optional parameter. M-CNCC IAM URI scheme	**DataType:** String **Range:** It can take either http or https value. By default, it is http.
global.mCnccCores.[]	id	This is a mandatory parameter (for M-CNCC Deployment). ID to uniquely identify the M-CNCC Core., usually its value will be same as global.mCnccIams.id value.	**DataType:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. Id may not start with a period or a dash and may contain a maximum of 40 characters
	fqdn	This is a mandatory parameter (if ip is not provided). M-CNCC Core URI FQDN	**DataType:** String
	ip	This is a mandatory parameter (if fqdn is not provided). M-CNCC Core URI IP	**DataType:** String
	port	This is an optional parameter. M-CNCC Core URI Port	**DataType:** String **Range:** It can take value in the range: 0-65535 Default value is 80
	scheme	This is an optional parameter. M-CNCC Core URI scheme	**DataType:** String **Range:** It can take either http or https value. By default, it is http.
	role	This is an optional parameter. It is an option to override M-CNCC site role. By default role value will be set as id value.	**DataType:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. Id may not start with a period or a dash and may contain a maximum of 40 characters

Attribute Name	SubType	Description	Details
global.aCnccs.[]	id	This is a mandatory parameter. ID to uniquely identify the A-CNCC	**DataType:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. Id may not start with a period or a dash and may contain a maximum of 40 characters
	fqdn	This is a mandatory parameter (if ip is not provided). A-CNCC URI FQDN	**DataType:** String
	ip	This is a mandatory parameter (if fqdn is not provided). A-CNCC URI IP	**DataType:** String
	port	This is an optional parameter. A-CNCC URI Port	**DataType:** String **Range:** It can take value in the range: 0-65535 Default value is 80
	scheme	This is an optional parameter. A-CNCC URI scheme	**DataType:** String **Range:** It can take either http or https value. By default, it is http.
	role	This is an optional parameter. It is an option to override A-CNCC site role. By default role value will be set as id value.	**DataType:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. Id may not start with a period or a dash and may contain a maximum of 40 characters
global.instances.[]	id	This is a mandatory parameter. ID to uniquely identify the NF Instance or OC-CNE Common Service Instance.	**DataType:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. Id may not start with a period or a dash and may contain a maximum of 80 characters
	type	This is a mandatory parameter. Type values are case sensitive, supported type values are BSF, NRF, NSSF, POLICY, SCP, SEPP, UDR-CONFIG, UDR-PROV, CS.	**DataType:** String **Range:**It can take one of these values: BSF, NRF, NSSF, POLICY, SCP, SEPP, UDR-CONFIG, UDR-PROV, CS.

Attribute Name	SubType	Description	Details
	fqdn	This is a mandatory parameter (if ip is not provided). FQDN of NF Instance or OC-CNE Common Service Instance	**DataType:** String
	ip	This is a mandatory parameter (if fqdn is not provided). IP of NF Instance or OCCNE Common Service Instance	**DataType:** String
	port	This is an optional parameter. Port of NF Instance or OCCNE Common Service Instance	**DataType:** String **Range:** It can take value in the range: 0-65535 Default value is 80
	scheme	This is an optional parameter. Scheme of NF Instance or OC-CNE Common Service Instance	**DataType:** String **Range:** It can take either http or https value. By default, it is http.
	owner	This is a mandatory parameter. Owner of NF Instance or OC-CNE Common Service Instance.	**DataType:** String **Range:** It takes the name of deployment that owns the Instance
	apiPrefix	This is an optional parameter. ApiPrefix used for routing OCCNE Common Service instance and used for routing OCCNE Common Service instance. Note: ApiPrefix is not required in case of NF instances Example : /<Cluster Prefix>/grafana	**DataType:** String

5
Validating CNCC Core Configuration

A pre install hook named Validation-hook is introduced for validating M-CNCC Core and A-CNCC Core helm configurations.

Validation hook pod will be auto deleted if validation succeeds, in case of failure pod status shows as *Error*.

In case of Validation hook errors, refer log for the following listed error codes:

Table 5-1 Validation Hook Errors

Error Code	Error Message Format	Error Scenarios	Sample Error Messages
1002	Duplicate value. Resource: <Configuration Name>, ID: <ID>, Attribute: <Attribute>. <More Info>	• All A-CNCC IDs must be unique • API prefix must be unique for all instances • Owner(Cluster) must have unique CS subtype	Duplicate value(s). Resource: aCncc, ID: [Cluster3], Attribute: id.
1003	Invalid Reference. Resource: <Configuration Name>, ID: <ID>, Attribute: <Attribute>. <More Info>	• All the Instance owners must be referenced in M-CNCC IAM IDs or A-CNCC IDs • M-CNCC IAM IDs and M-CNCC Core IDs must be same	• Invalid Reference. Resource: instance, ID: Cluster5, Attribute: Owner. Not present in mCncc ids or aCncc ids. • Invalid Reference. Resource: instance, ID: N/A, Attribute: N/A. M-Cncc Iam ids and M-Cncc Core ids do not match.

Table 5-1 (Cont.) Validation Hook Errors

Error Code	Error Message Format	Error Scenarios	Sample Error Messages
1001	Invalid value. Resource: <Configuration Name>, ID: <ID>, Attribute: <Attribute>. <More Info>	• Port should be Numeric • Scheme should be either HTTP/HTTPS • IDs should follow the alphanumeric pattern • Max Limit should be satisfied for M-CNCC IAM, A-CNCC and Instance • Max Length for Instance Id • Max Length for Self Cncc Id • CS instance must have one of these CS subtypes <grafana, kibana, jaeger, prometheus, alertmanager> • Both ip and fqdn cannot be provided. • Unsupported type • InvalidConfig • multicluster flag should be false in case of single-cluster deployment – cncc-iam, mcncc-core, acncc-core flags should be set to true • multicluster flag should be true in case of multi cluster deployment – Multi Cluster(manager only) - cncc-iam, mcncc-cncc should be set to true and acncc-core should be set to false – Multi Cluster(managing local NF's) - cncc-iam, mcncc-cncc and acncc-core should be set to true. – Multi cluster(agent only) - cncc-iam, mcncc-cncc should be set to false and acncc-core should be set to true • multicluster flag should be false in case of single-cluster deployment • multicluster flag should be true in case of multi-cluster deployment	• Invalid value. Resource: mCnccIam, ID: Cluster1, Attribute: Port. It should be numeric value. • Invalid value. Resource: instance, ID: Cluster3-instance1, Attribute: Scheme. Allowed values are: [http, https]. • Invalid value. Resource: instance, ID: Cluster1-grafana##$$%, Attribute: id. Ids should be alphanumeric with hyphen allowed as special character. • The count of mCnccIam exceeded max limit. Allowed Value:x. Actual Value: y. • Max limit exceeded. Allowed Value:x. Actual Value: y. • Invalid value. Resource: aCncc, ID: Cluster3, Attribute: N/A. Both ip and fqdn cannot be provided. • Invalid value. Resource: isMultiClusterEnabled, ID:,Attribute: False. • isMultiClusterEnabled is set as false, only single cluster configuration is allowed. Verify cncc-iam, mcncc-core, acncc-core configuration values. • Invalid value. Resource: isMultiClusterEnabled, ID:,Attribute: True. • isMultiClusterEnabled is set as true, only multi cluster configuration is allowed. Verify cncc-iam, mcncc-core, acncc-core configuration values. • Invalid value. Resource: isMultiClusterEnabled, ID:,Attribute: False. • isMultiClusterEnabled is set as false, only single cluster configuration is allowed. • Invalid value. Resource: isMultiClusterEnabled, ID:,Attribute: True. • isMultiClusterEnabled is set as true, only multi cluster configuration is allowed.

Table 5-1 (Cont.) Validation Hook Errors

Error Code	Error Message Format	Error Scenarios	Sample Error Messages
1004	Missing value. Resource: <Configuration Name>, ID: <ID>, Attribute: <Attribute>. <More Info>	• Missing apiPrefix parameter for type CS • Either of IP/FQDN should be present(Manager) • SelfCnccId should be present	• Missing value. Resource: instance, ID: Cluster4-grafana, Attribute: apiPrefix. • Missing value. Resource: instance, ID: Cluster3-PolicyInstance, Attribute: N/A. Either ip or fqdn is required.

Validation hook Microservices and Logs

This section explains about Validation-hook Microservice for M-CNCC Core and A-CNCC Core.

Validation-hook Microservice for M-CNCC Core

```
NAME                                              READY   STATUS   RESTARTS   AGE
cncc-mcore-ingress-gateway-pre-install-rfpjq      0/1 Completed   0         40s
cncc-mcore-validation-hook-zt78c                  0/1 Completed   0         24s
```

Validation hook Logs

In case of validation hook error, check logs for the error details.

```
NAME                                              READY   STATUS   RESTARTS   AGE
cncc-mcore-validation-hook-zt78c                  0/1     Error       0    24s
```

Example:

```
kubectl logs -f cncc-mcore-validation-hook-zt78c -n cncc
```

Sample Validationhook Logs

```
{
   "instant":{
      "epochSecond":1628155229,
      "nanoOfSecond":4094479
   },
   "thread":"main",
   "level":"ERROR",
   "loggerName":"com.oracle.cgbu.cne.cncc.core.Multi-
ClusterConfigValidation",
   "message":"{\"type\":null,\"title\":\"Bad
Request\",\"status\":400,\"detail\":\"[{\\\"type\\\":null,\\\"title\\\":\\
```

```
\"Validation Error\\\",\\\"status\\\":1001,\\\"detail\\\":\\\"Invalid
value. Resource: instance, ID: Cluster1-grafana, Attribute: N/A. Both
ip and fqdn cannot be provided.\\\",\\\"instance\\\":null,\\\"cause\\
\":null},{\\\"type\\\":null,\\\"title\\\":\\\"Validation Error\\\",\\
\"status\\\":1001,\\\"detail\\\":\\\"Invalid value. Resource:
instance, ID: Cluster3-policy-instance, Attribute: Port. It should be
numeric value.\\\",\\\"instance\\\":null,\\\"cause\\\":null},{\\\"type\
\\":null,\\\"title\\\":\\\"Validation Error\\\",\\\"status\\\":1004,\\
\"detail\\\":\\\"Missing value. Resource: Instance, ID: Cluster3-
grafana, Attribute: apiPrefix. \\\",\\\"instance\\\":null,\\\"cause\\
\":null}]\",\"instance\":null,\"cause\":null}",
   "endOfBatch":false,
   "loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger",
   "contextMap":{

   },
   "threadId":1,
   "threadPriority":5,
   "messageTimestamp":"2021-08-05T09:20:29.004+0000",
   "application":"cncc",
   "engineering_version":"1.8.0",
   "marketing_version":"1.0.0",
   "microservice":"cncc-mcore-validation-hook",
   "cluster":"cncc-mcore",
   "namespace":"cncc",
   "node":"master",
   "pod":"cncc-mcore-validation-hook-zt78c"
}
```


Validation-hook Microservice for A-CNCC Core

```
NAME                                          READY   STATUS
RESTARTS   AGE
cncc-acore-ingress-gateway-pre-install-rfpjq   0/1 Completed
0          40s
cncc-acore-validation-hook-zt78c               0/1 Completed
0          24s
```

Validation hook Logs

In case of validation hook error, check logs for the error details.

```
NAME                                          READY   STATUS
RESTARTS   AGE
cncc-acore-validation-hook-zt78c               0/1     Error      0
24s
```

Example:

```
kubectl logs -f cncc-acore-validation-hook-zt78c -n cncc
```

Sample Validationhook Logs


```
{
   "instant":{
      "epochSecond":1628155569,
      "nanoOfSecond":4076479
   },
   "thread":"main",
   "level":"ERROR",
   "loggerName":"com.oracle.cgbu.cne.cncc.core.Multi-
ClusterConfigValidation",
   "message":"{\"type\":null,\"title\":\"Bad
Request\",\"status\":400,\"detail\":\"[{\\\"type\\\":null,\\\"title\\\":\\
\"Validation Error\\\",\\\"status\\\":1001,\\\"detail\\\":\\\"Invalid value.
Resource: instance, ID: Cluster1-grafana, Attribute: N/A. Both ip and fqdn
cannot be provided.\\\",\\\"instance\\\":null,\\\"cause\\\":null},{\\\"type\\
\":null,\\\"title\\\":\\\"Validation Error\\\",\\\"status\\\":1001,\\
\"detail\\\":\\\"Invalid value. Resource: instance, ID: Cluster3-policy-
instance, Attribute: Port. It should be numeric value.\\\",\\\"instance\\
\":null,\\\"cause\\\":null},{\\\"type\\\":null,\\\"title\\\":\\\"Validation
Error\\\",\\\"status\\\":1004,\\\"detail\\\":\\\"Missing value. Resource:
Instance, ID: Cluster3-grafana, Attribute: apiPrefix. \\\",\\\"instance\\
\":null,\\\"cause\\\":null}]\",\"instance\":null,\"cause\":null}",
   "endOfBatch":false,
   "loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger",
   "contextMap":{

   },
   "threadId":1,
   "threadPriority":5,
   "messageTimestamp":"2021-08-05T09:20:29.004+0000",
   "application":"cncc",
   "engineering_version":"1.8.0",
   "marketing_version":"1.0.0",
   "microservice":"cncc-acore-validation-hook",
   "cluster":"cncc-acore",
   "namespace":"cncc",
   "node":"master",
   "pod":"cncc-acore-validation-hook-zt78c"
}
```

6
Accessing CNC Console

This section explains about the different ways to access CNC Console.

Accessing M-CNCC IAM

This section explains about the different ways to access M-CNC Console IAM. The user can select any of the following methods:

Format:

```
<scheme>://<cncc-iam-ingress IP/FQDN>:<cncc-iam-ingress Port>
```

1. Node-IP and NodePort
   Example:

   ```
   http://10.75.xx.xx:30085/
   ```

2. DNS Resolvable FQDN and NodePort
   Example:

   ```
   http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:30085/
   ```

3. External LB-IP and ServicePort
   Example:

   ```
   http://10.xx.xx.xx:8080/
   ```

4. DNS Resolvable FQDN and ServicePort
   Example:

   ```
   http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:8080/
   ```

Accessing M-CNCC Core

This section explains about the different ways to access M-CNCC Core. The user can select any of the following methods:

Format:

```
<scheme>://<cncc-mcore-ingress IP/FQDN>:<cncc-mcore-ingress Port>
```

1. Node-IP and NodePort

Example:

```
http://10.75.xx.xx:30075/
```

2. DNS Resolvable FQDN and NodePort

   Example:

   ```
   http://cncc-mcore-ingress-gateway.cncc.svc.cluster.local:30075/
   ```

3. External LB-IP and ServicePort

   Example:

   ```
   http://10.75.xx.xx:8080/
   ```

4. DNS Resolvable FQDN and ServicePort
5. Example:

   ```
   http://cncc-mcore-ingress-gateway.cncc.svc.cluster.local:8080/
   ```

> **Note:**
>
> Login to CNC IAM and add redirect url pointing to M-CNCC Core. CNCC cannot be accessed before CNCC IAM is configured to redirect. For more information, see CNC Console IAM Postinstallation Steps section.

> **Note:**
>
> NFs or Common Services(CS) deployed with A-CNCC Core should be accessed through M-CNCC Core, direct access is restricted.

7

Upgrading CNC Console

This section explains about the CNC Console upgrade and rollback procedures for single cluster and Multicluster deployments. M-CNCC IAM, M-CNCC Core, and A-CNCC Core can be upgraded from current version to the latest version using helm upgrade feature.

User can upgrade and Rollback CNC Console from a source release to a target release using CDCS or CLI procedures as outlined in the following table:

Upgrade /Rollback Task	References	Applicable for CDCS	Applicable for CLI
CNCC IAM DB Backup	CNCC IAM DB Backup	Yes	Yes
CNCC Upgrade	CNCC Upgrade	See Oracle Communications CD Control Server Installation and Upgrade Guide	Yes
CNCC IAM DB Rollback/Restore	CNCC IAM DB Rollback or Restore	Yes	Yes
CNCC Rollback	CNCC Rollback	See Oracle Communications CD Control Server Installation and Upgrade Guide	Yes

Caution:

It is recommended to verify the copy pasted content especially when the hyphens or any special characters are part of copied content.

Preupgrade Tasks

While upgrading an existing CNC Console deployment, the running set of containers and pods are replaced with the new set of containers and pods. However, if there is no change in the pod configuration, the running set of containers and pods are not replaced.

> **Caution:**
>
> - No configuration should be performed during upgrade.
> - Do not exit from helm upgrade command manually. After running the helm upgrade command, it takes sometime (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.

Preupgrade Steps

1. Keep current *occncc_custom_values_<version>.yaml* file as backup, that is

   ```
   occncc_custom_values_<version to be upgraded>.yaml
   ```

2. Update the new custom_values.yaml defined for target CNC Console release. See CNC Console IAM Configuration Parameters section and CNC Console Core Configuration Parameters section for more details about helm configurable parameters.

Supported Upgrade Paths

For information about supported upgrade paths, see CNC Console Deployment Modes section.

CNC Console Upgrade Sequence

The following is the CNC Console Upgrade Sequence:

Table 7-1 CNC Console Upgrade Sequence

Deployment Mode	Source Version	Target Version	Upgrade Sequence	Comments
Single Cluster	22.3.0, 22.3.1	22.3.x	**Console Upgrade** Upgrade CNCC **NF Upgrade** Upgrade Instances (NF or CNE/OSO Common Services)	• CNC Console can be upgraded first as 22.2.0 onwards NF includes menu api to support Console first upgrade functionality. • Helm upgrade CNCC using existing release name (cncc-iam or cncc).

Table 7-1 (Cont.) CNC Console Upgrade Sequence

Deployment Mode	Source Version	Target Version	Upgrade Sequence	Comments
	22.2.x	22.3.x	**Console Upgrade** 1. Upgrade CNCC IAM 2. Uninstall CNCC Core **NF Upgrade** Upgrade Instances (NF or CNE/OSO Common Services)	• CNC Console can be upgraded first as 22.2.0 onwards NF includes menu api to support Console first upgrade functionality. • Helm upgrade using cncc-iam release name. This will use existing IAM database and it will install M-CNCC IAM, M-CNCC Core and A-CNCC Core if enabled. • Helm uninstall cncc-core using existing release name (cncc-core or cncc). This will uninstall M-CNCC Core and A-CNCC Core if deployed.
	22.1.x or older	22.3.x	**NF Upgrade** Upgrade Instances (NF or CNE/OSO Common Services) **Console Upgrade** 1. Upgrade CNCC IAM 2. Uninstall CNCC Core	• NF should be upgraded first as the base NF version does not include menu api to support Console first upgrade functionality. • Helm upgrade using cncc-iam release name. This will use existing IAM database and it will install M-CNCC IAM, M-CNCC Core and A-CNCC Core if enabled. • Helm uninstall cncc-core using existing release name (cncc-core or cncc). This will uninstall M-CNCC Core and A-CNCC Core if deployed.

Table 7-1 (Cont.) CNC Console Upgrade Sequence

Deployment Mode	Source Version	Target Version	Upgrade Sequence	Comments
Multi Cluster	22.3.0, 22.3.1	22.3.x	**Console Manager Upgrade** Upgrade CNCC **Console Agent Upgrade** Upgrade A-CNCC Core **NF Upgrade** Upgrade Instances (NF or CNE/OSO Common Services)	• CNC Console can be upgraded first as 22.2.0 onwards NF includes menu api to support Console first upgrade functionality. • Manager Clusters : – Helm upgrade CNCC using existing release name(cncc-iam or cncc). • Agent Clusters: – Helm upgrade A-CNCC using existing release name (cncc-core or cncc). This will upgrade A-CNCC Core.

Table 7-1 (Cont.) CNC Console Upgrade Sequence

Deployment Mode	Source Version	Target Version	Upgrade Sequence	Comments
	22.2.x	22.3.x	**Console Manager Upgrade** 1. Upgrade CNCC IAM 2. Uninstall CNCC Core **Console Agent Upgrade** Upgrade A-CNCC Core **NF Upgrade** Upgrade Instances (NF or CNE/OSO Common Services)	• CNC Console can be upgraded first as 22.2.0 onwards NF includes menu api to support Console first upgrade functionality. • Manager Clusters : – Helm upgrade using cncc-iam release name. This will use existing IAM database and it will install M-CNCC IAM, M-CNCC Core and A-CNCC Core if enabled. – Helm uninstall cncc-core using existing release name (cncc-core or cncc). This will uninstall M-CNCC Core and A-CNCC Core if deployed. • Agent Clusters: – Helm upgrade cncc-core using existing release name (cncc-core or cncc). This will upgrade A-CNCC Core.

Table 7-1 (Cont.) CNC Console Upgrade Sequence

Deployment Mode	Source Version	Target Version	Upgrade Sequence	Comments
	22.1.x or older	22.3.x	**NF Upgrade** Upgrade Instances (NF or CNE/OSO Common Services) **Console Manager Upgrade** 1. Upgrade CNCC IAM 2. Uninstall CNCC Core **Console Agent Upgrade** Fresh install A-CNCC Core	• NF should be upgraded first as the base NF version does not include menu api to support Console first upgrade functionality. Exception here is Policy NF, Console can be upgraded first for Policy NF. • Manager Clusters : – Helm upgrade using cncc-iam release name. This will use existing IAM database and it will install M-CNCC IAM, M-CNCC Core and A-CNCC Core if enabled. – Helm uninstall cncc-core using existing release name. • Agent Clusters: – Fresh install A-CNCC Core – or if A-CNCC Core deployment already exists then helm upgrade cncc-core using existing release name (cncc-core or cncc).

CNC Console Rollback Sequence

The following is the CNC Console Rollback Sequence:

CNC Console Rollback Sequence

Table 7-2 CNC Console Rollback Sequence

Deployment Mode	Source Version	Target Version	Rollback Sequence	Comments
Single Cluster	22.3.x	22.3.0, 22.3.1	**Console Rollback** 1. Rollback CNCC **NF Rollback** 1. Rollback Instances (NF or CNE/OSO Common Services)	• CNC Console can be rolled back first as 22.2.0 onwards NF includes menu api to support Console first upgrade functionality. • Helm rollback CNCC using existing release name (cncc-iam or cncc). This will rollback M-CNCC IAM M-CNCC Core and A-CNCC Core if enabled.
	22.3.x	22.3.x	**Console Rollback** 1. Rollback CNCC 2. Fresh install CNCC Core **NF Rollback** 1. Rollback Instances (NF or CNE/OSO Common Services)	• CNC Console can be rolled back first as 22.2.0 onwards NF includes menu api to support Console first upgrade functionality. • Helm rollback CNCC using existing release name (cncc-iam or cncc). This will rollback M-CNCC IAM and uninstall M-CNCC Core and A-CNCC Core if enabled. • Fresh install previous version of CNCC Core using previous release name (cncc-core or cncc). It will install M-CNCC Core and A-CNCC Core if enabled. • Fresh installation of CNCC Core is needed for rollback as cncc-core was uninstalled during upgrade.

Table 7-2 (Cont.) CNC Console Rollback Sequence

Deployment Mode	Source Version	Target Version	Rollback Sequence	Comments
	22.3.x	22.1.x or older	**NF Rollback** 1. Rollback Instances (NF or CNE/OSO Common Services) **Console Rollback** 1. Rollback CNCC 2. Fresh install CNCC Core	• NF should be rolled back first as the base NF version does not include menu api to support Console first upgrade functionality. • Helm rollback CNCC using existing release name (cncc-iam or cncc). This will rollback CNCC IAM and uninstall M-CNCC Core and A-CNCC Core if enabled. • Fresh install previous version of CNCC Core using previous release name (cncc-core). It will install only CNCC Core component. • Fresh installation of CNCC Core is needed for rollback as cncc-core was uninstalled during upgrade.
Multi Cluster	22.3.x	22.3.0, 22.3.1	**Console Manager Rollback** 1. Rollback CNCC **Console Agent Rollback** 1. Rollback A-CNCC Core **NF Rollback** 1. Rollback Instances (NF or CNE/OSO Common Services)	• CNC Console can be rolled back first as 22.2.0 onwards NF includes menu api to support Console first upgrade functionality. • Manager Clusters : – Helm rollback CNCC using existing release name (cncc-iam or cncc). This will rollback M-CNCC IAM and uninstall M-CNCC Core and A-CNCC Core if enabled. • Agent Clusters: – Helm rollback cncc-core using existing release name (cncc-core or cncc). This will rollback A-CNCC Core.

Table 7-2 (Cont.) CNC Console Rollback Sequence

Deployment Mode	Source Version	Target Version	Rollback Sequence	Comments
	22.3.x	22.2.x	**Console Manager Rollback** 1. Rollback CNCC 2. Fresh Install CNCC Core **Console Agent Rollback** 1. Rollback A-CNCC Core **NF Rollback** 1. Rollback Instances (NF or CNE/OSO Common Services)	• CNC Console can be rolled back first as 22.2.0 onwards NF includes menu api to support Console first upgrade functionality. • Manager Clusters : – Helm rollback CNCC using existing release name (cncc-iam or cncc). This will rollback M-CNCC IAM and uninstall M-CNCC Core and A-CNCC Core if enabled. – Fresh install previous version of CNCC Core using previous release name (cncc-core or cncc). It will install M-CNCC Core and A-CNCC Core if enabled. – Fresh installation of CNCC Core is needed for rollback as cncc-core was uninstalled during upgrade. • Agent Clusters: – Helm rollback cncc-core using existing release name (cncc-core or cncc). This will rollback A-CNCC Core.

Table 7-2 (Cont.) CNC Console Rollback Sequence

Deployment Mode	Source Version	Target Version	Rollback Sequence	Comments
	22.3.x	22.1.x or older	**NF Rollback** 1. Rollback Instances (NF or CNE/OSO Common Services) **Console Manager Rollback** 1. Rollback CNCC 2. Fresh Install CNCC Core **Console Agent Rollback** 1. Rollback A-CNCC Core	• NF should be rolled back first as the base NF version does not include menu api to support Console first upgrade functionality. Exception here is Policy NF, Console can be upgraded first for Policy NF 22.1.x onwards. • Manager Clusters : – Helm rollback CNCC using existing release name (cncc-iam or cncc). This will rollback CNCC IAM and uninstall M-CNCC Core and A-CNCC Core if enabled. – Fresh install previous version of CNCC Core using previous release name (cncc-core or cncc). It will install CNCC Core. – Fresh installation of CNCC Core is needed for rollback as cncc-core was uninstalled during upgrade. • Agent Clusters: – Helm rollback cncc-core using existing release name (cncc-core or cncc). This will rollback A-CNCC Core. – Agent cluster rollback is applicable for Policy NF 22.1.x onwards for other NF its not applicable, they can uninstall agent deployment.

Parameters and Definitions during CNC Console Upgrade

Parameters and Definitions during CNCC Upgrade

Table 7-3 Parameters and Definitions during CNCC Upgrade

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

CNC Console Upgrade and Rollback Procedure

This section provides details of CNCC upgrade procedure to upgrade following CNC Console components.

- M-CNCC IAM
- M-CNCC Core
- A-CNCC Core

The following steps must be followed while performing the upgrade or rollback:

1. CNCC IAM DB Backup
2. CNCC Upgrade
3. CNCC IAM DB Rollback/Restore
4. CNCC Rollback

WARNING:

Before proceeding with CNCC IAM helm upgrade, latest CNCC database backup must be taken. See CNCC IAM DB Backup and store backup file in a location which can be easily restored.

Caution:

It is recommended to verify the copy pasted content especially when the hyphens or any special characters are part of copied content.

CNCC IAM DB Backup

This section describes the procedure to do the DB backup of CNCC IAM.

Prerequisites

The prerequisites for the backup process are:

- MySQL NDB cluster should be in a healthy state.
- Every database node of the MySQL NDB cluster should be in running state.
- In case of cnDBTier to verify the prerequisites, check mysql pod is up and running.
- In case of VM based DB Tier to verify the prerequisites, log in to MCM client on one of the SQL node of the cluster.
- Run the following command to check the node status.

```
mcm> show status -r occnendbclustera;
```

cnDBTier

Log in to the SQL node and run the following command to take the dump (backup) of the database. The user is required to enter the password.

```
kubectl exec -i -n <namespace> <sql-node> -- mysqldump --single-
transaction --no-tablespaces -h 127.0.0.1 -u <username> -p <database-
name> | gzip > <backup_filename>.sql.gz
```

Example:

```
kubectl exec -i -n occne-ndb ndbmysqld-0 -- mysqldump --single-
transaction --no-tablespaces --no-create-info -h 127.0.0.1 -u cnccuser
-p cnccdb | gzip > cnccdbBackup.sql.gz
```

CNCC Upgrade

This section describes the procedure to upgrade CNCC.

> **Note:**
>
> - CNC Console Upgrade or Rollback procedure depends on base CNC Console version, see CNC Console Upgrade Sequence section for more details.
> - Existing release name must be used for upgrade or rollback.

1. Prepare *occncc_custom_values_<version>.yaml* file for upgrade.
2. Upgrade CNC Console using existing release release name (cncc-iam or cncc) by running the following command:

   ```
   $ helm upgrade <release_name> <helm_chart> -f
   <occncc_custom_values_<version>.yaml> -n <namespace>
   ```

Example:

```
$ helm upgrade cncc ocspf-helm-repo/cncc -f
occncc_custom_values_<version>.yaml -n cncc
```

3. Check the status of upgrade by running the following command:

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

Example:

```
$ helm status cncc -n cncc
```

Uninstall CNCC Core

> **Note:**
>
> Uninstalling CNCC Core procedure is applicable only while upgrading from listed base version to the current version. See CNC Console Upgrade Sequence to check uninstall of CNCC core is needed.

Uninstall cncc core using old release name (cncc-core or cncc). This uninstalls M-CNCC Core and A-CNCC Core if deployed.

Command:

```
$ helm uninstall <cncc_core_release_name> -n <namespace>
```

Example:

```
$ helm uninstall cncc-core -n cncc
```

> **Note:**
>
> In case of a upgrade failure, for clean up procedure, see CNC Console Cleanup during Failed Upgrade.

CNCC IAM DB Rollback or Restore

This section provides details of CNCC IAM Rollback. In case of CNC Console IAM Upgrade failure, rollback CNC Console IAM DB to previous version by following the below steps.

> **Note:**
>
> The latest backup must be used for rollback.

cnDBTier

1. Log in to the deployment cluster, drop the existing database and create a new database. Restore the new database with the DB Schema file provided as part of package (*rollback-iam-schema-<version>.sql*):
   Run the following command to drop the Database and create a new Database:

   ```
   DROP DATABASE <CNCCDatabase>
   CREATE DATABASE IF NOT EXISTS <CNCC Database>;
   ```

   Example:

   ```
   To be run in the mysql pod:
   DROP DATABASE cnccdb;
   CREATE DATABASE IF NOT EXISTS cnccdb;
   ```

2. Copy the DB Schema file provided as part of package into the MYSQl pod (occncc_rollback_iam_schema_<version>.sql).
   Run the following command to copy the DB Schema file:

   ```
   kubectl cp <backup_file name>.sql <namespace>/<pod-name>:<directory
   where you want your file placed>
   ```

   Example:

   ```
   kubectl cp occncc_rollback_iam_schema_<version>.sql cndbtier1/
   ndbmysqld-0:/home/mysql
   ```

3. Run the following command to connect to the SQL node of the NDB cluster or connect to the cnDbBTier:

   ```
   $ kubectl -n <cndbtier_namespace> exec -it <cndbtier_sql_pod_name> -
   c <cndbtier_sql_container_name> -- bash
   ```

   Example:

   ```
   $ kubectl -n cndbtier  exec -it ndbmysqld-0 -c mysqlndbcluster --
   bash
   ```

4. Restore this new database with the DB Schema file provided as part of package (occncc_rollback_iam_schema_<version>.sql).
   Run the following command to Restore DB Schema:

   ```
   mysql -h 127.0.0.1 -u root -p <DB name> < <DB Schema file name>
   ```

Example:

```
mysql -h 127.0.0.1 -u root -p cnccdb
<occncc_rollback_iam_schema_<version>.sql
```

5. The DB Dump has to be rearranged sequentially not to get any foreign key constraints issue. For that, create the ENV variables and run it in a loop.
   a. Run the following command to convert the mysqldump file which was taken as a backup (sql.gz file) to a sql file to rearrange it:
   Unzipping the gz file:

      ```
      gunzip -d <<backup_filename>.sql.gz>
      ```

      Example:

      ```
      gunzip -d cnccdbBackup.sql.gz
      ```

   b. Rearrange the backup sql file in correct order by using following procedure.
      i. Run the following command to rearrange the Table Data :

      ```
      export KC_TABLES="ADMIN_EVENT_ENTITY RESOURCE_SERVER
      RESOURCE_SERVER_POLICY
                ASSOCIATED_POLICY REALM CLIENT AUTHENTICATION_FLOW
      AUTHENTICATION_EXECUTION
                AUTHENTICATOR_CONFIG AUTHENTICATOR_CONFIG_ENTRY
      BROKER_LINK CLIENT_ATTRIBUTES
                CLIENT_AUTH_FLOW_BINDINGS KEYCLOAK_ROLE
      CLIENT_INITIAL_ACCESS CLIENT_NODE_REGISTRATIONS
                CLIENT_SCOPE CLIENT_SCOPE_ATTRIBUTES CLIENT_SCOPE_CLIENT
      CLIENT_SCOPE_ROLE_MAPPING
                USER_SESSION CLIENT_SESSION CLIENT_SESSION_AUTH_STATUS
      CLIENT_SESSION_NOTE
                CLIENT_SESSION_PROT_MAPPER CLIENT_SESSION_ROLE
      CLIENT_USER_SESSION_NOTE COMPONENT
                COMPONENT_CONFIG COMPOSITE_ROLE DATABASECHANGELOG
      USER_ENTITY CREDENTIAL
                DATABASECHANGELOGLOCK DEFAULT_CLIENT_SCOPE EVENT_ENTITY
      FEDERATED_IDENTITY FEDERATED_USER
                FED_USER_ATTRIBUTE FED_USER_CONSENT
      FED_USER_CONSENT_CL_SCOPE FED_USER_CREDENTIAL
                FED_USER_GROUP_MEMBERSHIP FED_USER_REQUIRED_ACTION
      FED_USER_ROLE_MAPPING KEYCLOAK_GROUP
                GROUP_ATTRIBUTE GROUP_ROLE_MAPPING IDENTITY_PROVIDER
      IDENTITY_PROVIDER_CONFIG
                IDENTITY_PROVIDER_MAPPER IDP_MAPPER_CONFIG
      MIGRATION_MODEL OFFLINE_CLIENT_SESSION
                OFFLINE_USER_SESSION POLICY_CONFIG PROTOCOL_MAPPER
      PROTOCOL_MAPPER_CONFIG REALM_ATTRIBUTE
                REALM_DEFAULT_GROUPS REALM_LOCALIZATIONS
      REALM_ENABLED_EVENT_TYPES REALM_EVENTS_LISTENERS
                REALM_REQUIRED_CREDENTIAL REALM_SMTP_CONFIG
      REALM_SUPPORTED_LOCALES REDIRECT_URIS
                REQUIRED_ACTION_CONFIG REQUIRED_ACTION_PROVIDER
      ```

```
RESOURCE_SERVER_RESOURCE
          RESOURCE_ATTRIBUTE RESOURCE_POLICY
RESOURCE_SERVER_SCOPE RESOURCE_SCOPE
          RESOURCE_SERVER_PERM_TICKET RESOURCE_URIS
ROLE_ATTRIBUTE SCOPE_MAPPING SCOPE_POLICY
          USERNAME_LOGIN_FAILURE USER_ATTRIBUTE USER_CONSENT
USER_CONSENT_CLIENT_SCOPE
          USER_FEDERATION_PROVIDER USER_FEDERATION_CONFIG
USER_FEDERATION_MAPPER
          USER_FEDERATION_MAPPER_CONFIG
USER_GROUP_MEMBERSHIP USER_REQUIRED_ACTION USER_ROLE_MAPPING
          USER_SESSION_NOTE WEB_ORIGINS";
```

ii. Run the following command to create an ENV pointing to the sql file to be filtered:

```
export KC_BACKUP="./<Backup SQL Dump File>";
```

Example:

```
export KC_BACKUP="./cnccdbBackup.sql";
```

iii. Run the following command to rearrange the dump file to make it in sequential insertion order:

```
for i in $KC_TABLES; do grep "INSERT INTO \`$i\`"$KC_BACKUP;
done > <file name along with its location>
```

Example:

```
for i in $KC_TABLES; do grep "INSERT INTO \`$i\`"$KC_BACKUP;
done > /tmp/restore.sql
```

6. Run the following command to Copy file into the pod:

```
kubectl cp <backup_file name>.sql <namespace>/<pod-name>:<directory
where you want your file placed>
```

Example:

```
kubectl cp restore.sql cndbtier1/ndbmysqld-0:/home/mysql
```

7. Run the following command to connect to the SQL node of the NDB cluster or connect to the cnDBTier:

```
$ kubectl -n <cndbtier_namespace> exec -it <cndbtier_sql_pod_name> -
c <cndbtier_sql_container_name>-- bash
```

Example:

```
$ kubectl -n cndbtier  exec -it ndbmysqld-0 -c mysqlndbcluster -- bash
```

8. Populate the Database with data using the file that you have, after filtering the sqldump file.
9. Run the following command to restore Database Data:

```
mysql -h 127.0.0.1 -u root -p <DB name> < <backup_filename>
```

Example:

```
mysql -h 127.0.0.1 -u root -p cnccdb < restore.sql
```

10. Login to the MySQL prompt and confirm that the databases are restored.
11. Run the following command to Delete the sql files copied into the pod after the restore process is complete and successful (by logging into the SQL node):

```
rm -rf <DB Schema file name>
rm -rf <backup_filename>
```

Example:

```
rm -rf occncc_rollback_iam_schema_<version>.sql
rm -rf restore.sql
```

CNCC Rollback

This section describes the procedure to Rollback CNCC.

> **Note:**
>
> - CNC Console Upgrade and Rollback procedure depends on base Console version, see CNC Console Upgrade Sequence section for more details.
> - Existing release name must be used for upgrade or rollback.

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

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

Example:

```
$ helm history cncc -n cncc
```

2. Run the following command to rollback to the required revision :

```
$ helm history cncc -n cncc
```

Example:

```
$ helm rollback cncc 1 -n cncc
```

Fresh Install CNCC Core

> **Note:**
>
> Fresh installation of CNCC Core is applicable only while performing rollback from listed version to the base version. See CNC Console Rollback Sequence to check if fresh installation CNCC Core is needed.

Install the previous version of CNCC Core using previous release name (cncc-core or cncc). It installs M-CNCC Core and A-CNCC Core, if enabled.

Command:

```
$ helm install <release name> <helm repo> -f <custom_values_yaml> --
namespace <namespace> --version <version no.>
```

Example:

```
$ helm install cncc ocscp-helm-repo/cncc -f
occncc_custom_values_22.3.2.yaml --namespace cncc --version 22.3.2
```

8
Uninstalling CNC Console

This section explains uninstallation procedure of CNC Console and its details in MySQL.

> **Note:**
>
> kubectl commands might vary based on the platform deployment. Replace kubectl with Kubernetes environment-specific command line tool to configure Kubernetes resources through kube-api server. The instructions provided in this document are as per the Oracle Communications Cloud Native Environment (OCCNE) version of kube-api server.

> **Caution:**
>
> User, computer and applications, and character encoding settings may cause an issue when copy-pasting commands or any content from PDF. PDF reader version also affects the copy-pasting functionality. It is recommended to verify the pasted content especially when the hyphens or any special characters are part of the copied content.

Deleting CNC Console Deployment

Perform the following procedure to uninstall the CNC Console.The steps needs to be run from a server that has access to Kubectl and helm commands.

Run the following command to uninstall CNC Console:

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

Example:

```
$ helm uninstall cncc --namespace cncc
```

Cleaning CNC Console Deployment

Perform the following procedure to clean up the CNC Console deployment.

To delete the Jobs after CNC Console is uninstalled or purged:

1. Run the following command to check whether the jobs are present after uninstalling CNC Console:

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

   Example:

   ```
   $ kubectl get jobs -n cncc
   ```

2. If jobs are present, delete jobs using following command:

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

   Example:

   ```
   $ kubectl delete jobs --all -n cncc
   ```

Deleting the CNC Console MySQL details

This section explains the complete removal procedure of CNC Console MySQL database and MySQL user.

1. Log in to the machine which has permission to access the SQL nodes of NDB cluster.
2. Connect to the SQL node of NDB cluster successively.
3. Log in to the MySQL prompt using root permission or as a user with permission to drop the tables.

   For example:

   ```
   mysql -h 127.0.0.1 -uroot -p
   ```

   > **Note:**
   >
   > This command may vary from system to system, path for mysql binary, root user and root password. After running this command, user need to enter the password specific to the user mentioned in the command.

4. Run the following command to remove CNC Console databases:
   a. Run the following command to remove CNCC Console IAM database:

      ```
      $ DROP DATABASE if exists <CNCC IAM Database>;
      ```

      Example:

      ```
      $ DROP DATABASE if exists cnccdb;
      ```

b. Run the following command to remove CNCC Console Core database:

```
$ DROP DATABASE if exists <CNCC Core Common Config Database>;
```

Example:

```
$ DROP DATABASE if exists <CNCC Core Common Config Database>;
$ DROP DATABASE if exists mcncccommonconfig;
$ DROP DATABASE if exists acncccommonconfig;
```

5. Run the following command to remove the CNCC MySQL Users:

```
$ DROP USER IF EXISTS <CNCC User Name>;
```

Example:

```
$ DROP USER IF EXISTS cnccusr;
```

> **Caution:**
>
> Remove MySQL users on all the SQL nodes from all the CNCC clusters.

6. Exit from MySQL prompt and SQL node.

CNC Console Cleanup During Upgrade Failure

The CNC Console 22.2.x or earlier releases used two helm chart deployment for deploying `cncc-iam` and `cncc-core` components. From CNC Console 22.3.0 onwards, single helm chart is used for deploying all the console components.

This section discuss about the required clean up, if the helm upgrade fails due to miss configuration. This procedure needs to performed before running helm upgrade again.

Following resources require cleanup

- Roles
- RoleBindings
- ServiceAccounts
- ConfigMaps
- PodDisruptionBudgets
- Services

> **Note:**
>
> While upgrading from older CNCC version to CNCC 22.3.0, it is recommended to upgrade using the existing IAM release name ("cncc-iam"). The examples mentioned below are based on this assumption that release name is cncc-iam.
>
> For more details about the upgrade, see CNC Console Upgrade and Rollback Procedure section.

Cleanup Steps:

- **For Roles**

1. Run the following command to check for the roles created:

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

Example:

```
$ kubectl get roles -n cncc
```

2. Run the following command to delete the following roles if exists:

```
$ kubectl delete roles -n <namespace_name> <release_name>-acore-
ingressgateway-role
                                           <release_name>-mcore-
ingressgateway-role
                                           <release_name>-iam-
ingressgateway-role
                                           <release_name>-helmtest-
role
```

Example:

```
$ kubectl delete roles -n cncc cncc-iam-acore-ingressgateway-role
      cncc-iam-mcore-ingressgateway-role cncc-iam-iam-
ingressgateway-role
      cncc-iam-helmtest-role
```

- **RoleBindings**

1. Run the following command to check whether the RoleBindings are already exist:

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

Example:

```
$ kubectl get rolebindings -n cncc
```

2. Run the following command to delete the following RoleBindings if exists:

```
$ kubectl delete rolebindings -n <namespace_name> <release_name>-acore-
ingressgateway-rolebinding-v1
                                                  <release_name>-mcore-
ingressgateway-rolebinding-v1
                                                  <release_name>-iam-
ingressgateway-rolebinding-v1
                                                  <release_name>-helmtest-
rolebinding
```

Example:

```
$ kubectl delete rolebindings -n cncc cncc-iam-acore-ingressgateway-
rolebinding-v1
        cncc-iam-mcore-ingressgateway-rolebinding-v1 cncc-iam-iam-
ingressgateway-rolebinding-v1
        cncc-iam-helmtest-
rolebinding
```

- **ServiceAccounts**

1. Run the following command to check whether the ServiceAccounts are already exist:

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

Example:

```
$ kubectl get sa -n cncc
```

2. Run the following command to delete the following ServiceAccounts if exists:

```
$ kubectl delete sa -n cncc <release_name>-acore-ingress-gateway
                            <release_name>-helmtest-serviceaccount
                            <release_name>-mcore-ingress-gateway
```

Example:

```
$ kubectl delete sa -n cncc cncc-iam-acore-ingress-gateway cncc-iam-
helmtest-serviceaccount
      cncc-iam-mcore-ingress-gateway
```

- **ConfigMaps**

1. Run the following command to check whether the ConfigMaps are already exist:

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

Example:

```
$ kubectl get configmap -n cncc
```

2. Run the following command to delete the following ConfigMaps if exists:

```
$ kubectl delete configmap -n <namespace_name> <release_name>-acore-
cmservice
                                                <release_name>-acore-
ingress-gateway
                                                <release_name>-mcore-
cmservice
                                                <release_name>-mcore-
ingress-gateway
```

Example:

```
$ kubectl delete configmap -n cncc cncc-iam-acore-cmservice cncc-
iam-acore-ingress-gateway
      cncc-iam-mcore-cmservice cncc-iam-mcore-ingress-gateway
```

- **PodDisruptionBudgets**

1. Run the following command to check whether the PodDisruptionBudgets are already exist:

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

Example:

```
$ kubectl get poddisruptionbudget -n cncc
```

2. Run the following command to delete the following PodDisruptionBudgets if exists:

```
$ kubectl delete poddisruptionbudget -n <namespace_name>
<release_name>-acore-cmservice-podDisruptionBudget

<release_name>-acore-ingress-gateway-podDisruptionBudget

<release_name>-mcore-cmservice-podDisruptionBudget

<release_name>-mcore-ingress-gateway-podDisruptionBudget
```

Example:

```
$ $ kubectl delete poddisruptionbudget -n cncc cncc-iam-acore-
cmservice-podDisruptionBudget
      cncc-iam-acore-ingress-gateway-podDisruptionBudget
      cncc-iam-mcore-cmservice-podDisruptionBudget
      cncc-iam-mcore-ingress-gateway-podDisruptionBudget
```

- **Services**

1. Run the following command to check whether the Services are already exist:

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

Example:

```
$ kubectl get Services -n cncc
```

2. Run the following command to delete the following Services if exists:

```
$ kubectl delete svc -n <namespace_name> <release_name>-acore-cmservice
                                         <release_name>-acore-igw-cache
                                         <release_name>-acore-ingress-
gateway
                                         <release_name>-mcore-cmservice
                                         <release_name>-mcore-igw-cache
```

Example:

```
$ kubectl delete svc -n cncc cncc-iam-acore-cmservice cncc-iam-acore-igw-
cache
      cncc-iam-acore-ingress-gateway cncc-iam-mcore-cmservice cncc-iam-
mcore-igw-cache
      cncc-iam-iam-igw-cache
```

CNC Console Helm Test Cleanup

For the information about CNC Console Helm Test Cleanup, see Helm Test Cleanup section.

9
CNC Console IAM Postinstallation Steps

This section explains the postinstallation steps such as Configuring CNCC Redirection URL, Creating the User, and Assigning the Roles.

> **Note:**
>
> CNC Console multi cluster deployment supports cluster specific role. The user can create cluster roles in CNCC IAM and assign cluster specific role to the user similar to NF roles.
>
> Operators must ensure that the cluster role name must matches with the role name given in helm configuration.
>
> - For M-CNCC cluster role creation in M-CNCC IAM value of global.mCnccCores.id or global.mCnccCores.role name must be used
> - For A-CNCC cluster role creation in M-CNCC IAM value of global.aCnccs.id or global.aCnccs.role name must be used.
>
> > **Note:**
> >
> > Cluster role names are case sensitive.

Prerequisites

The CNC Console IAM and CNC Console Core must be deployed.

Admin must perform following tasks once CNCC IAM is deployed:

- Set the cncc redirection URL.
- Create the user and assign roles (applicable if not integrated with LDAP) .

Steps for configuring CNC Console redirection URL, create user, and assign the roles:

1. Log into CNC Console IAM Console using admin credentials provided during installation of CNCC IAM.
   Format:

   ```
   <scheme>://<cncc-iam-ingress IP/FQDN>:<cncc-iam-ingress Port>
   ```

   Node-IP and NodePort
   Example:

   ```
   http://10.75.xx.xx:30085/*
   ```

DNS Resolvable FQDN and NodePort
Example:

```
http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:30085/*
```

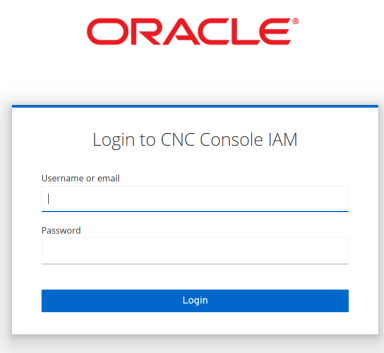
External LB-IP and ServicePort
Example:

```
http://10.75.xx.xx:8080/*
```

DNS Resolvable FQDN and ServicePort
Example:

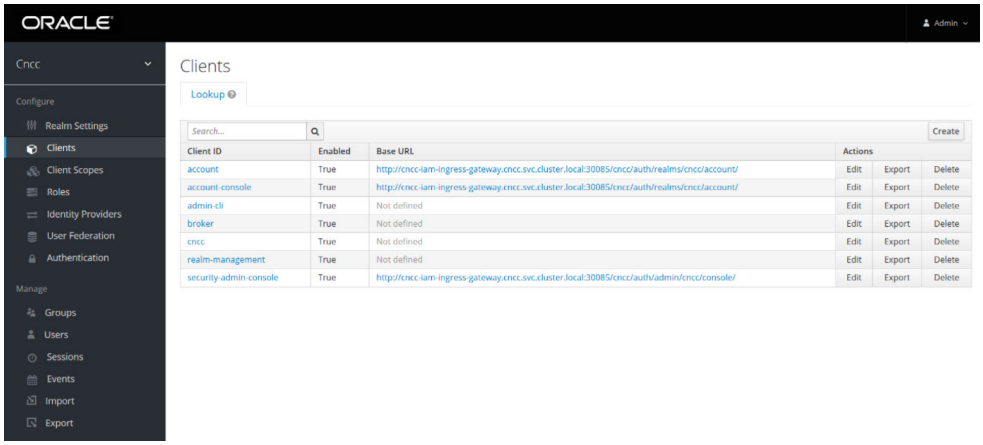
```
http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:8080/*
```

Figure 9-1 Login



1. Go to **Clients** option and click **Cncc**.

Figure 9-2 Clients tab



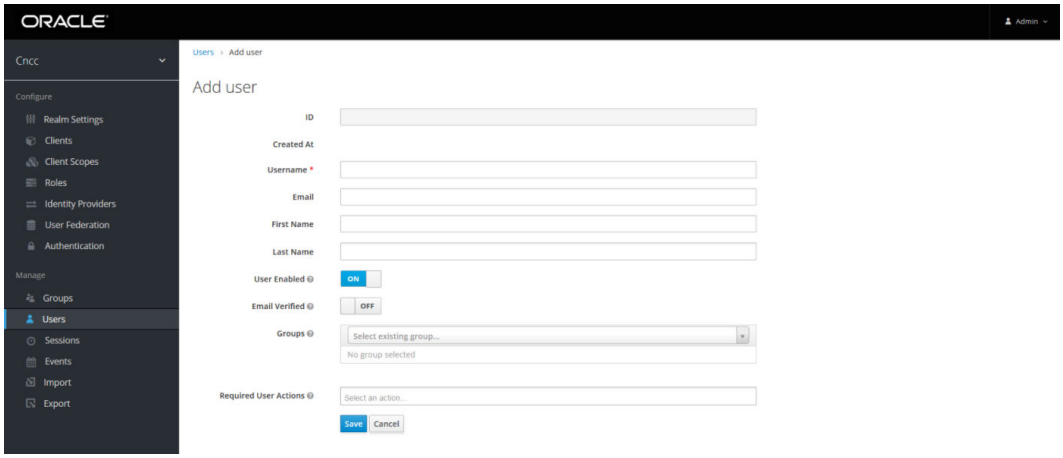
2. Enter CNCC Core Ingress URI in the **Root URIs** field and **Save.**

   ```
   <scheme>://<cncc-mcore-ingress IP/FQDN>:<cncc-mcore-ingress Port>
   ```

> **Note:**
>
> **Redirection URL** is pre-populated, only root url needs to be configured as part of Post-Installation procedure

3. Click **Manage**, click **Users,** and click **Add user** on the right pane.

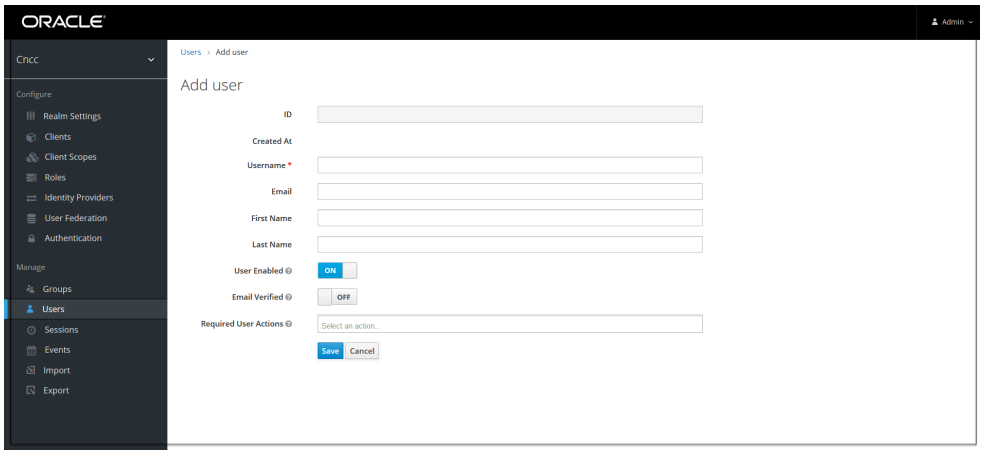
Figure 9-3 Add user



4. **Add user** screen appears. Add the user details and click **Save**.

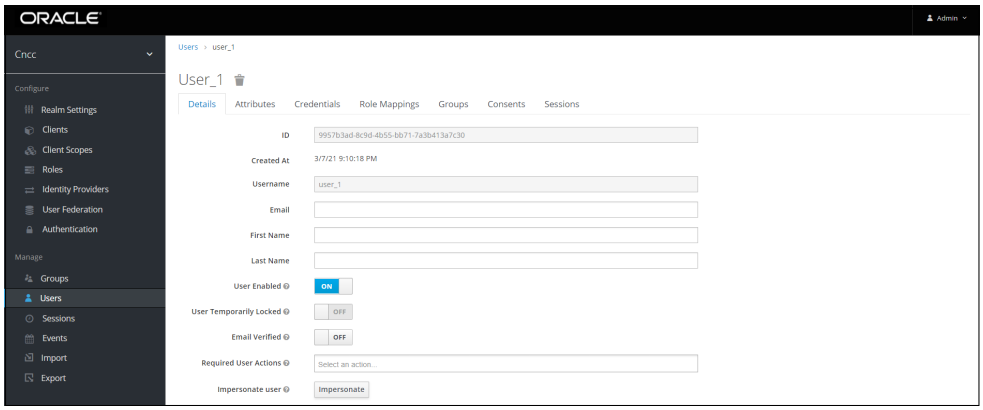
> **Note:**
>
> If you are deploying the 22.3.x package on an already existing database, the **Valid Redirect URIs** may not be pre-populated. In such a case, please use the rules provided in 22.1.x Installation Guide to manually fill the Valid Redirect URI. If you are manually filling the **Valid Redirect URI** according to the 22.1.x guide, then there is no need to fill the root URL.

Figure 9-4 Add user



5. The user has been created and the user details screen appears.

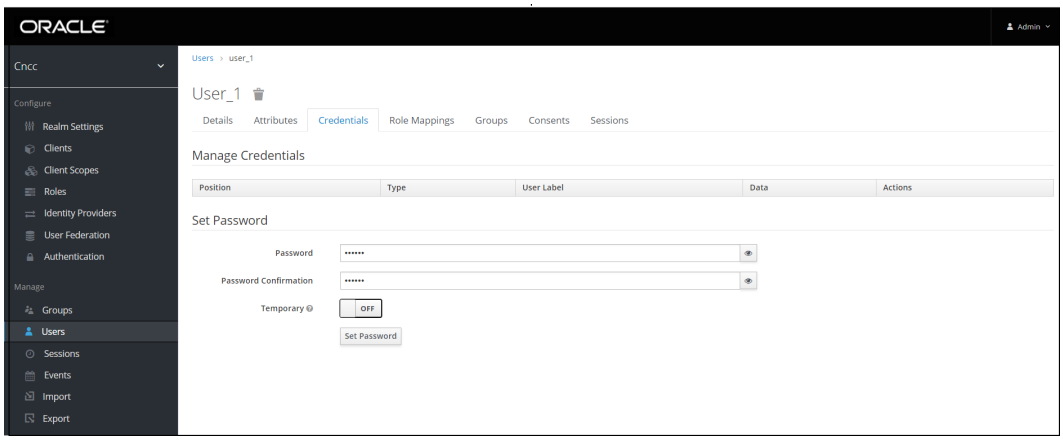
Figure 9-5 User details



6. For setting the password for the user, click **Credentials** tab and set the password for that user.

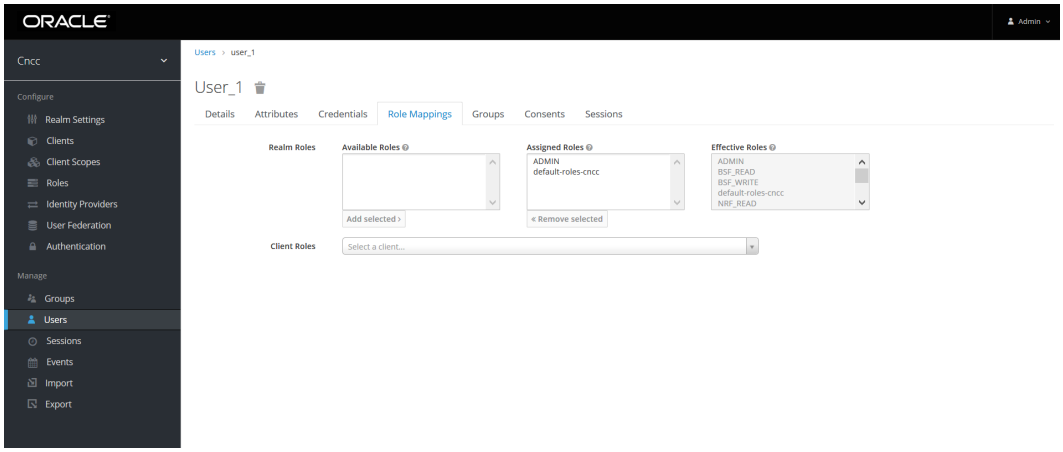
> **Note:**
>
> Setting **Temporary flag** as **ON** prompts the user to change the password when logging in to the CNCC Core GUI for the first time.

Figure 9-6 Credentials tab



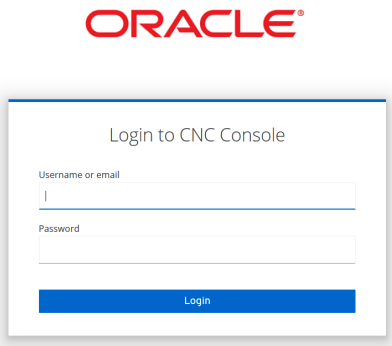
7. Navigate to the **Role Mappings** tab and assign roles to the user.

Figure 9-7 Role Mappings



8. Log into CNCC Core using the credentials of the user created earlier.

Figure 9-8 CNC Console Core login

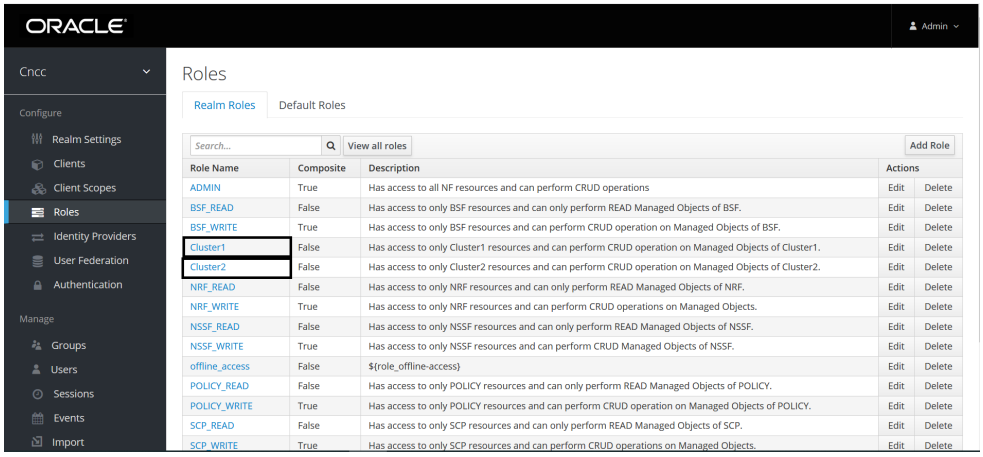


CNC Console Multi Cluster Deployment Roles

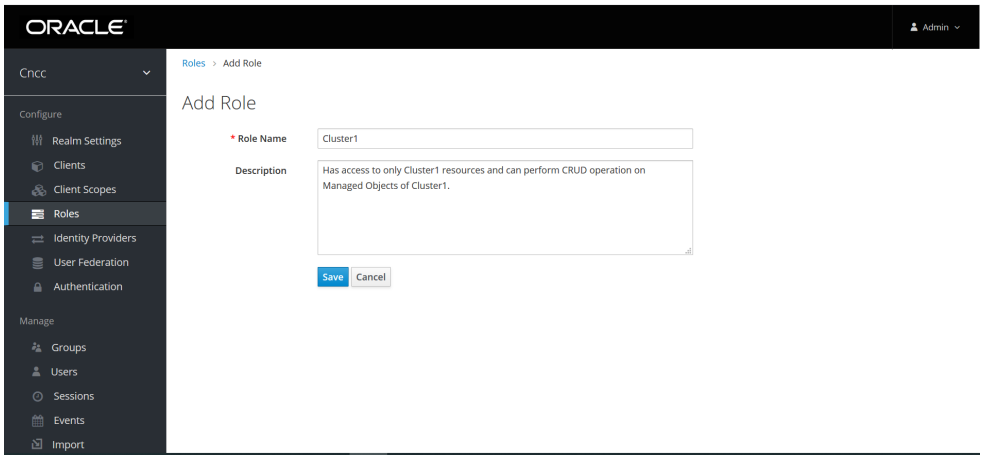
CNC Console Multi Cluster feature needs additional cluster specific roles to be created in M-CNCC IAM.

This section explains the steps to create the Roles.

1. Login to M-CNCC IAM and click the **Roles** present on the left pane. The roles defined in the realm is displayed on the right pane.



2. Click **Add Role**, the **Add Role** screen appears. Add the **Role Name** and click **Save**.



Note:

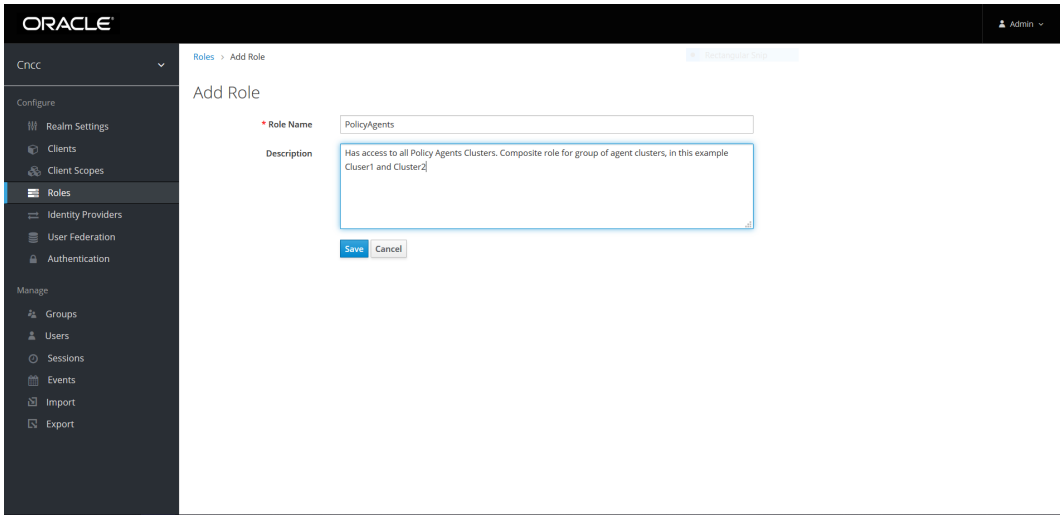
The user must ensure that the cluster role name must match with role name given in helm configuration.

- For M-CNCC cluster role creation in M-CNCC IAM, the value of `global.mCnccCores.id` or `global.mCnccCores.role name` must be used
- For A-CNCC cluster role creation in M-CNCC IAM, value of `global.aCnccs.id` or `global.aCnccs.role name` must be used.
- Cluster roles are case sensitive.

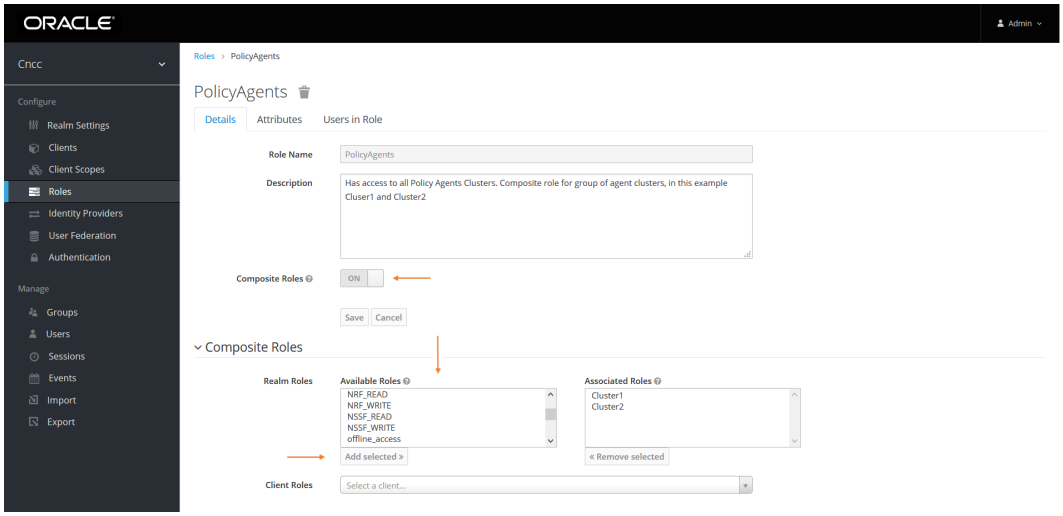
Composite Role Creation

CNCC IAM provides an option to create composite (group) the roles. This section explains the steps to create the composite roles.

1. Click **Add Role**, the **Add Role** screen appears. Add the **Role Name** and click **Save**.



2. Select the Added Role name (Example: PolicyAgents) from the **Roles** screen, the following screen appears.



3.
4. Enable the **Composite Roles** button (turn it ON).
5. This enables the **Composite Roles** section, from the **Realm Roles** select the required site roles and click **Add Selected**.

> **Note:**
>
> Here, the name "PolicyAgents" is used for composite role, that can be read as "PolicyAgentCnccs".

Note:

For more information about the Roles, see **Role Based Access Control in CNC Console**section in *Cloud Native Core Console User Guide*.

10
Performing Helm Test

Helm Test is a feature which validates CNCC Successful Installation along with readiness (Readiness probe URL configured will be checked for success) of all the pods. The pods 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.

To perform Helm test, perform the following steps:

> **Note:**
>
> The configurations mentioned in the step below must be done before running the helm install command.

1. Configure the helm test configurations which are under global sections in *occncc_custom_values_<version>.yaml* file. Refer the following configurations :
   CNCC Helm Test

```
global:
  # Helm test related configurations
  test:
    nfName: cncc
    image:
      name: occncc/nf_test
      tag: 22.3.2
      imagePullPolicy: IfNotPresent
    config:
      logLevel: WARN
      timeout: 240
```

1. Run the following command to run Helm test on the installation done:

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

   Example:

```
E.g. [root@master cncc]# helm test cncc -n cncc
Pod cncc-test pending
Pod cncc-test pending
```

```
Pod cncc-test pending
Pod cncc-test pending
Pod cncc-test running
Pod cncc-test succeeded
NAME: cncc
LAST DEPLOYED: Tue Jun  7 06:47:14 2022
NAMESPACE: cncc
STATUS: deployed
REVISION: 1
TEST SUITE:     cncc-test
Last Started:   Wed Jun  8 06:01:10 2022
Last Completed: Wed Jun  8 06:01:44 2022
Phase:          Succeeded
NOTES:
# Copyright 2020 (C), Oracle and/or its affiliates. All rights
reserved. Thank you for installing cncc. Your release is named
cncc , Release Revision: 1.
To learn more about the release, try:

  $ helm status cncc
  $ helm get cncc
```

2. Wait for the helm test to complete. Check for the output to see if the test job is successful.

Helm Test Kubernetes Resources Logging

The Helm test logging enhancement for Kubernetes resources lists the following details of Kubernetes resources:

1. Versions of Kubernetes resource available in OCCNE.
2. Preferred version for that Kubernetes resource on the OCCNE.
3. For each micro service, it list the version of Kubernetes resource used.

This information can be used in the following cases:

1. In case of CNE upgrade for customer, helm test shall list the Kubernetes resource versions used by NFs. The operator can use this information to run a pre-upgrade compatibility test to see if the Kubernetes resource versions are available on target CNE.
2. After CNE upgrade, there might be certain resources for which a newer version is available on CNE which is also supported by Console charts. If the output of helm test indicates failure of compliance for certain resource, then upgrade the Console to use the latest version of Kubernetes resource for which failure was indicated.
3. List all available versions on CNE. Console can use this detail as an input for apiVersion to be used in latest NF charts to which upgrade will be performed.

Note that this feature is tested and compatible with CNE version 1.10 and above.

To use this feature, set global.test.complianceEnable flag as true.

Separate helm test service account can be created and set at global.helmTestserviceAccountName, see Helm Test Service Account Configuration section.

> **Note:**
>
> 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.

```
# Custom service account for Helm test execution
helmTestserviceAccountName: ""

# Helm test related configurations
test:
  nfName: cncc
  image:
    name: occncc/nf_test
    tag: 22.3.2
    imagePullPolicy: IfNotPresent
  config:
    logLevel: WARN
    timeout: 240 #Beyond this duration helm test will be considered failure
  resources:
  - horizontalpodautoscalers/v1
  - deployments/v1
  - configmaps/v1
  - prometheusrules/v1
  - serviceaccounts/v1
  - poddisruptionbudgets/v1
  - roles/v1
  - statefulsets/v1
  - persistentvolumeclaims/v1
  - services/v1
  - rolebindings/v1
  complianceEnable: true
```

Run the following command to check the helm test logs:

```
helm test <releaseName> --logs -n <namespace>
```

Example:

```
helm test cncc --logs -n cncc
```

The output lists:

1. The versions of the Kubernetes resources used by Console, which helps in running pre upgrade compatibility test before CNE upgrade.
2. Compliance check for each Kubernetes resource, if compliance check is false we need to upgrade the Console charts as latest version is supported both by charts and available in new CNE.
3. Available Kubernetes resource versions on CNE.

Log Example:


```
{
  "horizontalpodautoscalers" : {
    "availableVersionOnCne" : [ "v1", "v2beta1", "v2beta2" ],
    "avaliableOnDeployment" : [ ],
    "compliant" : true,
    "prefferedVersionOnCne" : "v1",
    "maxNFVersion" : "v1"
  },
  "deployments" : {
    "availableVersionOnCne" : [ "v1" ],
    "avaliableOnDeployment" : [ ],
    "compliant" : true,
    "prefferedVersionOnCne" : "v1",
    "maxNFVersion" : "v1"
  },
  "configmaps" : {
    "availableVersionOnCne" : [ "v1" ],
    "avaliableOnDeployment" : [ ],
    "compliant" : true,
    "prefferedVersionOnCne" : "v1",
    "maxNFVersion" : "v1"
  },
  "serviceaccounts" : {
    "availableVersionOnCne" : [ "v1" ],
    "avaliableOnDeployment" : [ ],
    "compliant" : true,
    "prefferedVersionOnCne" : "v1",
    "maxNFVersion" : "v1"
  },
  "poddisruptionbudgets" : {
    "availableVersionOnCne" : [ "v1beta1" ],
    "avaliableOnDeployment" : [ ],
    "compliant" : true,
    "prefferedVersionOnCne" : "v1beta1",
    "maxNFVersion" : "v1"
  },
  "roles" : {
    "availableVersionOnCne" : [ "v1", "v1beta1" ],
    "avaliableOnDeployment" : [ ],
    "compliant" : true,
    "prefferedVersionOnCne" : "v1",
    "maxNFVersion" : "v1"
  },
  "statefulsets" : {
    "availableVersionOnCne" : [ "v1" ],
    "avaliableOnDeployment" : [ ],
    "compliant" : true,
    "prefferedVersionOnCne" : "v1",
    "maxNFVersion" : "v1"
  },
  "persistentvolumeclaims" : {
    "availableVersionOnCne" : [ "v1" ],
    "avaliableOnDeployment" : [ ],
```

```
    "compliant" : true,
    "prefferedVersionOnCne" : "v1",
    "maxNFVersion" : "v1"
  },
  "services" : {
    "availableVersionOnCne" : [ "v1" ],
    "avaliableOnDeployment" : [ ],
    "compliant" : true,
    "prefferedVersionOnCne" : "v1",
    "maxNFVersion" : "v1"
  },
  "rolebindings" : {
    "availableVersionOnCne" : [ "v1", "v1beta1" ],
    "avaliableOnDeployment" : [ ],
    "compliant" : true,
    "prefferedVersionOnCne" : "v1",
    "maxNFVersion" : "v1"
  }
}
```

Helm Test Cleanup

After running the helm test, the pod moves to completed state, to remove the pod manually, run the following command:

```
kubectl delete pod release_name -test -n <namespace_name>
```

Example:

```
kubectl delete pod cncc-test -n cncc
```

11

Configuring CNC Console to support ASM and OSO

Introduction

Oracle CNCC leverages the Istio or Envoy service mesh (Aspen Service Mesh) for all internal and external communication. The service mesh integration provides inter-NF communication and allows API gateway co-working with service mesh. The service mesh integration supports the services by deploying a special sidecar proxy in the environment to intercept all network and external communication between microservices.

> **Note:**
>
> For ASM installation and configuration, refer Official Aspen Service Mesh website for details.

Predeployment Configuration

Following are the prerequisites to install CNCC with support for ASM:

- Enabling Auto sidecar Injection for Namespace
- Set the connectivity to database (DB) service
- Set the mTLS Connection from Client Browser to ASM
- Create Service Account, Role and Role bindings with ASM annotations
  - Single Cluster Deployment
  - Multi Cluster Deployment

Enabling Auto sidecar Injection for Namespace

This section explains how to enable auto sidecar injection for namespace.

1. Run the following command to enable auto sidecar injection to automatically add the sidecars in all of the pods spawned in CNCC namespace:

   ```
   $ kubectl label ns <cncc-namespace> istio-injection=enabled
   ```

   Example:

   ```
   $ kubectl label ns cncc istio-injection=enabled
   ```

Set the Connectivity to Database (DB) Service

> **Note:**
>
> **Optional Step:**
>
> Creation of Destination-Rule and Service-Entry is applicable only if ASM is not enabled for database. That is, Istio-sidecar is not injected to the database pods.

Following are the steps for VM based DB deployment:

1. Run the following command to create a Headless service for DB connectivity in CNCC namespace:

   ```
   $ kubectl apply -f db-connectivity.yaml
   ```

   Sample db-connectivity.yaml

   ```
   # db-connectivity.yaml
   apiVersion: v1
   kind: Endpoints
   metadata:
     name: cncc-db-connectivity-service-headless
     namespace: <db-namespace>
   subsets:
   - addresses:
     - ip: <10.7x.2xx.xx> # IP Endpoint of DB service.
     ports:
     - port: 3306
       protocol: TCP
   ---
   apiVersion: v1
   kind: Service
   metadata:
     name: cncc-db-connectivity-service-headless
     namespace: <db-namespace>
   spec:
     clusterIP: None
     ports:
     - port: 3306
       protocol: TCP
       targetPort: 3306
     sessionAffinity: None
     type: ClusterIP
   ---
   apiVersion: v1
   kind: Service
   metadata:
     name: cncc-db-connectivity-service
     namespace: <cncc-namespace>
   ```

```
spec:
  externalName: cncc-db-connectivity-service-headless.<db-
namespace>.svc.<domain>
  sessionAffinity: None
  type: ExternalName
```

2. Run the following command to create ServiceEntry and DestinationRule for DB connectivity service:

```
$ kubectl apply -f db-se-dr.yaml -n <cncc-namespace>
```

Sample db-se-dr.yaml

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: cncc-db-external-se
  namespace: <cncc-namespace>
spec:
  exportTo:
  - "."
  hosts:
  - cncc-db-connectivity-service-headless.<db-namespace>.svc.<domain>
  ports:
  - number: 3306
    name: mysql
    protocol: MySQL
  location: MESH_EXTERNAL
  resolution: NONE
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: cncc-db-external-dr
  namespace: <cncc-namespace>
spec:
  exportTo:
  - "."
  host: cncc-db-connectivity-service-headless.<db-namespace>.svc.<domain>
  trafficPolicy:
    tls:
      mode: DISABLE
```

Set the mTLS Connection from Client Browser to ASM

Prerequisites
Enable certificateCustomFields in ASM values.yaml

> **Note:**
>
> Ensure that ASM is deployed with **certificateCustomFields** enabled.
>
> ```
> ASM values.yaml
>
> global:
>  certificateCustomFields: true
> ```

Using ASM self-signed CA (Default)

1. ASM creates *istio-ca* secrets (ca-certs, ca-key) in istio-namespace which contains CA public and private key.
   a. Run the following command to verify if the certificate is created :

      ```
      $ kubectl get secrets -n istio-system -o yaml istio-ca-secret
      ```

      > **Note:**
      >
      > Export the ca-*cert.pem*, ca-*key.pem* from the secret **istio-ca-secret** to your local machine where browser is installed.
      >
      > ```
      > ca-cert.pem → Istio CA public
      >
      > ca-key.pem → CA private key
      > ```

   b. Run the following commands to get ASM Istio CA certificate with base64 decoded and copy the output to a file in you local machine:

      ```
      kubectl get secret istio-ca-secret -n istio-system -o go-
      template='{{ index .data "ca-cert.pem" | base64decode}}'
      kubectl get secret istio-ca-secret -n istio-system -o go-
      template='{{ index .data "ca-key.pem" | base64decode}}'
      ```

2. Create client certificate using Openssl commands using ca-cert.pem and ca-key.pem obtained in Step1 and import it to the browser. Refer to your browser specific documentation on how to import certificate and key.
3. Update the browser configuration to trust the CA certificate (ca-cert.pem) obtained from Step 1. Refer to your browser specific documentation on how to trust the CA certificate.

Existing Organization CA

1. Create client certificate using Openssl commands using Organization CA public and private key and import it to the browser. Refer to your browser specific documentation on how to import certificate and key.
2. Update the browser configuration to trust the Organization CA. Refer to your browser specific documentation on how to trust the CA certificate.

Create Service Account, Role and Role bindings with ASM annotations

While creating service account for M-CNCC IAM, M-CNCC Core and A-CNCC Core you need to provide following ASM annotations in the given format:

```
certificate.aspenmesh.io/customFields:'{"SAN":{"DNS":["<helm-release-name>-
ingress-gateway.<cncc_namespace>.svc.<cluster_domain>"]}}'
```

Sample ASM annotations for M-CNCC-IAM, M-CNCC and A-CNCC

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    certificate.aspenmesh.io/customFields: '{ "SAN": { "DNS": [ "cncc-iam-
ingress-gateway.cncc.svc.cluster.local", "cncc-acore-ingress-
gateway.cncc.svc.cluster.local", "cncc-mcore-ingress-
gateway.cncc.svc.cluster.local"] } }'
```

Single Cluster Deployment

For Single cluster deployment, where M-CNCC IAM, M-CNCC Core and A-CNCC Core are deployed in same cluster or site can share same service account, role, and rolebinding.

Sample example for M-CNCC IAM, M-CNCC Core and A-CNCC Core| cncc-sa-role-rolebinding.yaml

```
kubectl apply -n cncc -f - <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cncc-serviceaccount
  labels:
    app.kubernetes.io/component: internal
  annotations:
    sidecar.istio.io/inject: "false"
    "certificate.aspenmesh.io/customFields": '{ "SAN": { "DNS": [ "cncc-iam-
ingress-gateway.cncc.svc.cluster.local", "cncc-acore-ingress-
gateway.cncc.svc.cluster.local", "cncc-mcore-ingress-
gateway.cncc.svc.cluster.local"] } }'
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: cncc-role
  labels:
    app.kubernetes.io/component: internal
  annotations:
    sidecar.istio.io/inject: "false"
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
```

```
  - services
  - configmaps
  - pods
  - secrets
  - endpoints
  - persistentvolumeclaims
  verbs:
  - get
  - watch
  - list
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: cncc-rolebinding
  labels:
    app.kubernetes.io/component: internal
  annotations:
    sidecar.istio.io/inject: "false"
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: cncc-role
subjects:
- kind: ServiceAccount
  name: cncc-serviceaccount
---
EOF
```

Multi Cluster Deployment
For Multi cluster deployment,

1. Where M-CNCC IAM, M-CNCC Core and A-CNCC Core are deployed in same site/cluster can share same service account, role and rolebinding.

   See the above single cluster service account example.

   > **Note:**
   >
   > In multi cluster deployment A-CNCC Core is an optional component in manager cluster, make sure to take out "cncc-ccore-ingress-gateway.cncc.svc.cluster.local" from "certificate.aspenmesh.io/customFields" in case A-CNCC Core is not deployed in manager cluster

2. Where M-CNCC IAM and M-CNCC Core which are in same cluster can still share same service account, role and rolebinding and A-CNCC deployed in different cluster, a separate service account, role and rolebinding needs to be created. **Example for M-CNCC IAM, M-CNCC Core | cncc-sa-role-rolebinding.yaml**:

   ```
   kubectl apply -n cncc -f - <<EOF
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: cncc-serviceaccount
   ```

```
  labels:
    app.kubernetes.io/component: internal
  annotations:
    sidecar.istio.io/inject: "false"
    "certificate.aspenmesh.io/customFields": '{ "SAN": { "DNS": [ "cncc-
iam-ingress-gateway.cncc.svc.cluster.local", "cncc-mcore-ingress-
gateway.cncc.svc.cluster.local"] } }'
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: cncc-role
  labels:
    app.kubernetes.io/component: internal
  annotations:
    sidecar.istio.io/inject: "false"
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
  - services
  - configmaps
  - pods
  - secrets
  - endpoints
  - persistentvolumeclaims
  verbs:
  - get
  - watch
  - list
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: cncc-rolebinding
  labels:
    app.kubernetes.io/component: internal
  annotations:
    sidecar.istio.io/inject: "false"
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: cncc-role
subjects:
- kind: ServiceAccount
  name: cncc-serviceaccount
---
EOF
```

Example for A-CNCC Core | cncc-sa-role-rolebinding.yaml

```
kubectl apply -n cncc -f - <<EOF
apiVersion: v1
kind: ServiceAccount
```

```
metadata:
  name: cncc-serviceaccount
  labels:
    app.kubernetes.io/component: internal
  annotations:
    sidecar.istio.io/inject: "false"
    "certificate.aspenmesh.io/customFields": '{ "SAN": { "DNS":
[ "cncc-acore-ingress-gateway.cncc.svc.cluster.local"] } }'
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: cncc-role
  labels:
    app.kubernetes.io/component: internal
  annotations:
    sidecar.istio.io/inject: "false"
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
  - services
  - configmaps
  - pods
  - secrets
  - endpoints
  - persistentvolumeclaims
  verbs:
  - get
  - watch
  - list
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: cncc-rolebinding
  labels:
    app.kubernetes.io/component: internal
  annotations:
    sidecar.istio.io/inject: "false"
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: cncc-role
subjects:
- kind: ServiceAccount
  name: cncc-serviceaccount
---
EOF
```

M-CNCC IAM, M-CNCC Core and A-CNCC Core configuration for ASM

This section explains about the CNCC IAM deployment configuration for ASM.

Update *occncc_custom_values_<version>.yaml* as follows:

1. Add Required Annotation:

```
global:
  # ********  Sub-Section Start: Common Global Parameters *************
  # ************************************************************************

    nonlbStatefulSets:
      labels: {}
      annotations:
        sidecar.istio.io/rewriteAppHTTPProbers: "true"

  # ********  Sub-Section End: Common Global Parameters
******************
  #
************************************************************************
```

2. Provide the service account name:

```
global:
  # *****  Sub-Section Start: Ingress Gateway Global Parameters *****
  serviceAccountName: &serviceAccountName <cncc-serviceaccount-name>
```

3. Enable Service Mesh Flag :

```
global:
 # Mandatory: This flag needs to set it "true" if Service Mesh would be
present where CNCC will be deployed
          serviceMeshCheck: true
```

4. If ASM is deployed with mTLS disabled, then set **serviceMeshHttpsEnabled** flag to *false*:

```
global:
             serviceMeshHttpsEnabled:false
```

M-CNCC IAM, M-CNCC Core, and A-CNCC Core Configuration for OSO

Add Annotation **oracle.com/cnc: "true"** under *global.customExtention.lbDeployments.annotations* section in

occncc_custom_values_<version>.*yaml* file to indicate OSO to scrape metrics from ingress pod.

```
global:
  # ****  Sub-Section Start: Common Global Parameters *****


  customExtension:
    lbDeployments:
      labels: {}
      annotations:
        oracle.com/cnc: "true"

  # ****  Sub-Section End: Common Global Parameters *******
```

12
CNC Console Debug Tools

Overview

The Debug Tools provides third party troubleshooting tools for debugging the runtime issues for both lab and production environment. Following are the available tools:

- tcpdump
- ip
- netstat
- curl
- ping
- dig

Prerequisites

This section explains the prerequisites for using debug tool.

Configurations in CNE

The following configurations must be performed in the Bastion Host.

> **Note:**
>
> These steps are needed only when you have PSP admission controller enabled in your kubernetes environment.

PodSecurityPolicy (PSP) Creation

1. Log in to the Bastion Host.

2. Create a new PSP by running the following command. The parameters *readOnlyRootFileSystem*, *allowPrivilegeEscalation*, *allowedCapabilities* are needed by debug container.

> **Note:**
>
> Other parameters are mandatory for PSP creation and can be customized as per the CNE environment. Default values are recommended.

.

PodSecurityPolicy

```
kubectl apply -f - <<EOF
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: debug-tool-psp
spec:
  readOnlyRootFilesystem: false
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - NET_ADMIN
  - NET_RAW
  fsGroup:
    ranges:
    - max: 65535
      min: 1
    rule: MustRunAs
  runAsUser:
    rule: MustRunAsNonRoot
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
  - configMap
  - downwardAPI
  - emptyDir
  - persistentVolumeClaim
  - projected
  - secret
EOF
```

Table 12-1 PodSecurityPolicy

Parameter	Description
apiVersion	APIVersion defines the versioned schema of this representation of an object.
kind	Kind is a string value representing the REST resource this object represents.

Table 12-1 (Cont.) PodSecurityPolicy

Parameter	Description
metadata	Standard object's metadata.
metadata.name	Name must be unique within a namespace.
spec	spec defines the policy enforced.
spec.readOnlyRootFilesystem	Controls whether the containers run with a read-only root filesystem (that is no writable layer).
spec.allowPrivilegeEscalation	Gates whether or not a user is allowed to set the security context of a container to allowPrivilegeEscalation=true.
spec.allowedCapabilities	Provides a list of capabilities that are allowed to be added to a container.
spec.fsGroup	Controls the supplemental group applied to some volumes. RunAsAny allows any fsGroup ID to be specified.
spec.runAsUser	Controls which user ID the containers are run with. RunAsAny allows any runAsUser to be specified.
spec.seLinux	RunAsAny allows any seLinuxOptions to be specified.
spec.supplementalGroups	Controls which group IDs containers add. RunAsAny allows any supplementalGroups to be specified.
spec.volumes	Provides a list of allowed volume types. The allowable values correspond to the volume sources that are defined when creating a volume.

Role Creation

Create a role for the PSP by executing the following commands:

Role

```
kubectl apply -f - <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: debug-tool-role
  namespace: cncc
rules:
- apiGroups:
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
  resourceNames:
  - debug-tool-psp
EOF
```

Table 12-2 Role

Parameter	Description
apiGroups	APIGroups is the name of the APIGroup that contains the resources.

Table 12-2 (Cont.) Role

Parameter	Description
apiVersion	APIVersion defines the versioned schema of this representation of an object.
kind	Kind is a string value representing the REST resource this object represents.
metadata	Standard object's metadata.
metadata.name	Name must be unique within a namespace.
metadata.namespace	Namespace defines the space within which each name must be unique.
rules	Rules holds all the PolicyRules for this Role
rules.resourceNames	ResourceNames is an optional white list of names that the rule applies to.
rules.resources	Resources is a list of resources this rule applies to.
rules.verbs	Verbs is a list of Verbs that apply to ALL the ResourceKinds and AttributeRestrictions contained in this rule.

RoleBinding Creation

Run the following command to attach the service account for your namespace with the role created for the tool PSP:

RoleBinding

```
kubectl apply -f - <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: debug-tool-rolebinding
  namespace: cncc
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: debug-tool-role
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
EOF
```

Table 12-3 RoleBinding

Parameter	Description
apiVersion	APIVersion defines the versioned schema of this representation of an object.
kind	Kind is a string value representing the REST resource this object represents.
metadata	Standard object's metadata.
metadata.name	Name must be unique within a namespace.

Table 12-3 (Cont.) RoleBinding

Parameter	Description
metadata.namespace	Namespace defines the space within which each name must be unique.
roleRef	RoleRef can reference a Role in the current namespace or a ClusterRole in the global namespace.
roleRef.apiGroup	APIGroup is the group for the resource being referenced
roleRef.kind	Kind is the type of resource being referenced
roleRef.name	Name is the name of resource being referenced
subjects	Subjects holds references to the objects the role applies to.
subjects.kind	Kind of object being referenced. Values defined by this API group are "User", "Group", and "ServiceAccount".
subjects.apiGroup	APIGroup holds the API group of the referenced subject.
subjects.name	Name of the object being referenced.

Configuration changes in CNCC Helm Charts

To enable debug tools container, make changes to *occncc_custom_values_<version>.yaml* file at Global Level by setting *extraContainers: ENABLED.*

```
global:
# Allowed Values: DISABLED, ENABLED
extraContainers: ENABLED
extraContainersTpl: |
  - command:
      - /bin/sleep
      - infinity
    image: {{ .Values.global.dockerRegistry }}/cncc/debug_tools:22.3.2
    imagePullPolicy: Always
    name: {{ printf "%s-tools-%s" (include "getprefix" .) (include
"getsuffix" .) | trunc 63 | trimPrefix "-" | trimSuffix "-"  }}
    resources:
      limits:
        ephemeral-storage: "4Gi"
        cpu: "1"
        memory: "2Gi"
      requests:
        ephemeral-storage: "2Gi"
        cpu: "0.5"
        memory: "1Gi"
    securityContext:
      allowPrivilegeEscalation: true
      capabilities:
        drop:
        - ALL
        add:
        - NET_RAW
        - NET_ADMIN
      readOnlyRootFilesystem: false
```

To enable debug tools at service level, make changes to *occncc_custom_values_<version>.yaml* file at service level by setting *extraContainers: USE_GLOBAL_VALUE*

occncc_custom_values_<version>.yaml

```
cncc-iam:
kc:

 # Allowed Values: DISABLED, ENABLED, USE_GLOBAL_VALUE
 extraContainers: USE_GLOBAL_VALUE


ingress-gateway:

 # Allowed Values: DISABLED, ENABLED, USE_GLOBAL_VALUE
 extraContainers: USE_GLOBAL_VALUE

mcncc-core:
  cmservice:
    # Allowed Values: DISABLED, ENABLED, USE_GLOBAL_VALUE
    extraContainers: USE_GLOBAL_VALUE

  ingress-gateway:
    # Allowed Values: DISABLED, ENABLED, USE_GLOBAL_VALUE
    extraContainers: USE_GLOBAL_VALUE



acncc-core:
  cmservice:
    # Allowed Values: DISABLED, ENABLED, USE_GLOBAL_VALUE
    extraContainers: USE_GLOBAL_VALUE

  ingress-gateway:
    # Allowed Values: DISABLED, ENABLED, USE_GLOBAL_VALUE
    extraContainers: USE_GLOBAL_VALUE
```

> **Note:**
>
> **User ID: `runAsUser`**
>
> Debug Tool Container comes up with the default user ID - 7000. If the operator wants to override this default value, it can be done using the **`runAsUser`** field, otherwise the field can be skipped.
>
> ```
> Default value: uid=7000(debugtool) gid=7000(debugtool)
> groups=7000(debugtool)
> ```
>
> To override runAsUser add this line under securityContext in extraContainersTpl
>
> ```
> runAsUser: <user-id>
> ```

Configuration Options

Table 12-4 Configuration Options

Parameter	Description
command	String array used for container command.
image	Docker image name
imagePullPolicy	Image Pull Policy
name	Name of the container
resources	Compute Resources required by this container
resources.limits	Limits describes the maximum amount of compute resources allowed
resources.requests	Requests describes the minimum amount of compute resources required
resources.limits.cpu	CPU limits
resources.limits.memory	Memory limits
resources.limits.ephemeral-storage	Ephemeral Storage limits
resources.requests.cpu	CPU requests
resources.requests.memory	Memory requests
resources.requests.ephemeral-storage	Ephemeral Storage requests
securityContext	Security options the container should run with.
securityContext.allowPrivilegeEscalation	AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This boolen directly controls if the no_new_privs flag will be set on the container process
secuirtyContext.readOnlyRootFilesystem	Whether this container has a read-only root filesystem. Default is false.
securityContext.capabilities	The capabilities to add or drop when running containers. Defaults to the default set of capabilities granted by the container runtime.
securityContext.capabilities.drop	Removed capabilities
secuirtyContext.capabilities.add	Added capabilities
securityContext.runAsUser	The UID to run the entrypoint of the container process.

Debug Tool Usage

Following is the procedure to run Debug Tool:

1. Run the following command to retrieve the POD details:

   ```
   $ kubectl get pods -n <k8s namespace>
   ```

   After installation the debug-tool container will get injected into the pods, sample get pod output is here :

   ```
    [root@master ~]# kubectl get po -n cncc
   NAME                                          READY   STATUS
   RESTARTS   AGE
   cncc-acore-cmservice-947cf4c89-76vq6          2/2     Running
   0          19m
   cncc-acore-ingress-gateway-764f7f5f77-qnr5p   2/2     Running
   0          19m
   cncc-iam-ingress-gateway-55987f7dc9-x5nt2     2/2     Running
   0          147m
   cncc-iam-kc-0                                 2/2     Running
   0          147m
   cncc-mcore-cmservice-947cf4c89-76vq6          2/2     Running
   0          19m
   cncc-mcore-ingress-gateway-764f7f5f77-qnr5p   2/2     Running
   0          19m
   ```

2. Run the following command to enter Debug Tools Container:

   ```
   kubectl exec -it <pod name> -c <debug_container name> -n
   <namespace> bash
   ```

   Example:

   ```
   kubectl exec -it cncc-mcore-ingress-gateway-599d858867-x9pvz -c
   tools -n cncc bash
   ```

3. Run the debug tools:

   ```
   bash -4.2$ <debug_tools>
   ```

   Example:

   ```
   bash -4.2$ tcpdump
   ```

4. Run the following command to copy output files from container to host:

   ```
   $ kubectl cp -c <debug_container name>  <pod name>:<file location
   in container> -n <namespace>  <destination location>
   ```

   Example:

   ```
   $ kubectl cp -c tools -n cncc cncc-mcore-ingress-gateway-764f7f5f77-
   qnr5p:/tmp/capture.pcap /tmp/
   ```

Steps to Enable Debug Tools Container

Debug tools container can be enabled or disabled for CNCC by using helm install or helm upgrade command.

CNC Console

Run the following command to enable or disable CNC Console IAM after updating *occncc_custom_values_<version>.yaml* file on a installed setup:

```
$ helm upgrade <release_name> -f occncc_custom_values_<version>.yaml <helm-repo> --version <helm_version>
```

Example :

```
$ helm upgrade cncc -f occncc_custom_values_<version>.yaml ocspf-helm-repo/cncc --version 22.3.2
```

13
CNC Console Microservices to Port Mapping

This section contains CNC Console microservices to port mapping details.

Table 13-1 Microservices to Port Mapping

Flow Description	Source Node	Type	Destination Node	Nature of Port	Nature of IP	Network Type	Service Port	Protocol	Notes
cncc-mcore-ingress-gateway	User interface / LoadBalancer VM	service	cncc-mcore-ingress-gateway	External	LoadBalancer	RAN / F5	8080:30075	http	
cncc-iam-ingress-gateway	User interface / LoadBalancer VM	service	cncc-iam-ingress-gateway	External	LoadBalancer	RAN / F5	8080:30085	http	
DNS query	cncc-iam-ingress-gateway		DNS server	External			53	dns	
DNS query	cncc-mcore-ingress-gateway		DNS server	External			53	dns	
Pod-Service cmservice flow	cncc-mcore-cmservice	service	cncc-mcore-cmservice	Internal	ClusterIP	Internal / K8s	8442	http	
cncc-iam-http	cncc-iam-ingress-gateway	service	cncc-iam-kc-http	Internal	ClusterIP	Internal / K8s	8285	http	
cncc-iam-http	cncc-iam-ingress-gateway	pod	cncc-iam-kc-0	Internal	ClusterIP	Internal / K8s	8080	http	
cncc-iam-http	cncc-iam-kc-0	service	cnDBTier service	Internal	ClusterIP	Internal / K8s	3306	http	
cncc-iam-http	cncc-iam-kc-0	service	Thirdparty LDAP service	External				ldap	Depends on Customer LDAP

Table 13-1 (Cont.) Microservices to Port Mapping

Flow Description	Source Node	Type	Destination Node	Nature of Port	Nature of IP	Network Type	Service Port	Protocol	Notes
cncc-iam-http	cncc-iam-kc-0	service	Thirdparty SAML service	External				saml	Depends on Customer SAML Provider
cncc-mcore-cmservice	cncc-mcore-ingress-gateway	pod	cncc-mcore-cmservice	Internal	ClusterIP	Internal / K8s	8442	http	
cncc-mcore-cmservice	cncc-mcore-ingress-gateway	service	cncc-mcore-cmservice	Internal	ClusterIP	Internal / K8s	8442	http	
nf configuration	cncc-mcore-ingress-gateway	service	nf config service	Internal	ClusterIP	Internal / K8s	8000	http	Communication between NF and CNCC, there can be more than one NF supported in CNCC.
metrics	worker node	pod	cncc-iam-ingress-gateway	Internal	Internal	Internal / K8s	9090	http	GET /actuator/health
cncc-iam-ingress-gateway	worker node	pod	cncc-iam-ingress-gateway	Internal	Internal	Internal / K8s	8081	http	
Readiness	worker node	pod	cncc-mcore-cmservice	Internal	Internal	Internal / K8s	9000	http	liveness
cncc-iam-http	worker node	pod	cncc-iam-kc-0	Internal	Internal	Internal / K8s	8080	http	User Creation, configuration
cncc-mcore-ingress-gateway	worker node	pod	cncc-mcore-ingress-gateway	Internal	Internal	Internal / K8s	8081	http	

Table 13-1 (Cont.) Microservices to Port Mapping

Flow Description	Source Node	Type	Destination Node	Nature of Port	Nature of IP	Network Type	Service Port	Protocol	Notes
metrics	worker node	pod	cncc-mcore-ingress-gateway	Internal	Internal	Internal / K8s	9090	http	GET /actuator/health

CNC Console Instances Configuration Examples

The following section lists the CNC Console instances configuration examples for single cluster and multi cluster deployment of supported NFs.

> **Note:**
>
> In the following examples, mCnccIams port is assumed as "80". The mCnccIams port configuration must be added only if port is other than "80".

BSF Instances Configuration Examples

BSF Single Cluster Deployment Instance Configuration Examples

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
  instances:
    - id : Cluster1-bsf-instance1
      type: BSF
      owner: Cluster1
      fqdn: bsf-ocbsf-config-mgmt.bsf.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
  instances:
    - id : Cluster1-bsf-instance1
      type: BSF
      owner: Cluster1
      fqdn: bsf-ocbsf-config-mgmt.bsf.svc.cluster.local
      port: 80
```

BSF Multicluster Deployment Instance Configuration Examples

(M-CNCC Core on Cluster1, A-CNCC Core on Cluster1, and A-CNCC Core on Cluster2)

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
```

```
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id : Cluster1-bsf-instance1
      type: BSF
      owner: Cluster1
      fqdn: bsf-ocbsf-config-mgmt.bsf.svc.cluster.local
      port: 80
    - id : Cluster2-bsf-instance1
      type: BSF
      owner: Cluster2
      fqdn: bsf-ocbsf-config-mgmt.bsf1.svc.cluster.local
      port: 80
    - id : Cluster2-bsf-instance2
      type: BSF
      owner: Cluster2
      fqdn: bsf-ocbsf-config-mgmt.bsf2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
```

```
# Automatic route generation for CNCC Manager Deployment
self:
  cnccId: Cluster1
mCnccIams:
  - id: Cluster1
    scheme: https
    ip: 10.xx.xx.xx
mCnccCores:
  - id: Cluster1
aCnccs:
  - id: Cluster1
    role: Cluster1
    scheme: https
    fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
    port: 443
  - id: Cluster2
    role: Cluster2
    scheme: https
    ip: 10.xx.xx.xx
    port: 443
instances:
  - id : Cluster1-bsf-instance1
    type: BSF
    owner: Cluster1
    fqdn: bsf-ocbsf-config-mgmt.bsf.svc.cluster.local
    port: 80
  - id : Cluster2-bsf-instance1
    type: BSF
    owner: Cluster2
    fqdn: bsf-ocbsf-config-mgmt.bsf1.svc.cluster.local
    port: 80
  - id : Cluster2-bsf-instance2
    type: BSF
    owner: Cluster2
    fqdn: bsf-ocbsf-config-mgmt.bsf2.svc.cluster.local
    port: 80
```

Note:

In this example, local NF and A-CNCC Core is considered to be deployed in Manager Cluster, its an optional configuration.

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
```

```
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-bsf-instance1
      type: BSF
      owner: Cluster2
      fqdn: bsf-ocbsf-config-mgmt.bsf1.svc.cluster.local
      port: 80
    - id : Cluster2-bsf-instance2
      type: BSF
      owner: Cluster2
      fqdn: bsf-ocbsf-config-mgmt.bsf2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
```

```
  - id: Cluster2
    role: Cluster2
instances:
  - id : Cluster2-bsf-instance1
    type: BSF
    owner: Cluster2
    fqdn: bsf-ocbsf-config-mgmt.bsf1.svc.cluster.local
    port: 80
  - id : Cluster2-bsf-instance2
    type: BSF
    owner: Cluster2
    fqdn: bsf-ocbsf-config-mgmt.bsf2.svc.cluster.local
    port: 80
```

NRF Instances Configuration Examples

NRF Single Cluster Deployment Instance Configuration Examples

NRF Single-Cluster Configuration

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
  instances:
    - id : Cluster1-nrf-instance1
      type: NRF
      owner: Cluster1
      fqdn: ocnrf-nrfconfiguration.nrf.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
  instances:
    - id : Cluster1-nrf-instance1
      type: NRF
      owner: Cluster1
      fqdn: ocnrf-nrfconfiguration.nrf.svc.cluster.local
      port: 80
```

NRF Multicluster Deployment Instance Configuration Examples

(M-CNCC Core on Cluster1, A-CNCC Core on Cluster1, and A-CNCC Core on Cluster2)

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
```

```
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id : Cluster1-nrf-instance1
      type: NRF
      owner: Cluster1
      fqdn: ocnrf-nrfconfiguration.nrf.svc.cluster.local
      port: 80
    - id : Cluster2-nrf-instance1
      type: NRF
      owner: Cluster2
      fqdn: ocnrf-nrfconfiguration.nrf1.svc.cluster.local
      port: 80
    - id : Cluster2-nrf-instance2
      type: NRF
      owner: Cluster2
      fqdn: ocnrf-nrfconfiguration.nrf2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

 isMultiClusterDeployment: true
 # Automatic route generation for CNCC Manager Deployment
```

```
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
    - id: Cluster2
      role: Cluster2
      scheme: https
      ip: 10.xx.xx.xx
      port: 443
  instances:
    - id : Cluster1-nrf-instance1
      type: NRF
      owner: Cluster1
      fqdn: ocnrf-nrfconfiguration.nrf.svc.cluster.local
      port: 80
    - id : Cluster2-nrf-instance1
      type: NRF
      owner: Cluster2
      fqdn: ocnrf-nrfconfiguration.nrf1.svc.cluster.local
      port: 80
    - id : Cluster2-nrf-instance2
      type: NRF
      owner: Cluster2
      fqdn: ocnrf-nrfconfiguration.nrf2.svc.cluster.local
      port: 80
```

Note:

In this example, local NF and A-CNCC Core are considered to be deployed in Manager Cluster, it is an optional configuration.

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true
```

```
isMultiClusterDeployment: true
# Automatic route generation for CNCC Agent Deployment
self:
  cnccId: Cluster2
mCnccIams:
  - id: Cluster1
    ip: 10.xx.xx.xx
mCnccCores:
  - id: Cluster1
aCnccs:
  - id: Cluster2
    role: Cluster2
instances:
  - id : Cluster2-nrf-instance1
    type: NRF
    owner: Cluster2
    fqdn: ocnrf-nrfconfiguration.nrf1.svc.cluster.local
    port: 80
  - id : Cluster2-nrf-instance2
    type: NRF
    owner: Cluster2
    fqdn: ocnrf-nrfconfiguration.nrf2.svc.cluster.local
    port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-nrf-instance1
```

```
    type: NRF
    owner: Cluster2
    fqdn: ocnrf-nrfconfiguration.nrf1.svc.cluster.local
    port: 80
  - id : Cluster2-nrf-instance2
    type: NRF
    owner: Cluster2
    fqdn: ocnrf-nrfconfiguration.nrf2.svc.cluster.local
    port: 80
```

NSSF Instances Configuration Examples

NSSF Single Cluster Deployment Instance Configuration Examples

NSSF Single Cluster Configuration

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
  instances:
    - id : Cluster1-nssf-instance1
      type: NSSF
      owner: Cluster1
      fqdn: ocnssf-nsconfig.ocnssf-nssf.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs

- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
  instances:
    - id : Cluster1-nssf-instance1
      type: NSSF
      owner: Cluster1
      fqdn: ocnssf-nsconfig.ocnssf-nssf.svc.cluster.local
      port: 80
```

NSSF Multicluster Deployment Instance Configuration Examples

(M-CNCC Core on Cluster1, A-CNCC Core on Cluster1, and A-CNCC Core on Cluster2)

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
```

```
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id : Cluster1-nssf-instance1
      type: NSSF
      owner: Cluster1
      fqdn: ocnssf-nsconfig.ocnssf-nssf1.svc.cluster.local
      port: 80
    - id : Cluster2-nssf-instance1
      type: NSSF
      owner: Cluster2
      fqdn: ocnssf-nsconfig.ocnssf-nssf2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
```

```
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
    - id: Cluster2
      role: Cluster2
      scheme: https
      ip: 10.xx.xx.xx
      port: 443
  instances:
    - id : Cluster1-nssf-instance1
      type: NSSF
      owner: Cluster1
      fqdn: ocnssf-nsconfig.ocnssf-nssf1.svc.cluster.local
      port: 80
    - id : Cluster2-nssf-instance1
      type: NSSF
      owner: Cluster2
      fqdn: ocnssf-nsconfig.ocnssf-nssf2.svc.cluster.local
      port: 80
```

> **Note:**
>
> In this example local NF and A-CNCC Core is considered to be deployed in Manager Cluster, its a optional configuration.

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-nssf-instance1
```

```
      type: NSSF
      owner: Cluster2
      fqdn: ocnssf-nsconfig.ocnssf-nssf2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-nssf-instance1
      type: NSSF
      owner: Cluster2
      fqdn: ocnssf-nsconfig.ocnssf-nssf2.svc.cluster.local
      port: 80
```

Policy Instances Configuration Examples

Policy Single Cluster Deployment Instance Configuration Examples

CNCC Configuration

```
global:

  cncc-iam:
```

```
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
  instances:
    - id : Cluster1-policy-instance1
      type: POLICY
      owner: Cluster1
      fqdn: policy-occnp-config-mgmt.policy.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
```

```
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
  instances:
    - id : Cluster1-policy-instance1
      type: POLICY
      owner: Cluster1
      fqdn: policy-occnp-config-mgmt.policy.svc.cluster.local
      port: 80
```

Policy Multicluster Deployment Instance Configuration Examples

(M-CNCC Core on Cluster1, A-CNCC Core on Cluster1 and A-CNCC Core on Cluster2)

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id : Cluster1-policy-instance1
      type: POLICY
      owner: Cluster1
      fqdn: policy-occnp-config-mgmt.policy.svc.cluster.local
      port: 80
    - id : Cluster2-policy-instance1
      type: POLICY
      owner: Cluster2
```

```
      fqdn: policy-occnp-config-mgmt.policy1.svc.cluster.local
      port: 80
    - id : Cluster2-policy-instance2
      type: POLICY
      owner: Cluster2
      fqdn: policy-occnp-config-mgmt.policy2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

 isMultiClusterDeployment: true
 # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
    - id: Cluster2
      role: Cluster2
      scheme: https
      ip: 10.xx.xx.xx
      port: 443
  instances:
    - id : Cluster1-policy-instance1
      type: POLICY
      owner: Cluster1
      fqdn: policy-occnp-config-mgmt.policy.svc.cluster.local
      port: 80
    - id : Cluster2-policy-instance1
      type: POLICY
      owner: Cluster2
      fqdn: policy-occnp-config-mgmt.policy1.svc.cluster.local
```

```
    port: 80
  - id : Cluster2-policy-instance2
    type: POLICY
    owner: Cluster2
    fqdn: policy-occnp-config-mgmt.policy2.svc.cluster.local
    port: 80
```

> **Note:**
>
> In this example, local NF and A-CNCC Core are considered to be deployed in Manager Cluster, it is an optional configuration.

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-policy-instance1
      type: POLICY
      owner: Cluster2
      fqdn: policy-occnp-config-mgmt.policy1.svc.cluster.local
      port: 80
    - id : Cluster2-policy-instance2
      type: POLICY
      owner: Cluster2
      fqdn: policy-occnp-config-mgmt.policy2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-policy-instance1
      type: POLICY
      owner: Cluster2
      fqdn: policy-occnp-config-mgmt.policy1.svc.cluster.local
      port: 80
    - id : Cluster2-policy-instance2
      type: POLICY
      owner: Cluster2
      fqdn: policy-occnp-config-mgmt.policy2.svc.cluster.local
      port: 80
```

SCP Instances Configuration Examples

SCP Single Cluster Deployment Instance Configuration Examples

SCP Single Cluster Configuration

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true
```

```
isMultiClusterDeployment: false
# Automatic route generation for CNCC Manager Deployment
self:
  cnccId: Cluster1
mCnccIams:
  - id: Cluster1
    ip: 10.xx.xx.xx
mCnccCores:
  - id: Cluster1
aCnccs:
  - id: Cluster1
    role: Cluster1
    fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
    port: 80
instances:
  - id : Cluster1-scp-instance1
    type: SCP
    owner: Cluster1
    fqdn: ocscp-scpc-configuration.scp.svc.cluster.local
    port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
  instances:
```

```
  - id : Cluster1-scp-instance1
    type: SCP
    owner: Cluster1
    fqdn: ocscp-scpc-configuration.scp.svc.cluster.local
    port: 80
```

SCP Multicluster Deployment Instance Configuration Examples

(M-CNCC Core on Cluster1, A-CNCC Core on Cluster1 and A-CNCC Core on Cluster2)

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id : Cluster1-scp-instance1
      type: SCP
      owner: Cluster1
      fqdn: ocscp-scpc-configuration.scp.svc.cluster.local
      port: 80
    - id : Cluster2-scp-instance1
      type: SCP
      owner: Cluster2
      fqdn: ocscp-scpc-configuration.scp1.svc.cluster.local
      port: 80
    - id : Cluster2-scp-instance2
      type: SCP
      owner: Cluster2
```

```
      fqdn: ocscp-scpc-configuration.scp2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
    - id: Cluster2
      role: Cluster2
      scheme: https
      ip: 10.xx.xx.xx
      port: 443
  instances:
    - id : Cluster1-scp-instance1
      type: SCP
      owner: Cluster1
      fqdn: ocscp-scpc-configuration.scp.svc.cluster.local
      port: 80
    - id : Cluster2-scp-instance1
      type: SCP
      owner: Cluster2
      fqdn: ocscp-scpc-configuration.scp1.svc.cluster.local
      port: 80
    - id : Cluster2-scp-instance2
      type: SCP
      owner: Cluster2
```

```
      fqdn: ocscp-scpc-configuration.scp2.svc.cluster.local
      port: 80
```

> **Note:**
>
> In this example local NF and A-CNCC Core are considered to be deployed in Manager Cluster, it is an optional configuration.

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-scp-instance1
      type: SCP
      owner: Cluster2
      fqdn: ocscp-scpc-configuration.scp1.svc.cluster.local
      port: 80
    - id : Cluster2-scp-instance2
      type: SCP
      owner: Cluster2
      fqdn: ocscp-scpc-configuration.scp2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-scp-instance1
      type: SCP
      owner: Cluster2
      fqdn: ocscp-scpc-configuration.scp1.svc.cluster.local
      port: 80
    - id : Cluster2-scp-instance2
      type: SCP
      owner: Cluster2
      fqdn: ocscp-scpc-configuration.scp2.svc.cluster.local
      port: 80
```

SEPP Instances Configuration Examples

SEPP Single Cluster Deployment Instance Configuration Examples

SEPP Single Cluster Configuration

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true
```

```
  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
  instances:
    - id : Cluster1-sepp-instance1
      type: SEPP
      owner: Cluster1
      fqdn: ocsepp-config-mgr-svc.sepp.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Core

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
  instances:
    - id : Cluster1-sepp-instance1
```

```
    type: SEPP
    owner: Cluster1
    fqdn: ocsepp-config-mgr-svc.sepp.svc.cluster.local
    port: 80
```

SEPP Multicluster Deployment Instance Configuration Examples

(M-CNCC Core on Cluster1, A-CNCC Core on Cluster1, and A-CNCC Core on Cluster2)

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id : Cluster1-sepp-instance1
      type: SEPP
      owner: Cluster1
      fqdn: ocsepp-config-mgr-svc.sepp.svc.cluster.local
      port: 80
    - id : Cluster2-sepp-instance1
      type: SEPP
      owner: Cluster2
      fqdn: ocsepp-config-mgr-svc.sepp1.svc.cluster.local
      port: 80
    - id : Cluster2-sepp-instance2
      type: SEPP
      owner: Cluster2
```

```
      fqdn: ocsepp-config-mgr-svc.sepp2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

 isMultiClusterDeployment: true
 # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
    - id: Cluster2
      role: Cluster2
      scheme: https
      ip: 10.xx.xx.xx
      port: 443
  instances:
    - id : Cluster1-sepp-instance1
      type: SEPP
      owner: Cluster1
      fqdn: ocsepp-config-mgr-svc.sepp.svc.cluster.local
      port: 80
    - id : Cluster2-sepp-instance1
      type: SEPP
      owner: Cluster2
      fqdn: ocsepp-config-mgr-svc.sepp1.svc.cluster.local
      port: 80
    - id : Cluster2-sepp-instance2
      type: SEPP
      owner: Cluster2
```

```
      fqdn: ocsepp-config-mgr-svc.sepp2.svc.cluster.local
      port: 80
```

> **Note:**
>
> In this example, the local NF and A-CNCC Core are considered to be deployed in Manager Cluster, it is an optional configuration.

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-sepp-instance1
      type: SEPP
      owner: Cluster2
      fqdn: ocsepp-config-mgr-svc.sepp1.svc.cluster.local
      port: 80
    - id : Cluster2-sepp-instance2
      type: SEPP
      owner: Cluster2
      fqdn: ocsepp-config-mgr-svc.sepp2.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-sepp-instance1
      type: SEPP
      owner: Cluster2
      fqdn: ocsepp-config-mgr-svc.sepp1.svc.cluster.local
      port: 80
    - id : Cluster2-sepp-instance2
      type: SEPP
      owner: Cluster2
      fqdn: ocsepp-config-mgr-svc.sepp2.svc.cluster.local
      port: 80
```

UDR Instances Configuration Examples

UDR Single Cluster Deployment Instance Configuration Examples

UDR Single Cluster Configuration

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true
```

```
isMultiClusterDeployment: false
# Automatic route generation for CNCC Manager Deployment
self:
  cnccId: Cluster1
mCnccIams:
  - id: Cluster1
    ip: 10.xx.xx.xx
mCnccCores:
  - id: Cluster1
aCnccs:
  - id: Cluster1
    role: Cluster1
    fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
    port: 80
instances:
  - id : Cluster1-udr-instance1
    type: UDR-PROV
    owner: Cluster1
    fqdn: udr1-ingressgateway-prov.udr.svc.cluster.local
    port: 80
  - id : Cluster1-udr-instance1
    type: UDR-CONFIG
    owner: Cluster1
    fqdn: udr1-nudr-config.udr.svc.cluster.local
    port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
```

```
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
  instances:
    - id : Cluster1-udr-instance1
      type: UDR-PROV
      owner: Cluster1
      fqdn: udr1-ingressgateway-prov.udr.svc.cluster.local
      port: 80
    - id : Cluster1-udr-instance1
      type: UDR-CONFIG
      owner: Cluster1
      fqdn: udr1-nudr-config.udr.svc.cluster.local
      port: 80
```

UDR Multicluster Deployment Instance Configuration Examples

(M-CNCC Core on Cluster1 , A-CNCC Core on Cluster1, and A-CNCC Core on Cluster2)

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

 isMultiClusterDeployment: true
 # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id : Cluster1-udr-instance1
      type: UDR-PROV
      owner: Cluster1
      fqdn: udr1-ingressgateway-prov.udr.svc.cluster.local
      port: 80
```

```
    - id : Cluster1-udr-instance1
      type: UDR-CONFIG
      owner: Cluster1
      fqdn: udr1-nudr-config.udr.svc.cluster.local
      port: 80
    - id : Cluster2-udr-instance1
      type: UDR-PROV
      owner: Cluster2
      fqdn: udr1-ingressgateway-prov.udr1.svc.cluster.local
      port: 80
    - id : Cluster2-udr-instance1
      type: UDR-CONFIG
      owner: Cluster2
      fqdn: udr1-nudr-config.udr1.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Manager Cluster)

```
lobal:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.cluster.local
      port: 443
    - id: Cluster2
      role: Cluster2
      scheme: https
      ip: 10.xx.xx.xx
      port: 443
  instances:
    - id : Cluster1-udr-instance1
```

```
    type: UDR-PROV
    owner: Cluster1
    fqdn: udr1-ingressgateway-prov.udr.svc.cluster.local
    port: 80
  - id : Cluster1-udr-instance1
    type: UDR-CONFIG
    owner: Cluster1
    fqdn: udr1-nudr-config.udr.svc.cluster.local
    port: 80
  - id : Cluster2-udr-instance1
    type: UDR-PROV
    owner: Cluster2
    fqdn: udr1-ingressgateway-prov.udr1.svc.cluster.local
    port: 80
  - id : Cluster2-udr-instance1
    type: UDR-CONFIG
    owner: Cluster2
    fqdn: udr1-nudr-config.udr1.svc.cluster.local
    port: 80
```

Note:

In this example, local NF and A-CNCC Core are considered to be deployed in Manager Cluster, it is an optional configuration.

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-udr-instance1
      type: UDR-PROV
      owner: Cluster2
```

```
      fqdn: udr1-ingressgateway-prov.udr1.svc.cluster.local
      port: 80
    - id : Cluster2-udr-instance1
      type: UDR-CONFIG
      owner: Cluster2
      fqdn: udr1-nudr-config.udr1.svc.cluster.local
      port: 80
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
    - id : Cluster2-udr-instance1
      type: UDR-PROV
      owner: Cluster2
      fqdn: udr1-ingressgateway-prov.udr1.svc.cluster.local
      port: 80
    - id : Cluster2-udr-instance1
      type: UDR-CONFIG
      owner: Cluster2
      fqdn: udr1-nudr-config.udr1.svc.cluster.local
      port: 80
```

Common Service Instances Configuration Examples

Common Service Single Cluster Deployment Instance Configuration Examples

Common Service Single Cluster Configuration

CNCC CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.jazz
      port: 80
  instances:
    - id: Cluster1-grafana
      type: CS
      owner: Cluster1
      fqdn: occne-kube-prom-stack-grafana.occne-infra.svc
      apiPrefix: /jazz/grafana
    - id: Cluster1-kibana
      type: CS
      owner: Cluster1
      fqdn: occne-kibana.occne-infra.svc
      apiPrefix: /jazz/kibana
    - id: Cluster1-jaeger
      type: CS
      owner: Cluster1
      fqdn: occne-tracer-jaeger-query.occne-infra.svc
      apiPrefix: /jazz/jaeger
    - id: Cluster1-prometheus
      type: CS
      owner: Cluster1
      fqdn: occne-kube-prom-stack-kube-prometheus.occne-infra.svc
```

```
      apiPrefix: /jazz/prometheus
    - id: Cluster1-alertmanager
      type: CS
      owner: Cluster1
      fqdn: occne-kube-prom-stack-kube-alertmanager.occne-infra.svc
      apiPrefix: /jazz/alertmanager
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: false
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      scheme: https
      fqdn: cncc-acore-ingress-gateway.cncc.svc.jazz
      port: 443
  instances:
    - id: Cluster1-grafana
      type: CS
      owner: Cluster1
      fqdn: occne-kube-prom-stack-grafana.occne-infra.svc
      apiPrefix: /jazz/grafana
    - id: Cluster1-kibana
      type: CS
      owner: Cluster1
      fqdn: occne-kibana.occne-infra.svc
      apiPrefix: /jazz/kibana
    - id: Cluster1-jaeger
      type: CS
      owner: Cluster1
      fqdn: occne-tracer-jaeger-query.occne-infra.svc
      apiPrefix: /jazz/jaeger
```

```
  - id: Cluster1-prometheus
    type: CS
    owner: Cluster1
    fqdn: occne-kube-prom-stack-kube-prometheus.occne-infra.svc
    apiPrefix: /jazz/prometheus
  - id: Cluster1-alertmanager
    type: CS
    owner: Cluster1
    fqdn: occne-kube-prom-stack-kube-alertmanager.occne-infra.svc
    apiPrefix: /jazz/alertmanager
```

Common Service Multicluster Deployment Instance Configuration Examples

(M-CNCC Core on Cluster1 , A-CNCC Core on Cluster1, and A-CNCC Core on Cluster2)

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
  mcncc-core:
    enabled: true
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Manager Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.jazz
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id: Cluster1-grafana
      type: CS
      owner: Cluster1
      fqdn: occne-kube-prom-stack-grafana.occne-infra.svc
      apiPrefix: /jazz/grafana
    - id: Cluster1-kibana
      type: CS
```

```
    owner: Cluster1
    fqdn: occne-kibana.occne-infra.svc
    apiPrefix: /jazz/kibana
  - id: Cluster1-jaeger
    type: CS
    owner: Cluster1
    fqdn: occne-tracer-jaeger-query.occne-infra.svc
    apiPrefix: /jazz/jaeger
  - id: Cluster1-prometheus
    type: CS
    owner: Cluster1
    fqdn: occne-kube-prom-stack-kube-prometheus.occne-infra.svc
    apiPrefix: /jazz/prometheus
  - id: Cluster1-alertmanager
    type: CS
    owner: Cluster1
    fqdn: occne-kube-prom-stack-kube-alertmanager.occne-infra.svc
    apiPrefix: /jazz/alertmanager
  - id: Cluster2-grafana
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-grafana.occne-infra.svc
    apiPrefix: /rivendell-2210/grafana
  - id: Cluster2-kibana
    type: CS
    owner: Cluster2
    fqdn: occne-kibana.occne-infra.svc
    apiPrefix: /rivendell-2210/kibana
  - id: Cluster2-jaeger
    type: CS
    owner: Cluster2
    fqdn: occne-tracer-jaeger-query.occne-infra.svc
    apiPrefix: /rivendell-2210/jaeger
  - id: Cluster2-prometheus
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-kube-prometheus.occne-infra.svc
    apiPrefix: /rivendell-2210/prometheus
  - id: Cluster2-alertmanager
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-kube-alertmanager.occne-infra.svc
    apiPrefix: /rivendell-2210/alertmanager
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCncclams and aCnccs
- Https port should be updated in mCncclams and aCnccs

CNCC Configuration (Manager Cluster)

```
global:

  cncc-iam:
    enabled: true
```

```
mcncc-core:
  enabled: true
acncc-core:
  enabled: true

isMultiClusterDeployment: true
# Automatic route generation for CNCC Manager Deployment
self:
  cnccId: Cluster1
mCnccIams:
  - id: Cluster1
    scheme: https
    ip: 10.xx.xx.xx
mCnccCores:
  - id: Cluster1
aCnccs:
  - id: Cluster1
    role: Cluster1
    scheme: https
    fqdn: cncc-acore-ingress-gateway.cncc.svc.jazz
    port: 443
  - id: Cluster2
    role: Cluster2
    scheme: https
    ip: 10.xx.xx.xx
    port: 443
instances:
  - id: Cluster1-grafana
    type: CS
    owner: Cluster1
    fqdn: occne-kube-prom-stack-grafana.occne-infra.svc
    apiPrefix: /jazz/grafana
  - id: Cluster1-kibana
    type: CS
    owner: Cluster1
    fqdn: occne-kibana.occne-infra.svc
    apiPrefix: /jazz/kibana
  - id: Cluster1-jaeger
    type: CS
    owner: Cluster1
    fqdn: occne-tracer-jaeger-query.occne-infra.svc
    apiPrefix: /jazz/jaeger
  - id: Cluster1-prometheus
    type: CS
    owner: Cluster1
    fqdn: occne-kube-prom-stack-kube-prometheus.occne-infra.svc
    apiPrefix: /jazz/prometheus
  - id: Cluster1-alertmanager
    type: CS
    owner: Cluster1
    fqdn: occne-kube-prom-stack-kube-alertmanager.occne-infra.svc
    apiPrefix: /jazz/alertmanager
  - id: Cluster2-grafana
    type: CS
    owner: Cluster2
```

```
    fqdn: occne-kube-prom-stack-grafana.occne-infra.svc
    apiPrefix: /rivendell-2210/grafana
  - id: Cluster2-kibana
    type: CS
    owner: Cluster2
    fqdn: occne-kibana.occne-infra.svc
    apiPrefix: /rivendell-2210/kibana
  - id: Cluster2-jaeger
    type: CS
    owner: Cluster2
    fqdn: occne-tracer-jaeger-query.occne-infra.svc
    apiPrefix: /rivendell-2210/jaeger
  - id: Cluster2-prometheus
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-kube-prometheus.occne-infra.svc
    apiPrefix: /rivendell-2210/prometheus
  - id: Cluster2-alertmanager
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-kube-alertmanager.occne-infra.svc
    apiPrefix: /rivendell-2210/alertmanager
```

Note:

In this example, local NF and A-CNCC Core are considered to be deployed in Manager Cluster, it is an optional configuration.

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
  instances:
```

```
  - id: Cluster2-grafana
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-grafana.occne-infra.svc
    apiPrefix: /rivendell-2210/grafana
  - id: Cluster2-kibana
    type: CS
    owner: Cluster2
    fqdn: occne-kibana.occne-infra.svc
    apiPrefix: /rivendell-2210/kibana
  - id: Cluster2-jaeger
    type: CS
    owner: Cluster2
    fqdn: occne-tracer-jaeger-query.occne-infra.svc
    apiPrefix: /rivendell-2210/jaeger
  - id: Cluster2-prometheus
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-kube-prometheus.occne-infra.svc
    apiPrefix: /rivendell-2210/prometheus
  - id: Cluster2-alertmanager
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-kube-alertmanager.occne-infra.svc
    apiPrefix: /rivendell-2210/alertmanager
```

Sample Configuration for HTTPS enabled deployment

- Scheme should be updated to "https" in mCnccIams and aCnccs
- Https port should be updated in mCnccIams and aCnccs

CNCC Configuration (Agent Cluster)

```
global:

  cncc-iam:
    enabled: false
  mcncc-core:
    enabled: false
  acncc-core:
    enabled: true

  isMultiClusterDeployment: true
  # Automatic route generation for CNCC Agent Deployment
  self:
    cnccId: Cluster2
  mCnccIams:
    - id: Cluster1
      scheme: https
      ip: 10.xx.xx.xx
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster2
      role: Cluster2
```

```
instances:
  - id: Cluster2-grafana
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-grafana.occne-infra.svc
    apiPrefix: /rivendell-2210/grafana
  - id: Cluster2-kibana
    type: CS
    owner: Cluster2
    fqdn: occne-kibana.occne-infra.svc
    apiPrefix: /rivendell-2210/kibana
  - id: Cluster2-jaeger
    type: CS
    owner: Cluster2
    fqdn: occne-tracer-jaeger-query.occne-infra.svc
    apiPrefix: /rivendell-2210/jaeger
  - id: Cluster2-prometheus
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-kube-prometheus.occne-infra.svc
    apiPrefix: /rivendell-2210/prometheus
  - id: Cluster2-alertmanager
    type: CS
    owner: Cluster2
    fqdn: occne-kube-prom-stack-kube-alertmanager.occne-infra.svc
    apiPrefix: /rivendell-2210/alertmanager
```