Oracle® Communications

Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide

Release 23.4.4
F87107-05
December 2024

Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide, Release 23.4.4

F87107-05

Contents

5 Upgrading CNC Console

6 Rolling Back CNC Console

7 Uninstalling CNC Console

8 Fault Recovery

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 terminologies used in the document:

Table Acronyms

Acronym	Definition
AD	Active Directory
ASM	Aspen Service Mesh
BSF	Oracle Communications Cloud Native Core, Binding Support Function
cnDBTier	Oracle Communications Cloud Native Core, cnDBTier
CNC Console	Oracle Communications Cloud Native Configuration Console
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
CS	Common Service
CRUD Operations	CREATE, READ, UPDATE, DELETE
OCNADD	Oracle Communications Network Analytics Data Director
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 CNE 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 Core	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. A-CNCC has no IAM component. A-CNCC is also known as A-CNCC Core or aCncc Core.
M-CNCC Kubernetes cluster	Kubernetes cluster hosting M-CNCC
mTLS	Mutual Transport Layer Security
OCNWDAF	Oracle Communications Networks Data Analytics Function

Table (Cont.) Acronyms

Acronym	Definition
Instance	NF or CNE common service managed by either M-CNCC Core or A-CNCC Core.
Site	Kubernetes Cluster
CS	CNE 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	Oracle Communications Cloud Native Core, Network Repository Function
OCNF	Oracle Communications Network Function
OSDC	Oracle Software Delivery Cloud
OSO	Oracle Communications Operations Services Overlay
PROVGW	Provisioning Gateway
REST API	Representational State Transfer Application Programming Interface
SCP	Oracle Communications Cloud Native Core, Service Communication Proxy
SAML	Security Assertion Markup Language
SBA	Service Based Architecture
SEPP	Oracle Communications Cloud Native Core, Security Edge Protection Proxy
TLS	Transport Layer Security
UDR	Oracle Communications Cloud Native Core, Unified Data Repository
UE	User Equipment
URI	Subscriber Location Function
SSO	Single Sign On

What's New in This Guide

This section introduces the documentation updates for Release 23.4.x

Release 23.4.4 - F87107-05, December 2024

- Updated the release number to 23.4.4 in the entire document.
- Updated the Supported Upgrade Paths.
- Updated the Supported Rollback Paths.
- Updated the notes in the CNCC IAM DB Rollback or Restore section to reflect the correct rollback version of the schema file and the prerequisites to perform the the pre rollback tasks in one go.

Release 23.4.3 - F87107-04, October 2024

- Updated the release number to 23.4.3 in the entire document.
- Updated the Compatibility Matrix section to provide information on CNC Console compatibility with latest NF versions.
- Added the `cncc-iam.global.iamSettingEnabled` parameter to the CNC Console IAM Configuration Parameters section.
- Added the Configuring M-CNCC IAM to enable additional settings to the Configuring A-CNCC Core mTLS section.

Release 23.4.2 - F87107-03, July 2024

- Updated the release number to 23.4.2 in the entire document.
- Updated the Compatibility Matrix section to provide information on CNC Console compatibility with latest NF versions.

Release 23.4.1 - F87107-02, April 2024

Added Support for IPv4 or IPV6 Configuration for CNC Console with details on single and dual stack deployment configuration in the appendix.

Release 23.4.0 - F87107-01, December 2023

- The following sections have been updated to support cnDBTier read operations:
  - Added DBTIER to the Type subtype of the `global.instances.[]` parameter in the Instance Configuration Options table in the CNC Console Instances Configuration Options section.
  - Added DBTIER to the list of supported values in the CNC Console Instances Configurations section.
  - Updated the error scenarios for error code 1001 in the CNC Console Validation-hook Error Codes table.
  - Added the NF Instance Configuration With cnDBTier Menu Enabled section with cnDBTier configuration examples.
  - Added the CNC Console Instances Configuration With cnDBTier Menu Enabled section with cnDBTier configuration examples.
  - Added a note in the Supported Upgrade Paths to highlight that cnDBTier supports APIs only from 23.3.x onwards.
- The following sections have been updated to support OCCM:

  - Added OCCM to the Type subtype of the `global.instances.[]` parameter in the Instance Configuration Options table in the CNC Console Instances Configuration Options section.
  - Added OCCM to the list of supported values in the CNC Console Instances Configurations section.
  - Added the OCCM Instance Configuration Examples section with OCCM configuration examples.
- Updated the cnDBTier Profile section with updated cnDBTier resources.
- Added the following parameters in the Core Backend Parameters section:
  - cmservice.deployment.startupProbe.initialDelaySeconds
  - cmservice.deployment.startupProbe.periodSeconds
  - cmservice.deployment.startupProbe.timeoutSeconds
  - cmservice.deployment.startupProbe.successThreshold
  - cmservice.deployment.startupProbe.failureThreshold
- Added the following parameter in the IAM Backend Parameters section:
  - cncc-iam.kc.preferIpv6Stack.enabled
- Added a note in the Installing CNC Console Package section with details on pod security context and container security context.
- Added a note in the CNCC IAM DB Rollback or Restore with details on using the `<occncc_rollback_iam_schema_<version>.sql>` file from the CNC Console version to which you are performing the rollback.

1 Introduction

This guide describes how to install or upgrade Oracle Communications Cloud Native Configuration Console (CNC Console) in a cloud native environment. It also includes information on performing fault recovery for CNC Console.

Note

- This guide covers the installation instructions when Podman is the container platform with Helm as the Packaging Manager. For any other container platform, the operator must use the commands based on to their deployed container runtime environment.
- `kubectl` commands can 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 CNE 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.

1.1 Overview

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 CNE common services. CNCC Core module includes CNC Console and its integration with other Cloud Native Core network functions. The CNC Console provides user interface that can be used to configure parameters for the following CNC network functions:
  - Oracle Communications Cloud Native Core, Binding Support Function (BSF)
  - Oracle Communications Cloud Native Core, Service Communication Proxy (SCP)
  - Oracle Communications Cloud Native Core, Network Repository Function (NRF)
  - Oracle Communications Cloud Native Core, Converged Policy (Policy)
  - Oracle Communications Cloud Native Core, Security Edge Protection Proxy (SEPP)
  - Oracle Communications Cloud Native Core, Unified Data Repository (UDR)
  - CNE Common Services
  - Data Director (DD)
  - Oracle Communications Network Data Analytics Function (NWDAF)

  - Provisioning Gateway (PROVGW)
- 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 multisite Cloud Native Core (CNC) installations. For more information about CDCS, see *OOracle Communications CD Control Server User Guide.*.

CDCS is a centralized server that automates CNC Console deployment processes such as install, 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 postinstallation tasks. You must perform CNC Console installation tasks in the same sequence as outlined in the following table:

Task	Sub tasks	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
Preinstallation Tasks		Preinstallation Tasks	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	Installing CNC Console		
Global Configurations		Global Configurations	Yes	Yes
	CNC Console Configuration for Service Account	CNC Console Configuration for Service Account	Yes	Yes

Task	Sub tasks	Reference	Applicable for CDCS	Applicable for CLI
	Configuring ASM and OSO in M-CNCC IAM	\Configuring CNC Console to support Aspen Service Mesh	Yes	Yes
CNC Console IAM Pre-deployment Configuration		CNC Console IAM Pre-deployment 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
	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 Pre-deployment Configuration		M-CNCC Core Pre-deployment 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 Pre-deployment Configuration		A-CNCC Core Pre-deployment Configuration	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

Task	Sub tasks	Reference	Applicable for CDCS	Applicable for CLI
Uninstalling CNC Console		Uninstalling CNC Console	Yes	Yes
CNC Console IAM PostInstallation Steps		CNC Console IAM PostInstallation 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

1.2 CNC Console Compatibility Matrix

This section lists the versions of added or updated components in release 23.4.x. To know the list of all the supported versions, see Oracle Communications Cloud Native Core Release Notes.

Release 23.4.3

The following table lists the versions of added or updated components in release 23.4.3:

Table 1-1 Compatibility Matrix

Network Functions	Compatible Versions
BSF	23.4.x
NRF	23.4.x
NSSF	23.4.x
Policy	23.4.x
SCP	23.4.x
SEPP	23.4.x
UDR	23.4.x

CNC Console is compatible with the following components:

Table 1-2 Compatibility Matrix

Components	Compatible Versions
OCNADD	23.4.x
CNE	23.4.x, 23.3.x, 23.2.x
cnDBTier	23.4.x, 23.3.x, 23.2.x
CDCS	23.4.x, 23.3.x, 23.2.x
OSO	23.4.x, 23.3.x, 23.2.x
ASM	1.14.6-am1, 1.11.8-am1. 1.9.8-am1
OCNWDAF	23.4.x
PROVGW	23.4.x
OCCM	23.4.x

Release 23.4.2

The following table lists the versions of added or updated components in release 23.4.2:

Table 1-3 Compatibility Matrix

Network Functions	Compatible Versions
BSF	23.4.x
NRF	23.4.x
NSSF	23.4.x
Policy	23.4.x
SCP	23.4.x
SEPP	23.4.x
UDR	23.4.x

CNC Console is compatible with the following components:

Table 1-4 Compatibility Matrix

Components	Compatible Versions
OCNADD	23.4.x
CNE	23.4.x, 23.3.x, 23.2.x
cnDBTier	23.4.x, 23.3.x, 23.2.x
CDCS	23.4.x, 23.3.x, 23.2.x
OSO	23.4.x, 23.3.x, 23.2.x
ASM	1.14.6-am1, 1.11.8-am1. 1.9.8-am1
OCNWDAF	23.4.x
PROVGW	23.4.x
OCCM	23.4.x

Release 23.4.1

The following table lists the versions of added or updated components in release 23.4.1:

Table 1-5 Compatibility Matrix

Network Functions	Compatible Versions
BSF	23.4.x
NRF	23.4.x
NSSF	23.4.x
Policy	23.4.x
SCP	23.4.x
SEPP	23.4.x
UDR	23.4.x

CNC Console is compatible with the following components:

Table 1-6 Compatibility Matrix

Components	Compatible Versions
OCNADD	23.4.x
CNE	23.4.x, 23.3.x, 23.2.x
cnDBTier	23.4.x, 23.3.x, 23.2.x
CDCS	23.4.x, 23.3.x, 23.2.x
OSO	23.4.x, 23.3.x, 23.2.x
ASM	1.14.6-am1, 1.11.8-am1. 1.9.8-am1
OCNWDAF	23.4.x
PROVGW	23.4.x
OCCM	23.4.x

Release 23.4.0

The following table lists the versions of added or updated components in release 23.4.0:

Table 1-7 Compatibility Matrix

Network Functions	Compatible Versions
BSF	23.4.x
NRF	23.4.x
NSSF	23.4.x
Policy	23.4.x
SCP	23.4.x
SEPP	23.4.x
UDR	23.4.x

CNC Console is compatible with the following components:

Table 1-8 Compatibility Matrix

Components	Compatible Versions
OCNADD	23.4.x
CNE	23.4.x, 23.3.x, 23.2.x
cnDBTier	23.4.x, 23.3.x, 23.2.x
CDCS	23.4.x, 23.3.x, 23.2.x
OSO	23.3.x, 23.2.x, 22.3.x
ASM	1.14.6-am1, 1.11.8-am1. 1.9.8-am1
OCNWDAF	23.4.x
PROVGW	23.4.x
OCCM	23.4.x

1.3 CNC Console Deployment Architecture

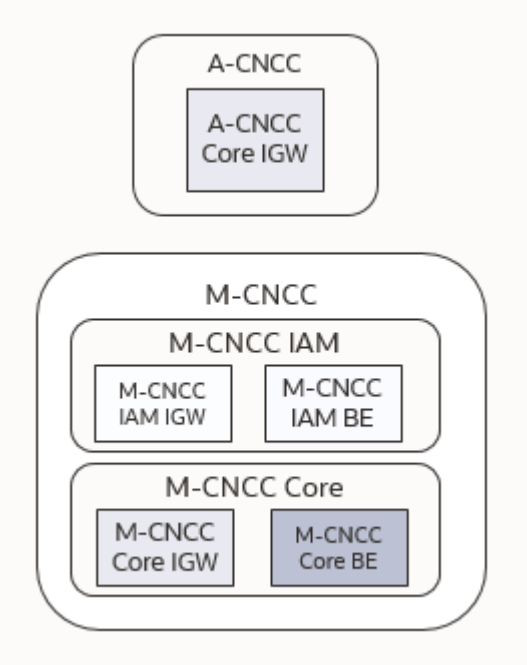
Introduction

The CNC Console supports both single and multiple cluster deployments. In a single cluster deployment, the CNC Console manages NFs and CNE common services deployed in the local Kubernetes clusters. In a multicluster deployment, the CNC Console manages NFs and CNE common services deployed in the remote Kubernetes clusters. This section explains the Console component overview, terminologies used, and Console single Cluster and multicluster 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 two components:

- M-CNCC
- A-CNCC

M-CNCC

Manager CNC Console or M-CNCC is a CNC Console 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 CNE common services(s). It is managed by M-CNCC.

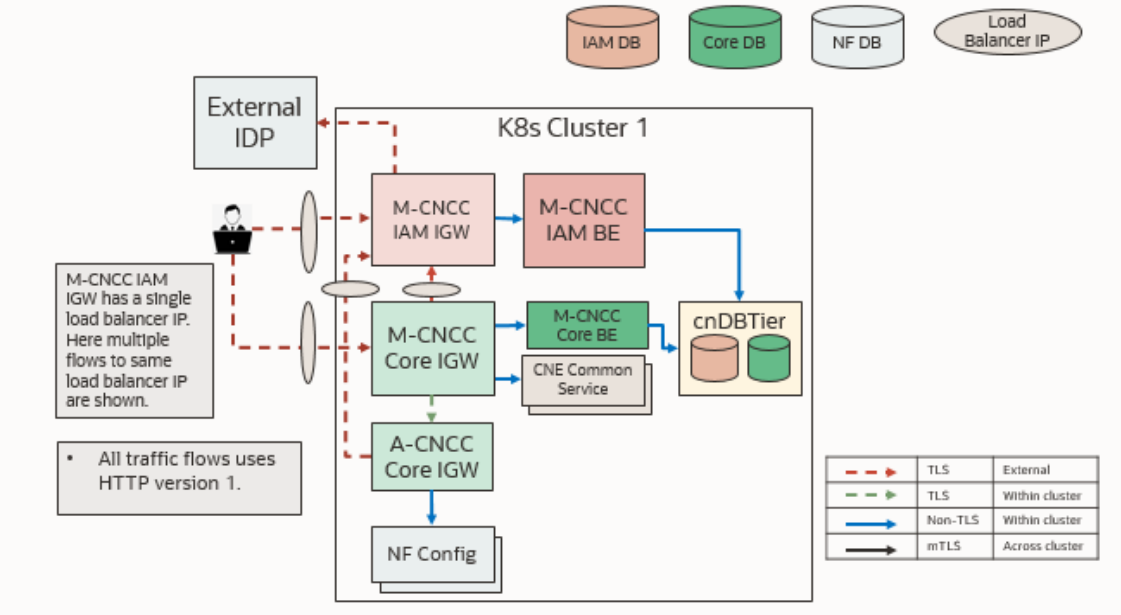
- A-CNCC Core contains A-CNCC Core Ingress Gateway.
- A-CNCC Core has no IAM component.

1.3.1 Single Cluster Deployment Architecture

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

The following diagram represents the CNC Console single cluster deployment:

Figure 1-2 CNC Console Single Cluster Deployment Architecture



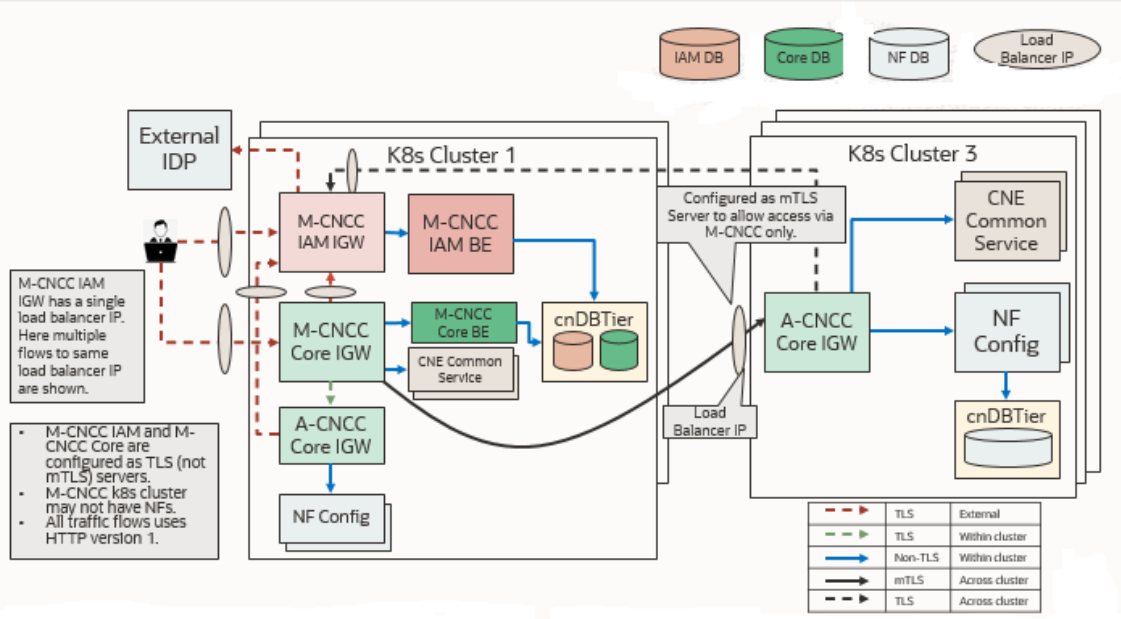
1.3.2 Multicluster Deployment Architecture

In a multicluster deployment, CNC Console can manage NFs and CNE 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 multicluster deployment:

Figure 1-3 CNC Console Multi Cluster Deployment Architecture



> **Note**
>
> - For a single cluster deployment both manager (CNCC IAM, M-CNCC Core) and agent (A-CNCC) are to be deployed on the same cluster.
> - For a multicluster deployment, if manager cluster has a local NF deployment then both manager (CNCC IAM, M-CNCC Core) and agent (A-CNCC) is 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.
> - 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 services in absence of a Manager in the same cluster.
> - Agent is needed only when NFs are present on the cluster.

1.4 Supported Deployment Models

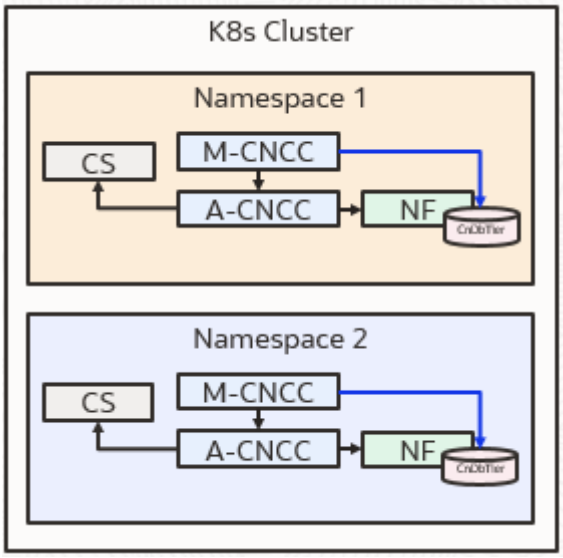
The following deployment models are supported by CNC Console:

- Single Cluster, Single Instance (Dedicated Console for each NF in a cluster)
- Single Cluster, Multiple Instances (One Console for many NFs or Instances in a cluster)
- Multiple Clusters, Single Instance (Multiple clusters with single NF or Instance in each cluster, M-CNCC/A-CNCC sitting in same or different clusters)
- Multiple Clusters, Multiple Instances (Multiple clusters with multiple NF/Instance in each cluster, M-CNCC/A-CNCC sitting in same or different clusters)

1.4.1 Deployment Model 1 - Single Cluster, Single Instance (Dedicated Console for each NF in a cluster)

This deployment model has dedicated Console for each NF in a cluster.

Figure 1-4 Deployment Model 1 - Single Cluster, Single Instance (Dedicated Console for each NF in a cluster)



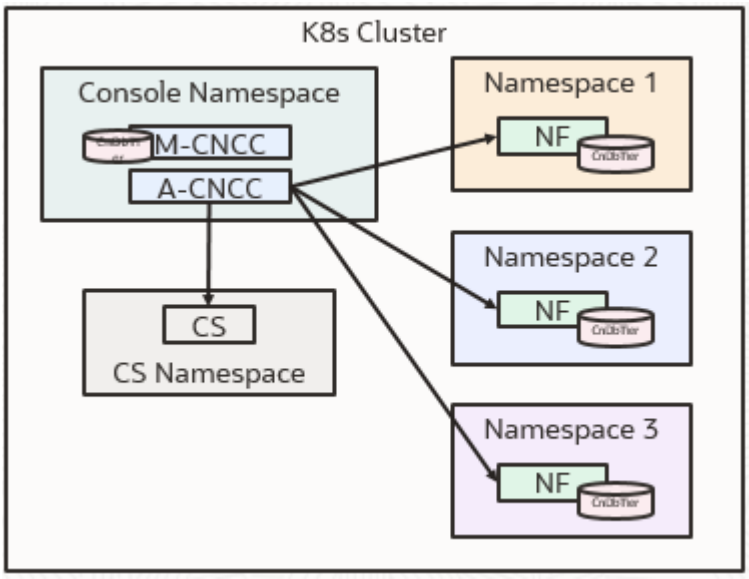
The Deployment Model 1has the following characteristics:

- Dedicated Console for a NF. Only single instance of NF is supported.
- Console shares the cnDBTier with NF.
- Any failure in NF cnDBTier impacts Console. Access to CNE Common Services (CS) is lost in case of any failure in NF cnDBTier.
- NFs and Console release compatibility must be maintained.
- M-CNCC and A-CNCC are managed by single Helm chart.

1.4.2 Deployment Model 2 - Single Cluster, Multiple Instances (One Console for many NFs/Instances in a cluster)

This deployment model has one Console for many NFs/Instances in a cluster.

Figure 1-5 Deployment Model 2 - Single Cluster, Multiple Instances (One Console for many NFs/Instances in a cluster)



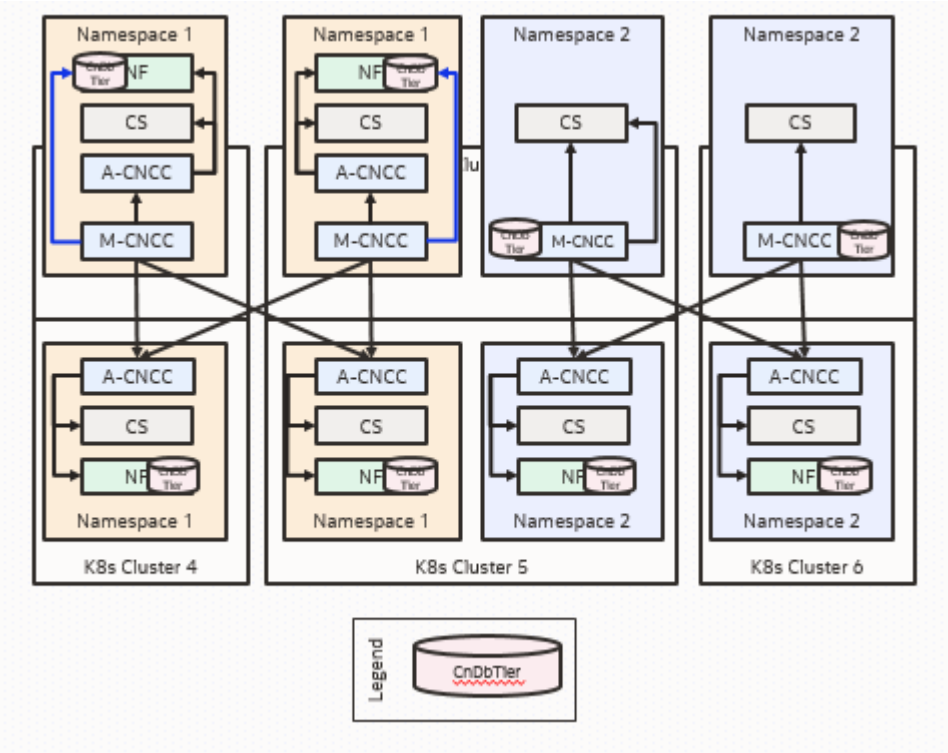
The Deployment Model 2 has the following characteristics:

- Shared Console for multiple NFs. NFs can be of same or different NF Type.
- Console needs dedicated cnDBTier for M-CNCC.
- NFs and Console release compatibility must be maintained.
- M-CNCC and A-CNCC are managed by single Helm chart.

1.4.3 Deployment Model 3 - Multiple Clusters, Single Instance (Multiple clusters with single NF or Instance in each cluster, M-CNCC or A-CNCC sitting in the same or different clusters)

This deployment model is for multiple clusters with single NF or Instance in each cluster. M-CNCC or A-CNCC can be in same or different clusters.

Figure 1-6 Deployment Model 3 - Multiple Clusters, Single Instance (Multiple clusters with single NF/Instance in each cluster, M-CNCC/A-CNCC sitting in same/different clusters)



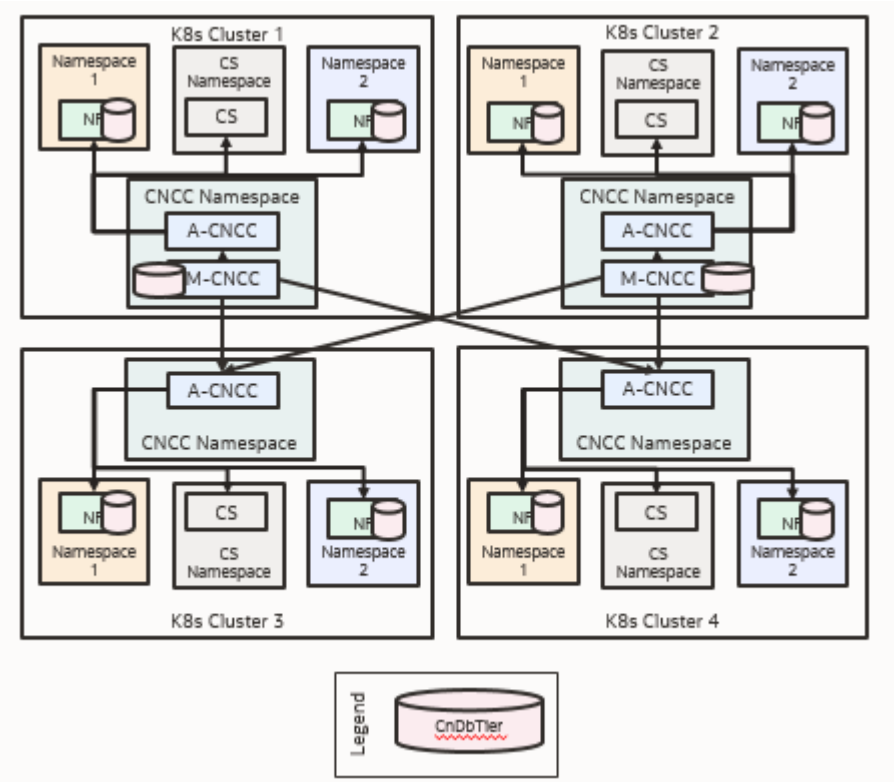
The deployment model 3 has the following characteristics:

- Dedicated Console for a single NF
- Console shares the cnDBTier with N.
- Any failure in NF cnDBTier impacts Console. Access to CNE Common Services (CS) is lost on any failure in NF cnDBTier.
- NFs and Console release compatibility must be maintained.
- Managers can be deployed as Active/Active/Active.
- Multiple Agents are supported.
- M-CNCC can be deployed without A-CNCC in case there are no local NFs to be managed at the Kubernetes Cluster.
- M-CNCC and A-CNCC are managed by single Helm chart.
- M-CNCC cannot manage NFs located in another M-CNCC Kubernetes Cluster.

1.4.4 Deployment Model 4 - Multiple Clusters, Multiple Instances (Multiple clusters with multiple NF/Instance in each cluster, M-CNCC/A-CNCC sitting in same/different clusters)

This deployment model is for multiple clusters with multiple NF/Instance in each cluster. M-CNCC/A-CNCC can be in same/different clusters.

Figure 1-7 Deployment Model 4 - Multiple Clusters, Multiple Instances (Multiple clusters with multiple NF/Instance in each cluster, M-CNCC/A-CNCC sitting in same/ different clusters)



The deployment model 4 has the following characteristics:

- Shared Console for multiple NFs. NFs can be of same or different NF Type.
- Console needs dedicated cnDBTier for M-CNCC.
- NFs and Console release compatibility must be maintained.
- Managers can be deployed as Active/Active/Active.
- Multiple Agents are supported.
- M-CNCC and A-CNCC are managed by Single Helm Chart.
- M-CNCC cannot manage NFs located in another M-CNCC Kubernetes Cluster.

1.5 CNC Console Deployment Model Matrix

CNC Console Deployment Model Matrix

The following table provides details on support of console deployment models for various network functions:

Table 1-9 CNC Console Deployment Model Matrix

Deployment Models	Policy	BSF	SCP	UDR	NRF	SEPP	NSSF	DD	PROVGW	NWDAF	OCCM
Model 1 - Single Cluster, Single Instance (Dedicated Console for each NF in a cluster)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Model 2 - Single Cluster, Multiple Instances (One Console for many NFs/ Instances in a cluster)	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes
Model 3 - Multiple Clusters, Single Instance (Multiple clusters with single NF/ Instance in each cluster, M-CNCC/A-CNCC sitting in same/ different clusters)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 1-9 (Cont.) CNC Console Deployment Model Matrix

Deployment Models	Policy	BSF	SCP	UDR	NRF	SEPP	NSSF	DD	PROVGW	NWDAF	OCCM
Model 4 - Multiple Clusters, Multiple Instances (Multiple clusters with multiple NF/ Instance in each cluster, M-CNCC/A-CNCC sitting in same/ different clusters)	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes

> **Note**
>
> - Single instance of CNC Console supports either single instances of all NFs or multiple instances of a single NF types within a Kubernetes cluster.
> - For example, a single instance of CNC Console can either handle a single instance of Policy, SCP, NRF and so on, or multiple instances of Policy or SCP within a Kubernetes cluster.

1.6 Configuration Workflow

This section explains the configuration workflow for the following scenarios:

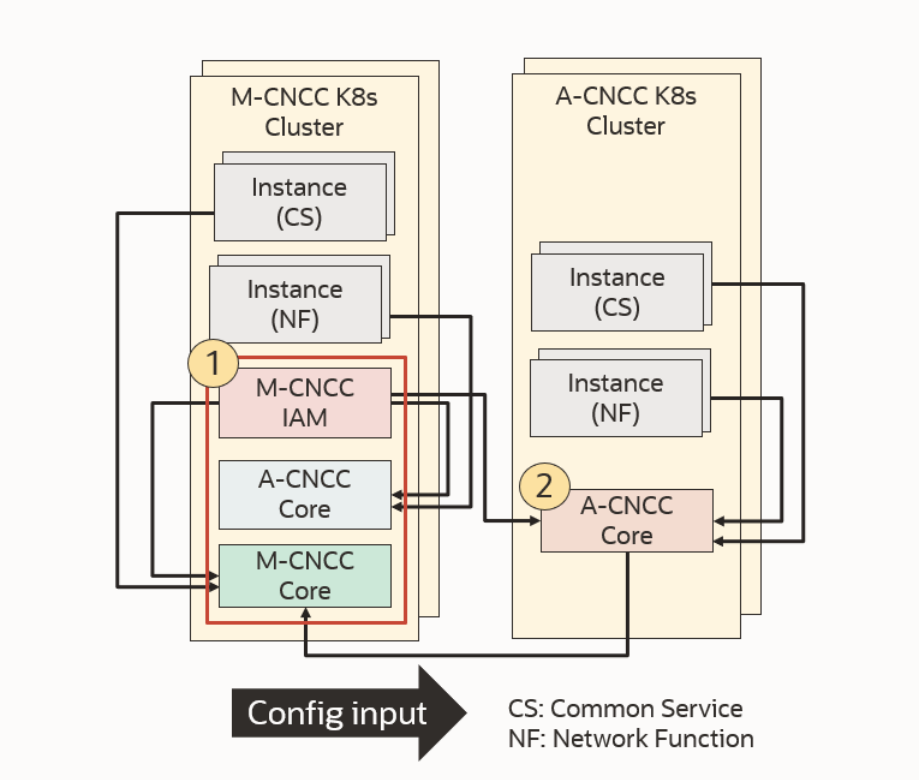
- Fresh Installation
- Add a new M-CNCC
- Add a new A-CNCC
- Remove M-CNCC from A-CNCC
- Remove A-CNCC from M-CNCC

1.6.1 Fresh Installation of M-CNCC and A-CNCC

This section explains how to do a fresh installation of M-CNCC and A-CNCC.

The following diagram represents the fresh installation of M-CNCC and A-CNCC.

Figure 1-8 Fresh Installation of M-CNCC and A-CNCC



Procedure:

1. Install M-CNCC IAM, A-CNCC Core, and 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 Kubernetes cluster is optional and needed only if there are NF instances on the M-CNCC Kubernetes 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 Kubernetes clusters.
   - Configuration Input: M-CNCC IAMs and local instances

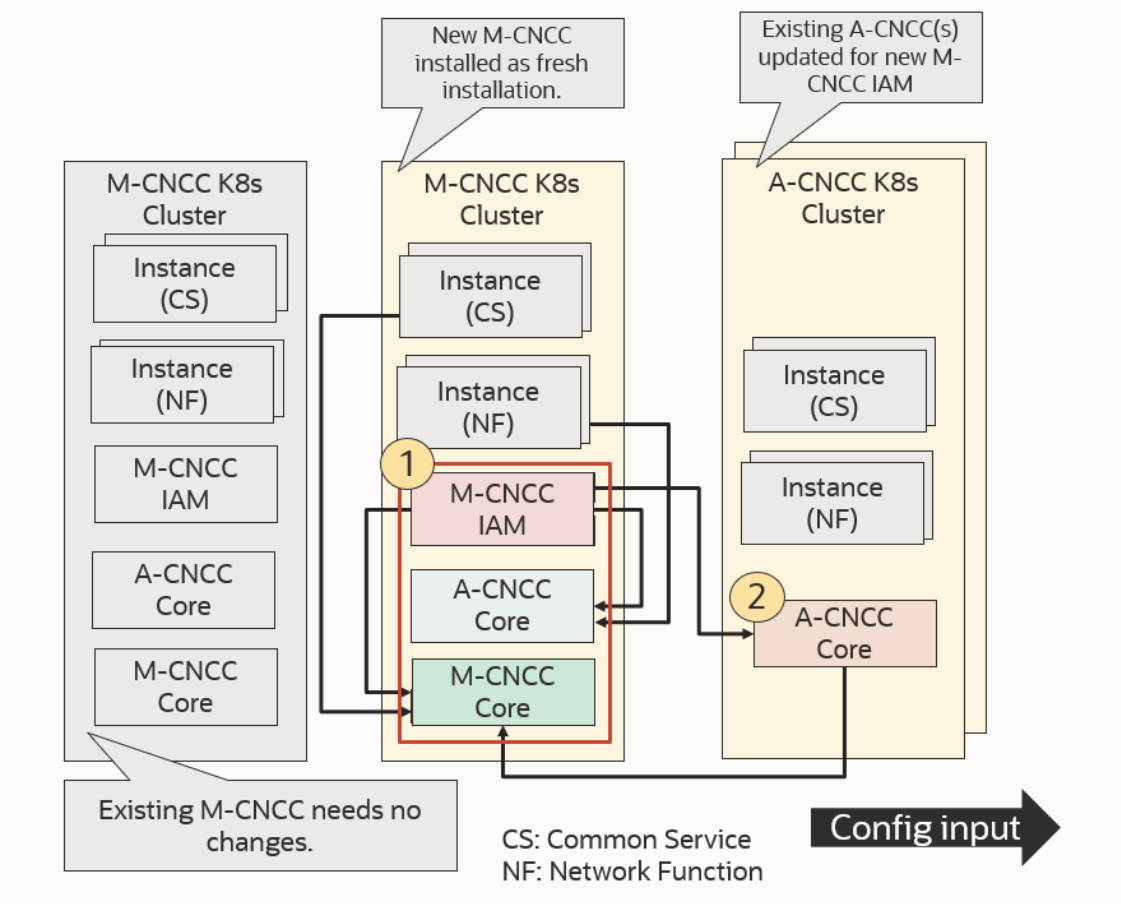
See Installing CNC Console section for installation procedure.

1.6.2 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 1-9 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 Kubernetes 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 Kubernetes cluster(s)

- Configuration Update: Newly added M-CNCC IAM

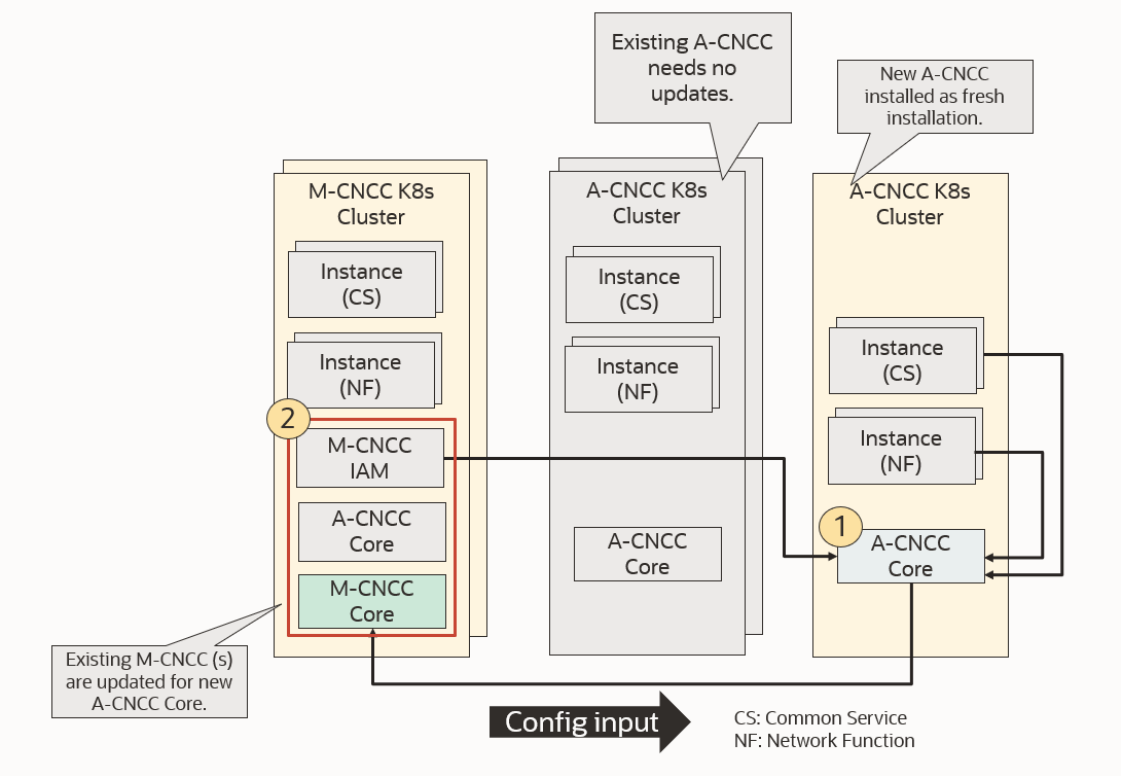
See Installing CNC Console section for installation procedure.

1.6.3 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 1-10 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.

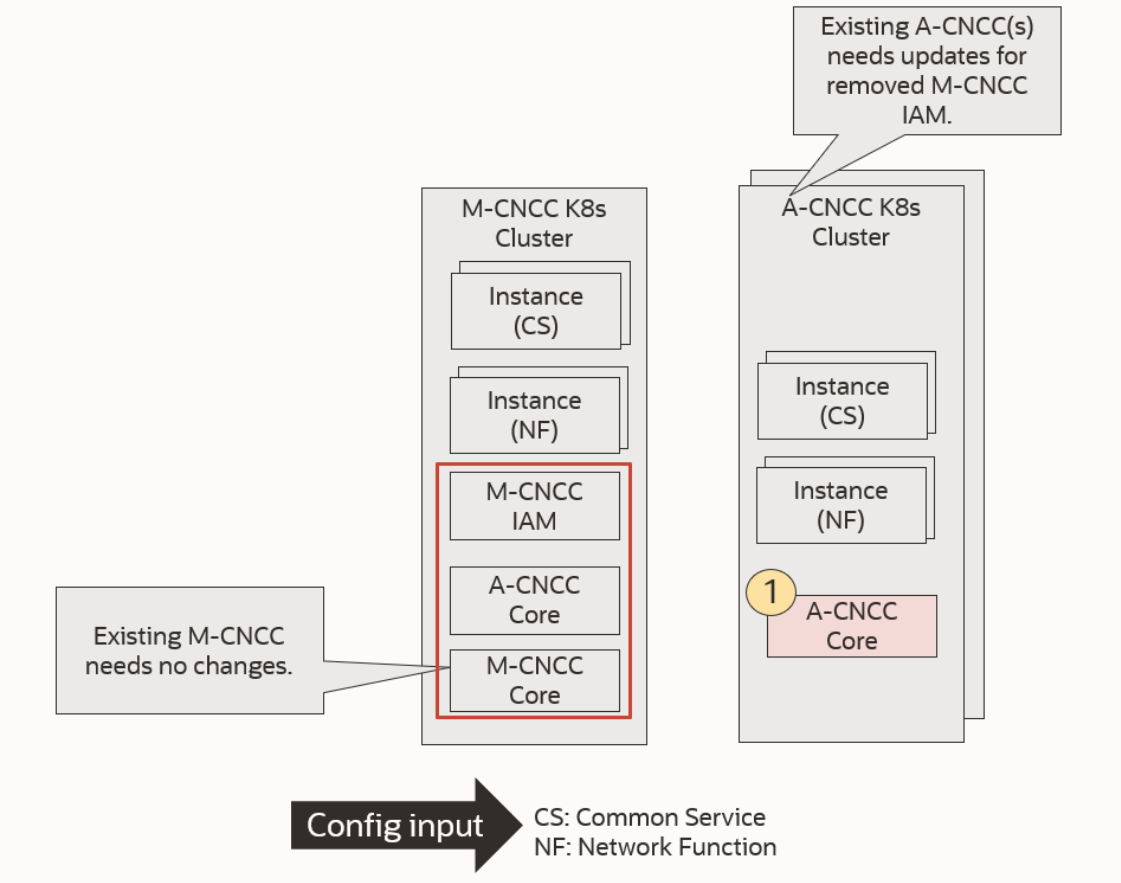
See Installing CNC Console section for installation procedure.

1.6.4 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 1-11 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

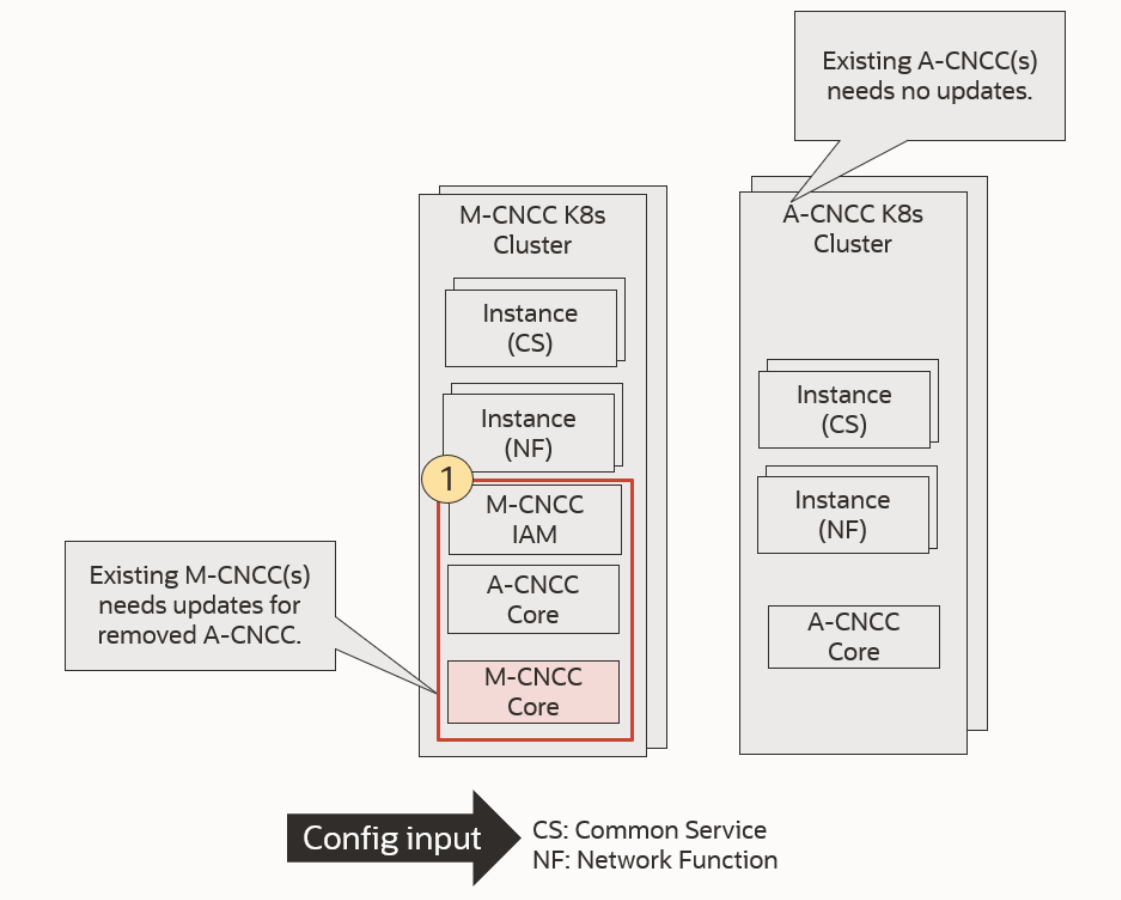
See Installing CNC Console section for installation procedure.

1.6.5 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 1-12 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.

1.7 CNC Console Configuration Maximum Limits

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

Table 1-10 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

1.8 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, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Service Communication Proxy Installation Guide*
- *Oracle Communications Cloud Native Core, Unified Data Repository Installation, Upgrade, and Fault Recovery 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*
- *Oracle Communications Networks Data Analytics Installation and Fault Recovery Guide*

2
Installing CNC Console

This chapter provides information about installing Oracle Communications Cloud Native Configuration Console (CNC Console) in a cloud native environment.

> **Note**
>
> CNC Console supports fresh installation. For more information on how to upgrade CNC Console, see Upgrading CNC Console section.

2.1 Prerequisites

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

2.1.1 Software Requirements

This section lists the software that must be installed before installing CNC Console:

Install the following software before installing CNC Console:

Table 2-1 Preinstalled Software

Software	Version
Kubernetes	1.27.x, 1.26.x, 1.25.x
Helm	3.12.3
Podman	4.4.1

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

```
kubectl version
```

```
helm version
```

```
podman version
```

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

```
helm ls -A
```

The list of additional software items, along with the supported versions and usage, is provided in the following table:

Table 2-2 Additional Software

Software	Version	Required For
Opensearch	2.3.0	Logging
OpenSearch Dashboard	2.3.0	Logging
Kyverno	1.9	Logging
FluentBit	1.9.4	Logging
Grafana	9.1.7	KPIs
Prometheus	2.44.0	Metrics
MetalLB	0.13.7	External IP
Jaeger	1.45.0	Tracing
snmp-notifier	1.2.1	Alerts

2.1.2 Environment Setup Requirements

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

Client Machine Requirement

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

Client machine must have:

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

Network Access Requirement

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

- Local helm repository, where the 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
  ```

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

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

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

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

where:

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

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

Server or Space Requirement

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

CNE Requirement

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

To check the CNE version, run the following command:

```
echo $OCCNE_VERSION
```

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

cnDBTier Requirement

CNC Console supports cnDBTier . cnDBTier must be configured and running before installing CNC Console. For more information about installation procedure, see *Oracle Communications cnDBTier Installation, Upgrade, and Fault Recovery 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*.

2.1.3 CNC Console Resource Requirement

This section lists the resource requirements to install and run CNC Console.

CNC Console and cnDBTier Resource Usage Guidelines

This section explains the guidelines for CNC Console and cnDBTier resource usage guidelines.

> **Note**
>
> In case of deployment using shared DBTier between NF and Console, you must include Console DB Profile sizing in NF DB Profile sizing.

> **Note**
>
> - DBProfile replica count to be updated as per GR setup.
> - Depending on GR setup of two, three, or four site choose replica count two, four, or six for **SQL** (ndbmysqld).

Table 2-3 CNC Console and cnDBTier Resource Usage

Deployment Model	cnDBTier Usage	DBTier Resource Profile	Console Resources
Model 1 - Single Cluster, Single Instance (dedicated Console for each NF in a cluster)	Console and NF have a single shared DBTier • Manager IAM and Manager Core on same Kubernetes cluster use shared DBTier	• DBProfile • Agent Core on a same Kubernetes cluster doesn't have any DBtier dependency. For the details, see cnDBTier Profiles	• For CNC Console Single Cluster Deployment Resource usage, see CNC Console Resource Requirement
Model 2 - Single Cluster, Multiple Instances (One Console for many NFs/ Instances in a cluster)	Dedicated DBTier for Console • Manager IAM and Manager Core on same Kubernetes cluster use shared DBTier	• DBProfile • Agent Core on a same Kubernetes cluster doesn't have any DBtier dependency. For the details, see cnDBTier Profiles	• For CNC Console Single Cluster Deployment Resource usage, see CNC Console Resource Requirement

Table 2-3 (Cont.) CNC Console and cnDBTier Resource Usage

Deployment Model	cnDBTier Usage	DBTier Resource Profile	Console Resources
Model 3 - Multiple Clusters, Single Instance. (Multiple clusters with single NF/ Instance in each cluster, M-CNCC/A-CNCC sitting in same/different clusters)	Console and NF have a single shared DBTier • Manager IAM and Manager Core on same Kubernetes cluster use shared DBTier	• Manager - DBProfile • Agent Core on a remote Kubernetes cluster doesn't have any DBtier dependency. For the details, see cnDBTier Profiles	• For CNC Console Manager with Agent Deployment, see CNC Console Resource Requirement • For CNC Console Manager Only Deployment, see CNC Console Resource Requirement • For CNC Console Agent Only Deployment, see CNC Console Resource Requirement
Model 4 - Multiple Clusters, Multiple Instances (Multiple clusters with multiple NF/Instance in each cluster, M-CNCC/A-CNCC sitting in same/ different clusters)	Dedicated DBTier for Console per Kubernetes cluster • Manager IAM and Manager Core on same Kubernetes cluster use single Console DBTier	• Manager - DBProfile • Agent Core on a remote Kubernetes cluster doesn't have any DBtier dependency. For the details, see cnDBTier profiles	• For CNC Console Manager with Agent Deployment, see CNC Console Resource Requirement • For CNC Console Manager Only Deployment , see CNC Console Resource Requirement • For CNC Console Agent Only Deployment , see CNC Console Resource Requirement

Note

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

Resource Usage for CNC Console Deployment

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

Resource Usage for CNC Console Single Cluster Deployment

Single Cluster Deployment includes M-CNCC IAM, M-CNCC Core and A-CNCC Core components. It also includes common resource needed for manager or agent deployment.

Table 2-4 Resource Usage for CNC Console Single Cluster Deployment

Component	Limits		Requests	
	CPU	Memory (Gi)	CPU	Memory (Gi)
M-CNCC IAM	4.5	4.5	4.5	4.5
M-CNCC Core	4	4	4	4
A-CNCC Core	2	2	2	2
CNCC Common Resource	2	2	2	2
Total	12.5	12.5	12.5	12.5

Formula

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

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

Table 2-5 Resource Usage for CNC Console Multicluster Deployment

Component	Limits		Requests	
	CPU	Memory (Gi)	CPU	Memory (Gi)
M-CNCC IAM	4.5	4.5	4.5	4.5
M-CNCC Core	4	4	4	4
A-CNCC Core	2	2	2	2
CNCC Common Resource	2	2	2	2
*No Of Agents In Other Clusters	2			
Total	18.5	18.5	18.5	18.5

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

Formula to calculate total resource usage:

Total Resource = M-CNCC IAM Resource + M-CNCC Core Resource + Common Resources + (No Of Agents In Other Clusters x (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.

Table 2-6 CNC Console Manager Only Deployment

Component	Limits		Requests	
	CPU	Memory (Gi)	CPU	Memory (Gi)

Table 2-6 (Cont.) CNC Console Manager Only Deployment

M-CNCC IAM	4.5	4.5	4.5	4.5
M-CNCC Core	4	4	4	4
A-CNCC Core	0	0	0	0
CNCC Common Resource	2	2	2	2
Total	10.5	10.5	10.5	10.5

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 2-7 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	2	2	2	2
CNCC Common Resource	2	2	2	2
Total	4	4	4	4

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 2-8 CNC Console Manager with Agent Deployment

Component	Limits		Requests	
	CPU	Memory (Gi)	CPU	Memory (Gi)
M-CNCC IAM	4.5	4.5	4.5	4.5
M-CNCC Core	4	4	4	4
A-CNCC Core	2	2	2	2
CNCC Common Resource	2	2	2	2
Total	12.5	12.5	12.5	12.5

CNC Console Component wise Resource Usage

Table 2-9 CNCC Common Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
hookJobResources	NA	2	2	2	2	Common Hook Resource
helm test	cncc-test	0	0	0	0	Uses hookJobResources
Total		2	2	2	2	

> **Note**
>
> - Debug tool resources are not considered in the calculation. Debug tool resources usage is per pod, if debug tool is enabled for more than one pod then max 1vCPU and 2Gi Memory per pod is needed.
> - Service Mesh (ASM) sidecar resources are not considered in the calculation. Service Mesh sidecar resources usage is per pod, that is, if Service Mesh is enabled and sidecar is injected, then max 1vCPU and 1Gi Memory per pod is needed.

Table 2-10 M-CNCC IAM Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
cncc-iam-ingress-gateway	ingress-gateway	2	2	2	2	
	init-service*	0	0	0	0	Applicable when HTTPS is enabled. *Init-service container's resources are not counted because the container gets terminated after initialization completes.
	common_config_hook	0	0	0	0	common_config_hook not used in IAM
cncc-iam-kc-http	kc	2	2	2	2	

Table 2-10 (Cont.) M-CNCC IAM Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
	init-service*	0	0	0	0	Optional, used for enabling LDAPS. *Init-service container's resources are not counted because the container gets terminated after initialization completes.
	healthcheck	0.5	0.5	0.3	0.3	
	cnnc-iam--pre-install	0	0	0	0	Uses hookJobResources
	cnnc-iam-pre-upgrade	0	0	0	0	Uses hookJobResources
	cnnc-iam-post-install	0	0	0	0	Uses hookJobResources
	cnnc-iam-post-upgrade	0	0	0	0	Uses hookJobResources
Total		4.5	4.5	4.5	4.5	

Table 2-11 M-CNCC Core Resource Usage

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

Table 2-11 (Cont.) M-CNCC Core Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
	init-service*	0	0	0	9	Applicable when HTTPS is enabled. *Init-service container's resources are not counted because the container gets terminated after initialization completes.
	common_config_hook*	0	0	0	0	Common Configuration Hook container creates databases which are used by Common Configuration Client. *common_config_hook container's resources are not counted because the container gets terminated after initialization completes.
cncc-mcore-cmservice	cmservice	2	2	2	2	
	validation-hook	0	0	0	0	Uses common hookJobResources
Total		4	4	4	4	

Table 2-12 A-CNCC Core Resource Usage

Microservice Name	Containers	Limits		Requests		Comments
		CPU	Memory	CPU	Memory	
cncc-acore-ingress-gateway	ingress-gateway	2	2	2	2	
	init-service*	0	0	0	0	Applicable when HTTPS is enabled. *Init-service container's resources are not counted because the container gets terminated after initialization completes.
	common_config_hook*	0	0	0	0	Common Configuration Hook container creates databases which are used by Common Configuration Client. *Init-service container's resources are not counted because the container gets terminated after initialization completes.
	validation-hook	0	0	0	0	Uses common hookJobResources
Total		2	2	2	2	

2.2 Installation Sequence

This section describes the CNC Console preinstallation, installation, and postinstallation tasks.

2.2.1 Preinstallation Tasks

Before installing CNC Console, perform the tasks described in this section.

> **Note**
>
> For IPv4 or IPv6 configurations, see Support for IPv4 or IPV6 Configuration for CNC Console.

2.2.1.1 Downloading CNC Console Package

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

1. Log in to My Oracle Support using your login credentials.
2. Click the Patches & Updates tab to locate the patch.
3. In the Patch Search 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 list.
5. From the Release drop-down list, select "Oracle Communications Cloud Native Configuration Console <release_number>".
   Where, <release_number> indicates the required release number of CNCC.
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 on Download.
   The File Download window appears.
9. Click on the `<p********_<release_number>_Tekelec>.zip` file.
10. Extract the release package zip file to download the network function patch to the system where network function must be installed.
11. Extract the release package zip file.
    Package is named as follows:

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

    `occncc_csar_23_4_4_0_0.zip`

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

2.2.1.2 Library of text variables

This library includes text variables. Each text variable is defined by means of a <ph> element in a separate paragraph. The <ph> attribute 'varid' should hold a unique name for the variable.

Cloud Native Configuration Console

23.4.4

23_4_4

2.2.1.3 Pushing the Images to Customer Docker Registry

CNC Console deployment package includes ready-to-use images and Helm charts to help orchestrate containers in Kubernetes. The communication between Pods of services of CNC Console products are preconfigured in the Helm charts.

To push the images to the registry, perform the following steps:

1. Unzip the CNC Console package file: Unzip the CNC Console package file to the specific repository:

   `unzip occncc_csar_<marketing-release-number>.zip`
   The package consists of following files and folders:

   a. **Files**: CNC Console Docker images file and CNC Console Helm Charts

   b. **Scripts**: Custom values and alert files:

   - CNC Console custom values file : `occncc_custom_values_<version>.yaml`
   - CNC Console cnDBTier custom values file : `occncc_dbtier_<cndbtier_version>_custom_values_<version>.yaml`
   - CNC Console network policy custom values file : `occncc_network_policy_custom_values_<version>.yaml`
   - CNC Console IAM Schema file for rollback to previous version : `occncc_rollback_iam_schema_<version>.sql`
   - CNC Console Metric Dashboard file : `occncc_metric_dashboard_<version>.json`
   - CNC Console Metric Dashboard file for CNE supporting Prometheus HA (CNE 1.9.x onwards): `occncc_metric_dashboard_promha_<version>.json`
   - CNC Console Alert Rules file: `occncc_alertrules_<version>.yaml`
   - CNC Console Alert Rules file for CNE supporting Prometheus HA (CNE 1.9.x onwards): `occncc_metric_dashboard_promha_<version>.json`
   - CNC Console MIB files: `occncc_mib_<version>.mib`, `occncc_mib_tc_<version>.mib`
   - CNC Top level mib file: `toplevel.mib`

   c. **Definitions**: Definitions folder contains the CNE Compatibility and definition files.

   - `occncc_cne_compatibility.yaml`
   - `occncc.yaml`

   d. TOSCA-Metadata: `TOSCA.meta`

e. The package folder structure is as follows:

```
Definitions
       occncc_cne_compatibility.yaml
       occncc.yaml
Files
    apigw-common-config-hook-23.4.10.tar
    apigw-configurationinit-23.4.10.tar
    ChangeLog.txt
    cncc-apigateway-23.4.10.tar
    cncc-core-validationhook-23.4.4.tar
    cncc-iam-23.4.4.tar
    cncc-iam-healthcheck-23.4.4.tar
    cncc-iam-hook-23.4.4.tar
    Helm
        occncc-23.4.4.tgz
        occncc-network-policy-23.4.4.tgz
    Licenses
    nf_test-23.4.3.tar
    ocdebug-tools--23.4.3.tar
    Oracle.cert
    Tests
occncc.mf
Scripts
    occncc_alerting_rules_promha_23.4.4.yaml
    occncc_alertrules_23.4.4.yaml
    occncc_custom_values_23.4.4.yaml
    occncc_dbtier_23.4.0_custom_values_23.4.4.yaml
    occncc_metric_dashboard_23.4.4.json
    occncc_metric_dashboard_promha_23.4.4.json
    occncc_mib_23.4.4.mib
    occncc_mib_tc_23.4.4.mib
    occncc_network_policy_custom_values_23.4.4.yaml
    occncc_rollback_iam_schema_23.4.4.sql
    toplevel.mib
TOSCA-Metadata
    TOSCA.meta
```

For example,

```
Example: unzip occncc_csar_23_4_4_0_0.zip
Archive: occncc_csar_23_4_4_0_0.zip
inflating: Definitions/occncc_cne_compatibility.yaml
inflating: Definitions/occncc.yaml
creating: Files/
creating: Files/Tests/
creating: Files/Helm/
inflating: Files/Helm/occncc-23.4.4.tgz
extracting: Files/Helm/occncc-network-policy-23.4.4.tgz
creating: Files/Licenses/
inflating: Files/apigw-configurationinit-23.4.10.tar
inflating: Files/apigw-common-config-hook-23.4.10.tar
inflating: Files/ocdebug-tools-23.4.3.tar
inflating: Files/cncc-apigateway-23.4.10.tar
inflating: Files/cncc-iam-23.4.4.tar
inflating: Files/cncc-iam-hook-23.4.4.tar
```

```
inflating: Files/cncc-iam-healthcheck-23.4.4.tar
inflating: Files/nf_test-23.4.3.tar
inflating: Files/cncc-core-validationhook-23.4.4.tar
inflating: Files/cncc-cmservice-23.4.4.tar
inflating: Files/ChangeLog.txt
extracting: Files/Oracle.cert
creating: Scripts/
inflating: Scripts/occncc_custom_values_23.4.4.yaml
inflating: Scripts/occncc_network_policy_custom_values_23.4.4.yaml
inflating: Scripts/occncc_mib_tc_23.4.4.mib
inflating: Scripts/occncc_mib_23.4.4.mib
inflating: Scripts/toplevel.mib
inflating: Scripts/occncc_metric_dashboard_23.4.4.json
inflating: Scripts/occncc_metric_dashboard_promha_23.4.4.json
inflating: Scripts/occncc_alertrules_23.4.4.yaml
inflating: Scripts/occncc_alerting_rules_promha_23.4.4.yaml
inflating: Scripts/occncc_rollback_iam_schema_23.4.4.sql
inflating: Scripts/ occncc_dbtier_23.4.0_custom_values_23.4.4.yaml
creating: TOSCA-Metadata/
inflating: TOSCA-Metadata/TOSCA.meta
inflating: occncc.mf
```

2. Run the following command to move to Files Folder To load docker image :

   ```
   cd Files
   ```

3. Run the following command to load the Docker images to docker registry:

   ```
   docker load --input <image-name>:<image-tag>.tar
   docker tag <image-name>:<image-tag> <docker-repo>/<image-name>:<image-tag>
   docker push <docker_repo>/<image_name>:<image-tag>
   ```

   Example:

   ```
   docker load --input apigw-common-config-hook-23.4.10.tar
   docker tag occncc/apigw-common-config-hook:23.4.10 <docker-repo>/occncc/
   apigw-common-config-hook:23.4.4
   docker push <docker-repo>/occncc/apigw-common-config-hook:23.4.10

   docker load --input cncc-core-validationhook-23.4.4.tar
   docker tag occncc/cncc-core-validationhook:23.4.4 <docker-repo>/occncc/
   cncc-core-validationhook:23.4.4
   docker push <docker-repo>/occncc/cncc-core-validationhook:23.4.4

   docker load --input nf_test-23.4.3.tar
   docker tag occncc/nf_test:23.4.3 <docker-repo>/occncc/nf_test:23.4.3
   docker push <docker-repo>/occncc/nf_test:23.4.3

   docker load --input apigw-configurationinit-23.4.10.tar
   docker tag occncc/apigw-configurationinit:23.4.10 <docker-repo>/occncc/
   apigw-configurationinit:23.4.10
   docker push <docker-repo>/occncc/apigw-configurationinit:23.4.10

   docker load --input cncc-iam-23.4.4.tar
   docker tag occncc/cncc-iam:23.4.4 <docker-repo>/occncc/cncc-iam:23.4.4
   ```

```
docker push <docker-repo>/occncc/cncc-iam:23.4.4

docker load --input ocdebug-tools-23.4.3.tar
docker tag occncc/ocdebug-tools:23.4.3 <docker-repo>/occncc/ocdebug-
tools:23.4.3
docker push <docker-repo>/occncc/ocdebug-tools:23.4.3

docker load --input cncc-iam-healthcheck-23.4.4.tar
docker tag occncc/cncc-iam-healthcheck:23.4.4 <docker-repo>/occncc/cncc-
iam-healthcheck:23.4.4
docker push <docker-repo>/occncc/cncc-iam-healthcheck:23.4.4

docker load --input cncc-iam-hook-23.4.4.tar
docker tag occncc/cncc-iam-hook:23.4.4 <docker-repo>/occncc/cncc-iam-
hook:23.4.4
docker push <docker-repo>/occncc/cncc-iam-hook:23.4.4

docker load --input cncc-apigateway-23.4.10.tar
docker tag occncc/cncc-apigateway:23.4.10 <docker-repo>/occncc/cncc-
apigateway:23.4.10
docker push <docker-repo>/occncc/cncc-apigateway:23.4.10

docker load --input cncc-cmservice-23.4.4.tar
docker tag occncc/cncc-cmservice:23.4.4<docker-repo>/occncc/cncc-
cmservice:23.4.4
docker push <docker-repo>/occncc/cncc-cmservice:23.4.4
```

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

   ```
   docker images
   ```

5. Run the following command to move to the Helm Directory

   ```
   cd Helm
   ```

   Example:

   ```
   helm cm-push --force occncc-23.4.4.tgz ocspf-helm-repo
   ```

6. Run the following command to push helm charts to Helm Repository:

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

   For example:

   ```
   helm cm-push --force occncc-23.4.4.tgz ocspf-helm-repo
   ```

2.2.1.4 Verifying and Creating CNC Console Namespace

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

> **Note**
>
> This is a mandatory procedure, run this before proceeding further with the installation. The namespace created or verified in this procedure is an input for the next procedures.

To verify and create a namespace:

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

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

   ```
   kubectl get namespaces
   ```

2. 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>
   ```

3. For Example:

   ```
   kubectl create namespace cncc
   ```

Sample output: `namespace/cncc created`

Naming Convention for Namespaces

The namespace should:

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

> **Note**
>
> It is recommended to avoid using prefix `kube-` when creating namespace. 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 CNC Console Debug Tools.

2.2.1.5 Creating Service Account, Role, and Rolebinding

2.2.1.5.1 Global Service Account Configuration

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

> **Note**
>
> The secrets should exist in the same namespace where CNC Console is getting deployed. This helps to bind the Kubernetes role with the given service account.

1. Run the following command to create an CNC Console resource file:

   ```
   vi <occncc-resource-file>
   ```

   Example:`vi occncc-resource-template.yaml`
2. Update the `occncc-resource-template.yaml` file with release specific information

   > **Note**
   >
   > Update `<helm-release>` and `<namespace>` with its respective CNC Console namespace and CNC Console Helm release name.

   A sample CNC Console service account yaml file is as follows:

   ```
   ## 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
```

3. Run the following command to create service account, role, and role binding:

   ```
   kubectl -n <occncc-namespace> create -f occncc-resource-template.yaml
   ```

4. Update the `serviceAccountName` parameter in the `occncc_custom_values_<version>.yaml` file with the value updated in the `name` field for ingress-gateway and keycloak.
   For Ingress Gateway and Keycloak provide custom service account under **global.serviceAccountName.** as follows:

   ```
   global:

     serviceAccountName: &serviceAccountName cncc-sa

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

2.2.1.5.2 Helm Test Service Account Configuration

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

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

```
global:

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

  helmTestServiceAccountName: cncc-helmtest-serviceaccount
```

1. Run the following command to create an CNC Console resource file:

   ```
   vi <occncc-resource-file>
   ```

   Example:`vi occncc-resource-template.yaml`

2. Update the `occncc-resource-template.yaml` file with release specific information

   > **Note**
   >
   > Update `<helm-release>` and `<namespace>` with its respective CNC Console namespace and CNC Console Helm release name.

   A sample CNC Console service account yaml file is as follows:

   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
```

3. Run the following command to create service account, role, and role binding:

   ```
   kubectl -n <occncc-namespace> create -f occncc-resource-template.yaml
   ```

4. Update the `global.helmTestServiceAccountName` parameter in the `occncc_custom_values_<version>.yaml` file with the value updated in the `name` field for ingress-gateway and keycloak.

> **Note**
>
> If user doesn't want to create the separate service account for helm test, then logging of resources and complaint check could also 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.

2.2.1.6 Configuring Database

This section explains how database administrators can create users and database in a single and multisite deployment.

> **Note**
>
> - Before running the procedure for georedundant sites, ensure that the DBTier for georedundant sites is up and replication channels are enabled.
> - While performing a fresh installation, if CNCC release is already deployed, purge the deployment, and remove the database and users that were used for the previous deployment. For uninstallation procedure, see Uninstalling CNC Console .
> - If cnDBTier 23.3.0 is used during installation, set the ndb_allow_copying_alter_table parameter to 'ON' in the occncc_dbtier_23.3.0_custom_values_23.3.0.yaml file before installing CNC Console. After CNC Console installation, the parameter value can be set to its default value 'OFF'.

> **Caution**
>
> Verify the value of the following parameters, before deploying CNC Console in a three site georedundency setup:
>
> ndb:
>
> MaxNoOfOrderedIndexes: 1024
>
> MaxNoOfTables: 1024
>
> NoOfFragmentLogFiles: 512

Naming Convention for CNCC Database

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
```

The name of the database should:

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

2.2.1.6.1 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 ndbappmysqld-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>';
```

> **Note**
>
> You must use a strong non-predictable password consisting of complex strings of more than 10 mixed characters.

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;
```

2.2.1.6.2 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**
>
> 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. 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 ndbappmysqld-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 M-CNCC IAM database already exists:
   a. Run the following command to check if database exists:

      ```
      show databases;
      ```

      i. If any of the previously configured databases are already present, remove them. Otherwise skip this step.
      Run the following command to drop the existing cnccdb database:

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

      Example:

      ```
      DROP DATABASE cnccdb;
      ```

   b. Run the following command to create the Database:

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

      Example:

      ```
      $ CREATE DATABASE IF NOT EXISTS cnccdb;
      ```

   c. 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:

      ```
      $ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, REFERENCES,
      INDEX, CREATE TEMPORARY TABLES, DELETE, UPDATE, EXECUTE ON cnccdb .*
      TO'cnccusr'@'%';
      ```

2.2.1.6.3 Configuring M-CNCC Core Database

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**
>
> 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 ndbappmysqld-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'@'%';
```

2.2.1.7 Configuring Kubernetes Secret

This section explains how to configure Kubernetes secrets for accessing the database and the M-CNCC IAM default user.

2.2.1.7.1 Configuring Kubernetes Secret for Accessing Database

This section explains how to configure Kubernetes secrets for accessing the database.

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

```
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>
```

> **Note**
>
> You must use a strong non-predictable password consisting of complex strings of more than 10 mixed characters.

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
```

2.2.1.7.2 Configuring Secret for the Admin User in M-CNCC IAM

To create and update 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
```

> **Note**
>
> This Command is just for reference. You must use a strong non-predictable password consisting of complex strings of more than 10 mixed characters.

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
   ```

2.2.1.8 Configuring Secrets for Enabling HTTPS

This section explains the steps to configure HTTPS at Ingress Gateway.

This step is optional. It is required only when SSL settings need to be enabled on Ingress Gateway microservices of CNC Console.

The passwords for TrustStore and KeyStore are stored in respective password files mentioned below.To create Kubernetes secret for HTTPS, following files are required:

- ECDSA private key and CA signed certificate of CNC Console, if initialAlgorithm is ES256
- RSA private key and CA signed certificate of CNC Console, if initialAlgorithm is RS256
- 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-----
  ```

> **Note**
>
> Creation process for private keys, certificates and passwords is at the discretion of the user or operator.

> **Note**
>
> For more information on how to enable HTTPS, see **CNC Console Instances Configuration Examples**

2.2.1.8.1 Configuring Secret to Enable HTTPS in M-CNCC IAM

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

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>
```

> **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-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**
>
> The names used in the above command must be as same as the names provided in the custom_values.yaml in CNC Console deployment.

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.

2.2.1.8.2 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.

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>
```

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

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

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.

2.2.1.8.3 Configuring Secret to Enable HTTPS in A-CNCC Core

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

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>
```

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

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

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.

2.2.1.9 Configuring CNC Console to support Aspen Service Mesh

2.2.1.9.1 Introduction

CNCC leverages the Platform Service Mesh (for example, Aspen Service Mesh (ASM)) for all internal and external TLS communication by deploying a special sidecar proxy in each pod to intercept all the network communications. The service mesh integration provides inter-NF communication and allows API gateway to co-work with service mesh. The service mesh integration supports the services by deploying a special sidecar proxy in each pods to intercept all the network communications between microservices.

Supported ASM version: 1.14.x

For ASM installation and configuration, see official Aspen Service Mesh website for details.

Aspen Service Mesh (ASM) configurations are categorized as follows:

- **Control Plane**: It involves adding labels or annotations to inject sidecar. The control plane configurations are part of the NF Helm chart.

- **Data Plane**: It helps in traffic management, such as handling NF call flows by adding Service Entries (SE), Destination Rules (DR), Envoy Filters (EF), and other resource changes such as apiVersion change between different versions. This configuration is done manually by considering each NF requirement and ASM deployment.

Data Plane Configuration

Data Plane configuration consists of following Custom Resource Definitions (CRDs):

- Service Entry (SE)
- Destination Rule (DR)
- Envoy Filter (EF)

> **Note**
>
> Use Helm charts to add or remove CRDs that you may require due to ASM upgrades to configure features across different releases.

The data plane configuration is applicable in the following scenarios:

- **NF to NF Communication**: During NF to NF communication, where sidecar is injected on both the NFs, SE and DR must communicate with the corresponding SE and DR of the other NF. Otherwise, the sidecar rejects the communication. All egress communications of NFs must have a configured entry for SE and DR.

  > **Note**
  >
  > Configure the core DNS with the producer NF endpoint to enable the sidecar access for establishing communication between cluster.

- **Kube-api-server**: For Kube-api-server, there are a few NFs that require access to the Kubernetes API server. The ASM proxy (mTLS enabled) may block this. As per F5 recommendation, the NF must add SE for the Kubernetes API server for its own namespace.
- **Envoy Filter**s: Sidecars rewrite the header with its own default value. Therefore, the headers from back-end services are lost. So, you need Envoy Filters to help in passing the headers from back-end services to use it as it is.

> **Note**
>
> For ASM installation and configuration, refer Official Aspen Service Mesh website for details.

2.2.1.9.2 Predeployment Configuration

Following are the prerequisites to install CNCC with support for ASM:

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
   ```

Update Default Kyverno Policy Restriction to add CNC Console Namespace

> **Note**
>
> You need admin privileges to edit/patch the **clusterpolicies** that are mentioned in following steps.

From 23.2.0 CNE, we have support for Kyverno policies. The default Kyerno policy restrictions don't allow istio-proxy containers. To allow running of istio-proxy containers on CNC Console with ASM deployment, you must perform the following kubectl operation on the namespace where the CNC Console is installed.

```
$ kubectl patch clusterpolicy disallow-capabilities --type=json \
  -p='[{"op": "add", "path": "/spec/rules/0/exclude/any/0/resources/
namespaces/-", "value": "<cncc_namespace>" }]'
```

For example,

```
$ kubectl patch clusterpolicy disallow-capabilities --type=json \
  -p='[{"op": "add", "path": "/spec/rules/0/exclude/any/0/resources/
namespaces/-", "value": "cncc" }]'
```

Establish connectivity to services that are not part of Service Mesh Registry

> **Note**
>
> This is an optional step. Depending on the underlying Service Mesh deployment, Service Entry and Destination Rule may be required for CNC Console to connect to other kubernetes microservices that are not a part of Service Mesh Registry. The kubernetes microservices may be DB service, NF services, or Common services.

You can use the following sample service entry and destination rule template:

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: <Unique ServiceEntry Name for Service>
  namespace: <CNCC-NAMESPACE>
spec:
  exportTo:
  - "."
```

```
  hosts:
  - <Service-public-FQDN>
  ports:
  - number: <Service-public-PORT>
    name: <Service-PORTNAME>
    protocol: <Service-PROTOCOL>
  location: MESH_EXTERNAL
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: <Unique DestinationRule Name for Service>
  namespace: <CNCC-NAMESPACE>
spec:
  exportTo:
  - "."
  host: <Service-public-FQDN>
  trafficPolicy:
    tls:
      mode: DISABLE
```

If kubernetes services only have IP and does not have public FQDN, then first create the necessary headless service and then create Service Entry and Destination Rule. Here is a sample headless service template.

```
apiVersion: v1
kind: Endpoints
metadata:
  name: <Unique Endpoint Name for Service>
  namespace: <CNCC_NAMESPACE>
subsets:
- addresses:
  - ip: <Service-public-IP>
  ports:
  - port: <Service-public-PORT>
    protocol: <Service-PROTOCOL>
---
apiVersion: v1
kind: Service
metadata:
  name: <Unique Endpoint Name for Service>-headless
  namespace: <CNCC_NAMESPACE>
spec:
  clusterIP: None
  ports:
  - port: <Service-public-PORT>
    protocol: <Service-PROTOCOL>
    targetPort: <Service-public-PORT>
  sessionAffinity: None
  type: ClusterIP
```

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.
>
> ```
> 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
```

2.2.1.9.3 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 the following sidecar resource configuration annotations:

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

```
  customExtension:
    allResources:
      labels: {}
      annotations:
        sidecar.istio.io/proxyMemory: "1Gi"
        sidecar.istio.io/proxyMemoryLimit: "1Gi"
        sidecar.istio.io/proxyCPU: "1"
        sidecar.istio.io/proxyCPULimit: "1"

  # ********  Sub-Section End: Common Global Parameters ******************
  # ************************************************************************
```

2. Add rewriteAppHTTPProbers annotation to make health checks on services with mTLS enforcement on work:

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

    nonlbStatefulSets:
      labels: {}
      annotations:
        sidecar.istio.io/rewriteAppHTTPProbers: "true"

  # ********  Sub-Section End: Common Global Parameters ******************
  # ************************************************************************
```

3. Provide the service account name:

```
global:
  # *****  Sub-Section Start: Ingress Gateway Global Parameters *****
  serviceAccountName: &serviceAccountName <cncc-serviceaccount-name>
```

4. 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
```

5. If ASM is deployed with mTLS disabled, then set **serviceMeshHttpsEnabled** flag to *false*:

```
global:
  serviceMeshHttpsEnabled: false
```

2.2.1.9.4 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 *******
```

2.2.1.10 Configuring Network Policies for CNC Console

Overview

Kubernetes network policies allow you to define ingress or egress rules based on Kubernetes resources such as Pod, Namespace, IP, and Port. These rules are selected based on Kubernetes labels in the application. These network policies enforce access restrictions for all the applicable data flows except communication from Kubernetes node to pod for invoking container probe.

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

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

> **Note**
>
> - If the traffic is blocked or unblocked between the pods even after applying network policies, check if any existing policy is impacting the same pod or set of pods that might alter the overall cumulative behavior.
> - If changing default ports of services such as Prometheus, Database, Jaegar, or if Ingress or Egress Gateway names are overridden, update them in the corresponding network policies.

Configuring Network Policies

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

Installing Network Policies

Prerequisite

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

Note:

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

1. Open the *occncc_network_policy_custom_values_<version>.yaml* file provided in the release package zip file. For downloading the file, see Downloading CNC Console Package and Pushing the Images to Customer Docker Registry.
2. Update the *occncc_network_policy_custom_values_<version>.yaml* file as per the requirement. For more information on the parameters, see the Configuration Parameters for network policy parameter table.
3. Run the following command to install the network policies:

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

   For Example:

   ```
   helm install occncc-network-policy -f
   occncc_network_policy_custom_values_23.4.4.yaml --namespace cncc occncc-
   network-policy-23.4.4.tgz
   ```

   Where,

   - `helm-release-name`: occncc-network-policy helm release name.
   - `custom-value-file`: occncc-network-policy custom value file.
   - `namespace`: CNC Console namespace.
   - `network-policy`: network-policy package.

> **Note**
>
> - Connections that were created before installing network policy and still persist are not impacted by the new network policy. Only the new connections would be impacted.
> - If you are using ATS suite along with network policies, it is required to install the <NF acronym> and ATS in the same namespace.

Upgrading Network Policies

1. Modify the *occncc_network_policy_custom_values_<version>.yaml* file to update/add/delete the network policy.
2. Run the following command to upgrade the network policies:

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

For Example:

```
helm upgrade occncc-network-policy -f
occncc_network_policy_custom_values_23.4.4.yaml --namespace cncc occncc-
network-policy-23.4.4.tgz
```

Verifying Network Policies

Run the following command to verify that the network policies have been applied successfully:

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

For Example:

```
kubectl get networkpolicy -n cncc
```

Uninstalling Network Policies

Run the following command to uninstall the network policies:

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

For Example:

```
$ helm uninstall occncc-network-policy --namespace cncc
```

> **Note**
>
> Uninstall removes all the network policies.

Configuration Parameters for Network Policies

Table 2-13 Supported Kubernetes Resource for Configuring Network Policies

Parameter	Description	Default Value	Value Range	Mandatory (M)/ Optional (O)/ Conditional (C)	Notes
apiVersion	This indicates the Kubernetes version for access control.	networking.k8s.io/v1	NA	M	This is the supported api version for network policy. This is a read-only parameter.
kind	This represents the REST resource this object represents.	NetworkPolicy	NA	M	This is a read-only parameter.

Configuration Parameters for Network Policies

Table 2-14 Configuration Parameters for Network Policies

Parameter	Description	Default Value	Value Range	Mandatory (M)/ Optional (O)/ Conditional (C)
metadata.name	This indicates a unique name for the network policy.	{{ .metadata.name } }	NA	M
spec.{}	This consists of all the information needed to define a particular network policy in the given namespace.	NA	NA	M

For more information about this functionality, see Network Policies in the *Oracle Communications Cloud Native Configuration Console User Guide*

2.2.1.11 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 2-15 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
```

2.2.1.11.1 M-CNCC IAM Predeployment Configuration

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

2.2.1.11.1.1 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
```

2.2.1.11.2 A-CNCC Core Predeployment Configuration

Following are the predeployment configuration procedures:

2.2.1.11.2.1 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.

Configuring M-CNCC IAM to Enable Additional Settings

CNC Console provides an option to enable additional settings in M-CNCC IAM by setting below mentioned flag to true in *occncc_custom_values_<version>.yaml* file.

The additional settings include some of the configuration settings such as authentication settings to configure password policies.

```
cncc-iam:
  global:
    iamSettingEnabled: false
```

2.2.2 Installation Tasks

This section explains how to install CNC Console. To install CNC Console using CDCS, see *Oracle Communications Cloud Native Core CD Control Server User Guide.*

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

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

CNC Console 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 scripts directory present in the `occncc_csar_<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 multicluster 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 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 services in absence of a Manager in the same cluster.
> - Agent is needed only when NFs are present on the cluster.

2.2.2.1 Installing CNC Console Package

This section provides the installation procedure 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      23.4.4         23.4.4     A Helm chart for CNC
   Console
   ```

2. Customize the `occncc_custom_values_<version>.yaml`file with the required deployment parameters. For customizing custom-cncc_values.yaml file, see CNC Console Configuration Parameters section.

   > **Note**
   >
   > For IPv4 or IPv6 configurations, see Support for IPv4 or IPV6 Configuration for CNC Console.

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: '{"http-tls":"TCP"}'` is required in global section of *occncc_custom_values_<version>.yaml* file when CNC Console is deployed with HTTPs enabled in CNE.

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

Note

The annotation `oracle.com.cnc/egress-network: oam` is required under global section if Ldap/Ldaps is integrated with console.

```
nonlbStatefulSets:
  labels: {}
    annotations:
      oracle.com.cnc/egress-network: oam
```

> **Note**
>
> The annotation `oracle.com.cnc/egress-network: oam` is required under global section if isMultiClusterDeployment flag is enabled for console.
>
> ```
> customExtension:
>     lbDeployments:
>       labels: {}
>       annotations:
>         oracle.com.cnc/egress-network: oam
> ```

> **Note**
>
> CNC Console IAM deployment has following Pod Security context and Container Security Context:
> Pod Security context: `PSC: map[fsGroup:1000]`
>
> Container Security Context: `runAsUser:1000 runAsNonRoot:true`

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

a. Update unique CNC Console 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. You must 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
```

> **Note**
>
> CNC Console only supports prometheus apiprefix and not promxy.

b. **Multicluster 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**
>
> CNCC helm install command appears hung for a while because Kubernetes job run by 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_<version>.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 user configuration

Example:

```
helm install cncc ocscp-helm-repo/cncc -f
occncc_custom_values_23.4.4.yaml --namespace cncc --version 23.4.4
```

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

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

Example:

```
helm install cncc -f occncc_custom_values_23.4.4.yaml --namespace cncc
occncc-23.4.4.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.

To upgrade:

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

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

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

Example:

```
$ helm upgrade cncc ocspf-helm-repo/cncc -f
occncc_custom_values_23.4.4.yaml --namespace cncc
```

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

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

Example:

```
helm  upgrade cncc -f occncc_custom_values_23.4.4.yaml --namespace cncc
occncc-23.4.4.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-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-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. Do not press "ctrl+c" to exit from helm install command. It may lead to anomalous behavior.

> **Note**
>
> **timeout duration (optional)**: If not specified, the default value will be 5m (5 minutes) in helm. 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.

2.2.3 Postinstallation Tasks

This section explains the postinstallation tasks for CNC Console.

> **Note**
>
> For IPv4 or IPv6 configurations, see Support for IPv4 or IPV6 Configuration for CNC Console.

2.2.3.1 Verifying CNC Console Installation

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

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

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

   For example:

   ```
   occncc -n cncc
   ```

   Status should be `deployed`

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

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

   For example:

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

   If the deployment is successful, the status for all pods changes to **Running** and **Ready**.

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

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

   For example:

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

> **Note**
>
> Take a backup of the following files, which are required during fault recovery:
>
> - Current custom-values.yaml file from which you are upgrading.
> - 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 Configuration ConsoleTroubleshooting Guide.*
> - For information on validation-hook errors, see CNC Console Validation-hook Error Codes.

2.2.3.2 Performing Helm Test

Helm Test is a feature which validates the successful installation of CNCC 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**
>
> Helm Test can be performed only on Helm3.

To perform Helm test, perform the following steps:

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: 23.4.4
      imagePullPolicy: IfNotPresent
    config:
      logLevel: WARN
      timeout: 240
```

2. Run the following command to perform the Helm test:

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

Where,

<release_name> is the release name.
<namspace> is the deployment namespace where CNCC is installed.

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
```

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

If the Helm test fails, see the *Oracle Communications Cloud Native Configuration Console Troubleshooting Guide.*

2.2.3.2.1 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: 23.4.4
    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"
  }
}
```

2.2.3.2.2 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
```

2.2.3.3 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 in to 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/*
   ```

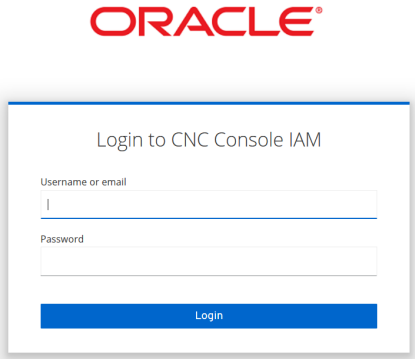
   External LB-IP and ServicePort
   Example:

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

DNS Resolvable FQDN and ServicePort
Example:

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

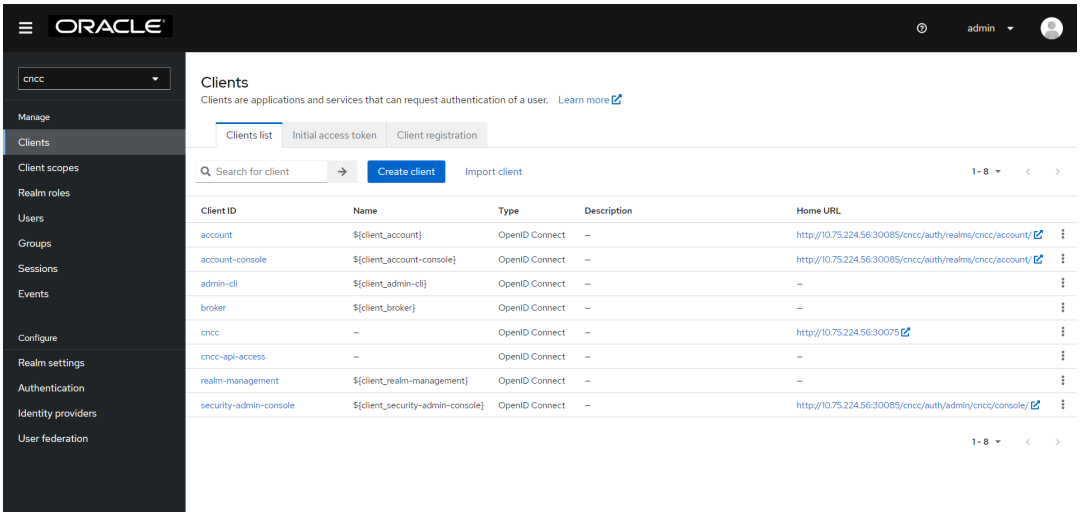
Figure 2-1 Login



> **Note**
>
> You must select CNCC from the Realm drop down on the left pane after logging in to CNC Console IAM.

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

Figure 2-2 Clients Tab

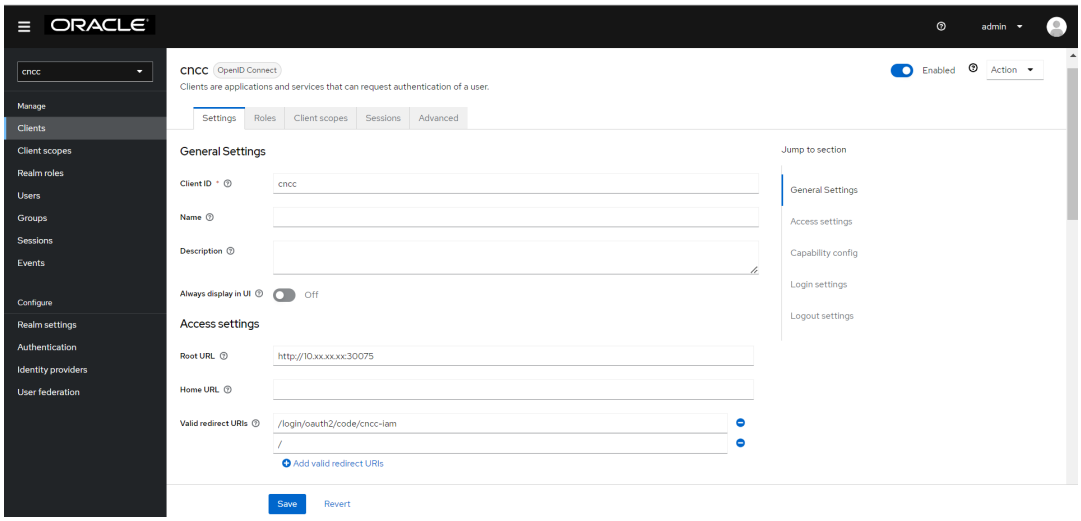


2. Enter CNCC Core Ingress URI in the **Root URIs** field and **Save.**

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

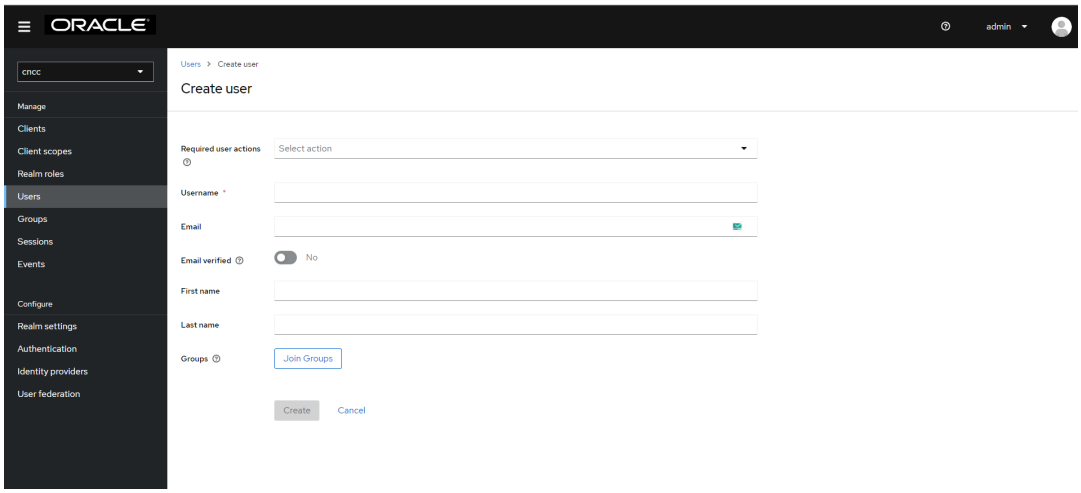
> **Note**
>
> **Redirection URL** is pre-populated, only root url needs to be configured as part of postinstallation procedure

Figure 2-3 General Settings



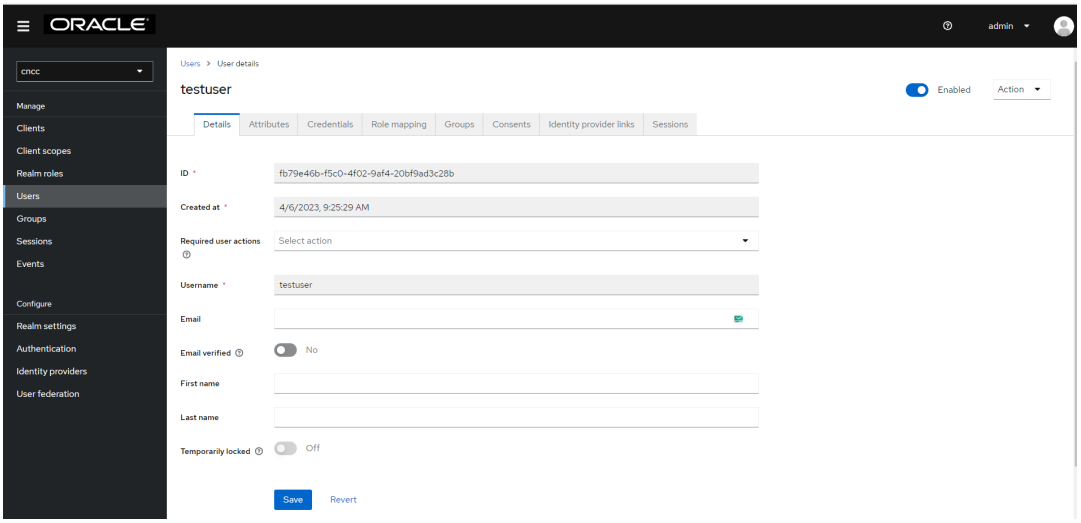
3. Click **Manage**, click **Users,** and click **Add user** on the right pane.

Figure 2-4 Add User



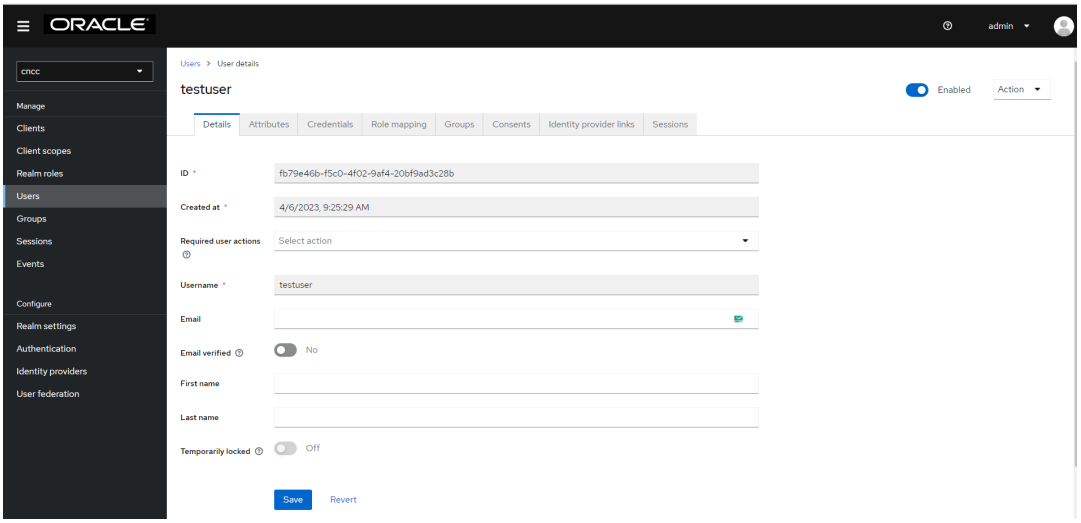
4. **Add user** screen appears. Add the user details and click **Save**.

Figure 2-5 Save User



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

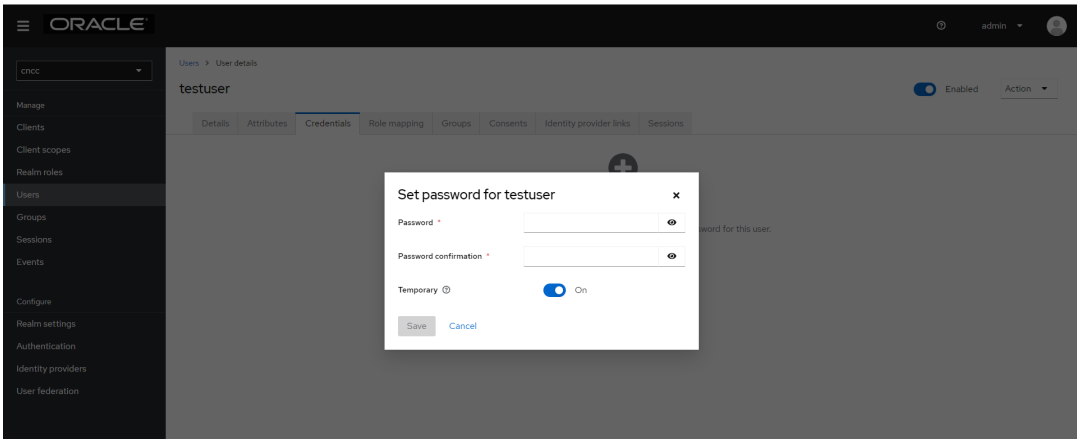
Figure 2-6 User Details



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

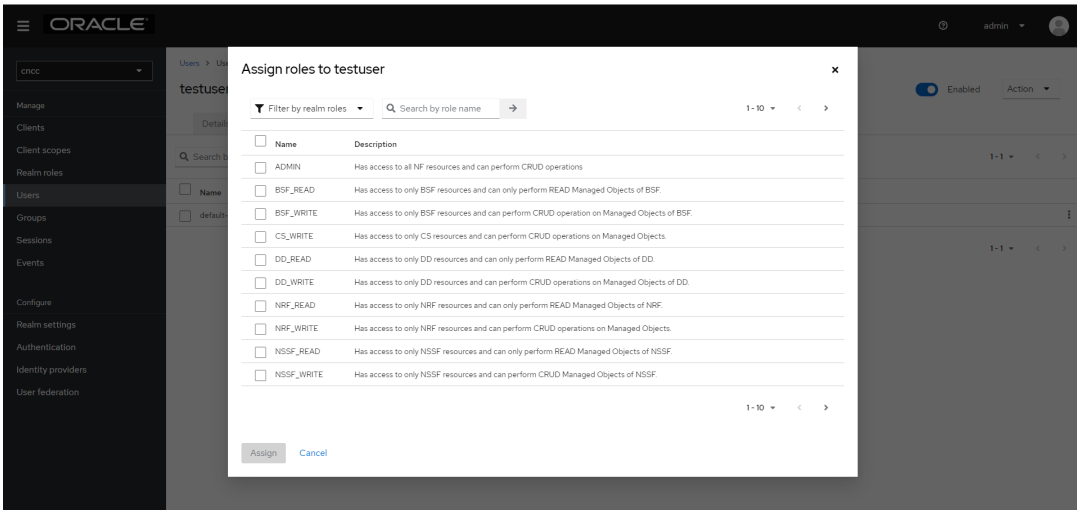
> **Note**
>
> Setting **Temporary flag** as **ON** prompts the user to change the password when logging in to the CNCC Core GUI for the first time.

Figure 2-7 Set Password



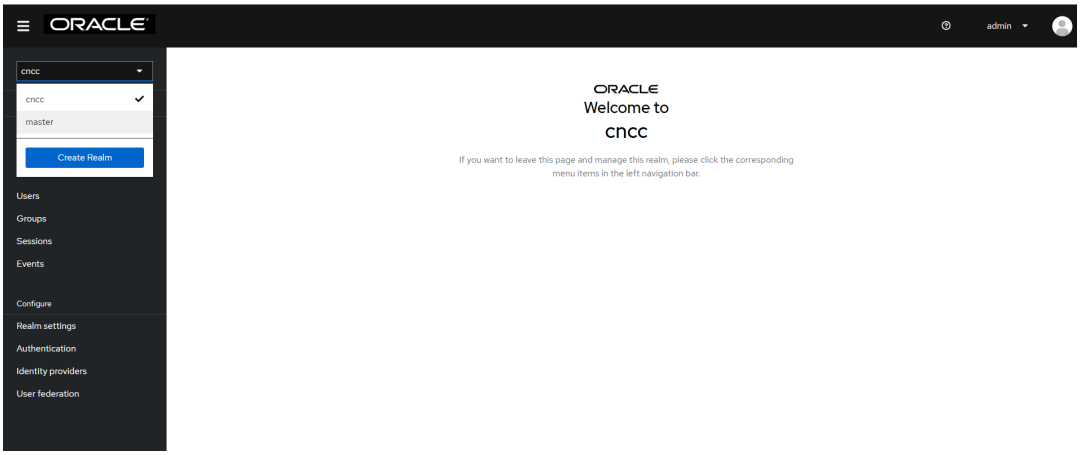
7. Navigate to the **Role Mappings** tab and assign roles to the user.

Figure 2-8 Role Mappings



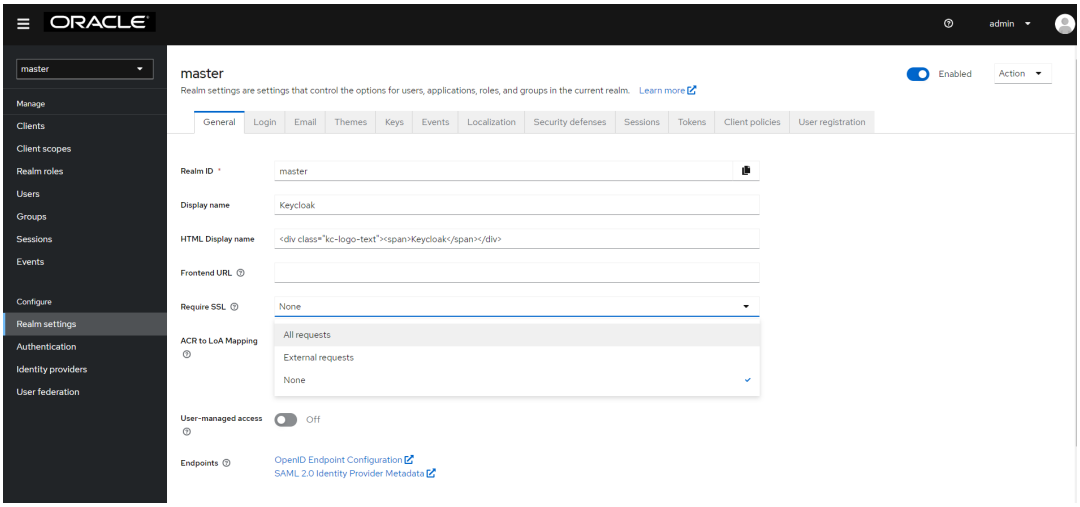
8. Select **Master Realm** from the Realm drop down list.

Figure 2-9 Master Realm



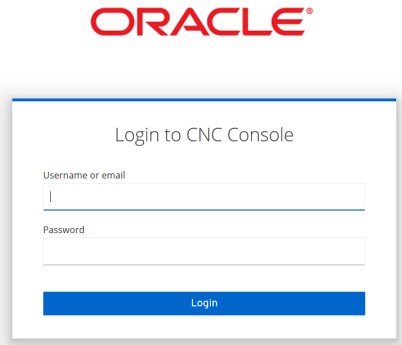
9. From the **Realm Setting**s tab navigate to the **General** tab. Set Require SSL to **All Requests** and click **Save**.

Figure 2-10 Realm Settings



10. Log in to CNCC Core using the credentials of the user created earlier.

Figure 2-11 CNC Console Core login



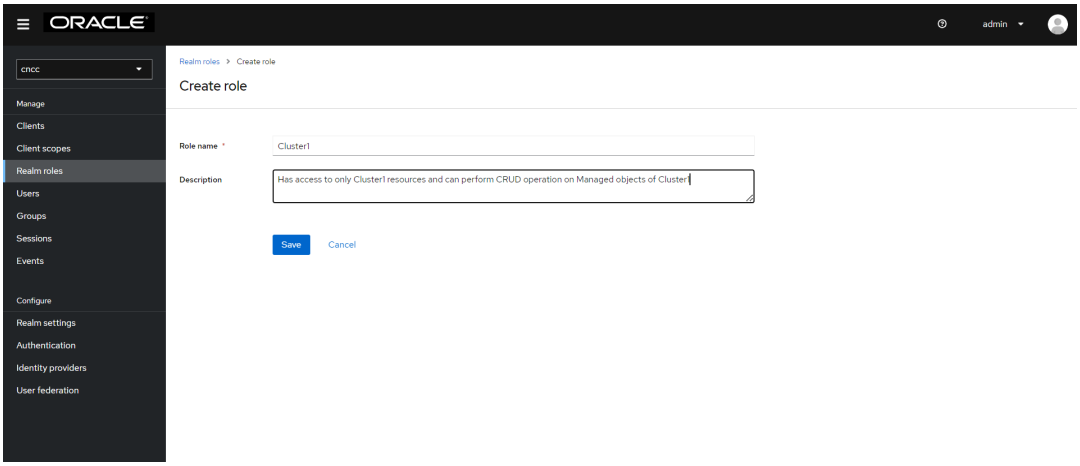
2.2.3.3.1 CNC Console Multicluster Deployment Roles

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

This section explains the steps to create the Roles.

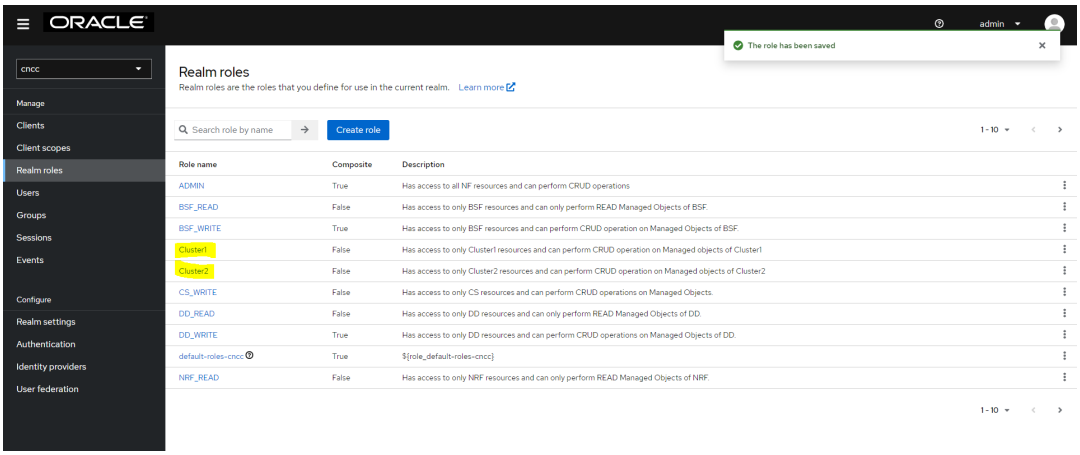
1. Log in to M-CNCC IAM and click **Realm Roles** present on the left pane. The roles defined in the realm is displayed on the right pane.

Figure 2-12 Realm Roles



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

Figure 2-13 Create Role



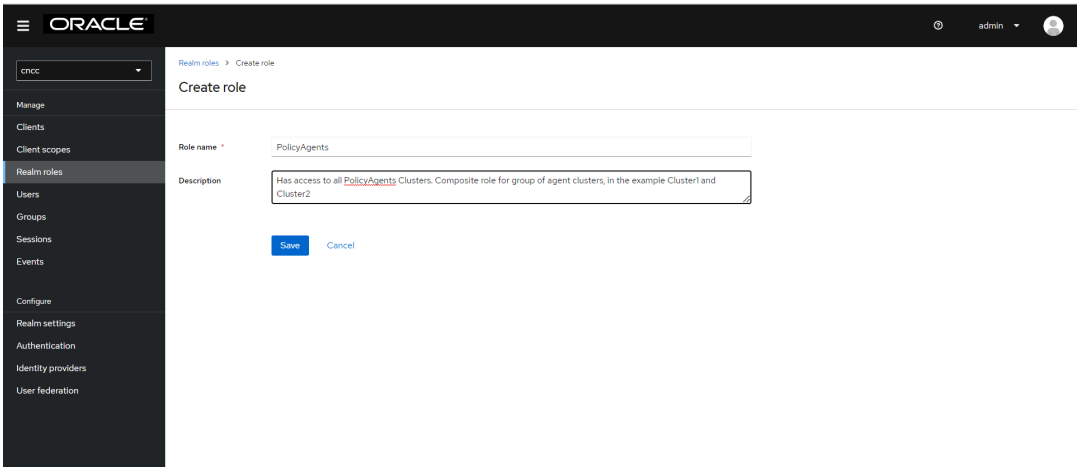
> **Note**
>
> The user must ensure that the cluster role name must match with role name given in helm configuration.
>
> - For M-CNCC cluster role creation in M-CNCC IAM, the value of `global.mCnccCores.id` or `global.mCnccCores.role name` must be used
> - For A-CNCC cluster role creation in M-CNCC 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.

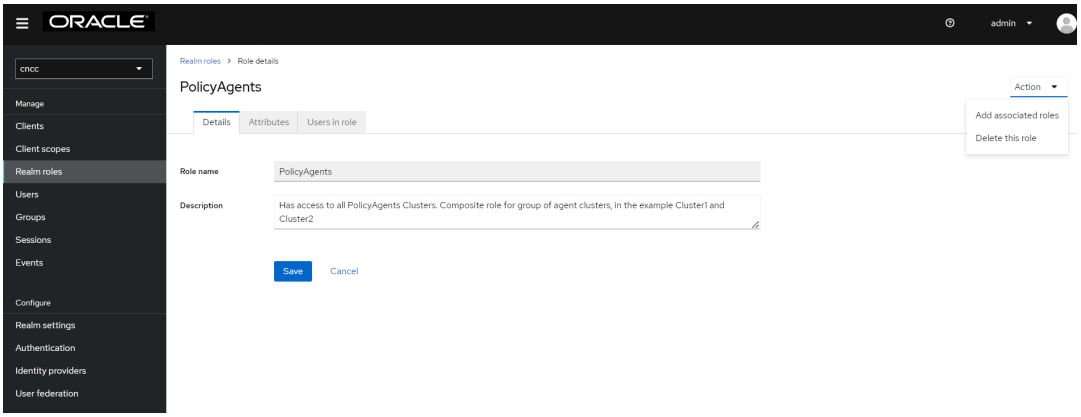
1. Click **Create Role**, the **Create Role** screen appears. Add the **Role Name** and click **Save**.

Figure 2-14 Add Role Name



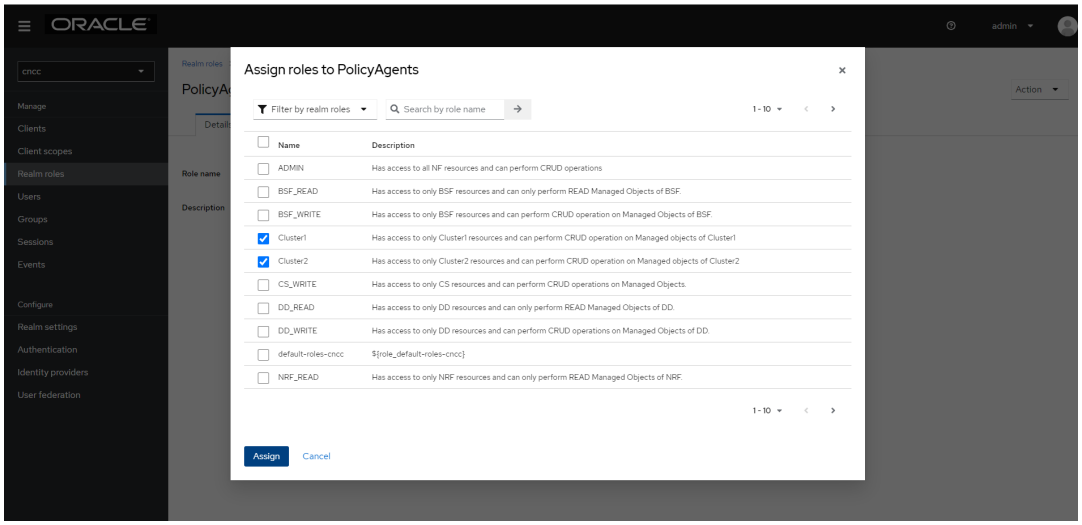
2. From the **Action** drop down, select **Add associated roles.**

Figure 2-15 Assign Role



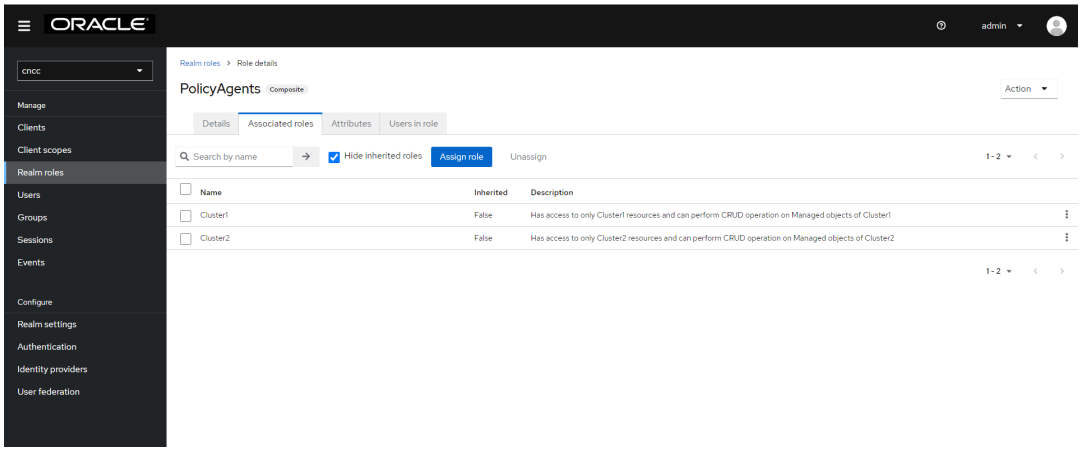
3. This enables the **Composite Roles** functionality and the **Assign roles** screen appears.
4. Select the required roles from **Realm Roles** and click **Assign**.

Figure 2-16 Assign Roles



5. The **Associated roles** tab appears in the **Role details** screen.

Figure 2-17 Associates Roles



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

> **Note**
>
> For more information about the Roles, see the **Role Based Access Control in CNC Console** section in *Oracle Communications Cloud Native Configuration Console User Guide*.

3
Customizing CNC Console

This chapter provides information about customizing CNC Console.

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 the CNC Console package file: Unzip the CNC Console package file to the specific repository:

   `unzip occncc_csar_<marketing-release-number>.zip`
   The package consists of following files and folders:

   a. **Files**: CNC Console Docker images file and CNC Console Helm Charts

   b. **Scripts**: Custom values and alert files:

   - CNC Console custom values file: `occncc_custom_values_<version>.yaml`
   - CNC Console cnDBTier custom values file: `occncc_dbtier_<cndbtier_version>_custom_values_<version>.yaml`
   - CNC Console network policy custom values file: `occncc_network_policy_custom_values_<version>.yaml`
   - CNC Console IAM Schema file for rollback to previous version: `occncc_rollback_iam_schema_<version>.sql`
   - CNC Console Metric Dashboard file: `occncc_metric_dashboard_<version>.json`
   - CNC Console Metric Dashboard file for CNE supporting Prometheus HA (CNE 1.9.x onwards): `occncc_metric_dashboard_promha_<version>.json`
   - CNC Console Alert Rules file: `occncc_alertrules_<version>.yaml`
   - CNC Console Alert Rules file for CNE supporting Prometheus HA (CNE 1.9.x onwards): `occncc_metric_dashboard_promha_<version>.json`
   - CNC Console MIB files: `occncc_mib_<version>.mib`, `occncc_mib_tc_<version>.mib`
   - CNC Top level mib file: `toplevel.mib`

   c. **Definitions**: Definitions folder contains the CNE Compatibility and definition files.

   - `occncc_cne_compatibility.yaml`
   - `occncc.yaml`

   d. TOSCA-Metadata: `TOSCA.meta`

   e. The package folder structure is as follows:

```
Definitions
       occncc_cne_compatibility.yaml
       occncc.yaml
Files
    apigw-common-config-hook-23.4.4.tar
```

```
        apigw-configurationinit-23.4.4.tar
        ChangeLog.txt
        cncc-apigateway-23.4.4.tar
        cncc-cmservice-23.4.4.tar
        cncc-core-validationhook-23.4.4.tar
        cncc-iam-23.4.4.tar
        cncc-iam-healthcheck-23.4.4.tar
        cncc-iam-hook-23.4.4.tar
        Helm
            occncc-23.4.4.tgz
            occncc-network-policy-23.4.4.tgz
        Licenses
        nf_test-23.4.3.tar
        ocdebug-23.4.3.tar
        Oracle.cert
        Tests
    occncc.mf
    Scripts
        occncc_alerting_rules_promha_23.4.4.yaml
        occncc_alertrules_23.4.4.yaml
        occncc_custom_values_23.4.4.yaml
        occncc_dbtier_23.3.0_custom_values_23.4.4.yaml
        occncc_metric_dashboard_23.4.4.json
        occncc_metric_dashboard_promha_23.4.4.json
        occncc_mib_23.4.4.mib
        occncc_mib_tc_23.4.4.mib
        occncc_network_policy_custom_values_23.4.4.yaml
        occncc_rollback_iam_schema_23.4.4.sql
        toplevel.mib
    TOSCA-Metadata
        TOSCA.meta
```

For example,

```
Example: unzip occncc_csar_23_4_4_0_0.zip
Archive: occncc_csar_23_4_4_0_0.zip
inflating: Definitions/occncc_cne_compatibility.yaml
inflating: Definitions/occncc.yaml
creating: Files/
creating: Files/Tests/
creating: Files/Helm/
inflating: Files/Helm/occncc-23.4.4.tgz
extracting: Files/Helm/occncc-network-policy-23.4.4.tgz
creating: Files/Licenses/
inflating: Files/apigw-configurationinit--23.4.10.tar
inflating: Files/apigw-common-config-hook--23.4.10.tar
inflating: Files/ocdebug-tools-23.4.4.tar
inflating: Files/cncc-apigateway--23.4.10.tar
inflating: Files/cncc-iam-23.4.4.tar
inflating: Files/cncc-iam-hook-23.4.4.tar
inflating: Files/cncc-iam-healthcheck-23.4.4.tar
inflating: Files/nf_test-23.4.4.tar
inflating: Files/cncc-core-validationhook-23.4.4.tar
inflating: Files/cncc-cmservice-23.4.4.tar
inflating: Files/ChangeLog.txt
extracting: Files/Oracle.cert
```

```
creating: Scripts/
inflating: Scripts/occncc_custom_values_23.4.4.yaml
inflating: Scripts/occncc_network_policy_custom_values_23.4.4.yaml
inflating: Scripts/occncc_mib_tc_23.4.4.mib
inflating: Scripts/occncc_mib_23.4.4.mib
inflating: Scripts/toplevel.mib
inflating: Scripts/occncc_metric_dashboard_23.4.4.json
inflating: Scripts/occncc_metric_dashboard_promha_23.4.4.json
inflating: Scripts/occncc_alertrules_23.4.4.yaml
inflating: Scripts/occncc_alerting_rules_promha_23.4.4.yaml
inflating: Scripts/occncc_rollback_iam_schema_23.4.4.sql
inflating: Scripts/ occncc_dbtier_23.4.4_custom_values_23.4.4.yaml
creating: TOSCA-Metadata/
inflating: TOSCA-Metadata/TOSCA.meta
inflating: occncc.mf
```

1. Customize the appropriate files.
2. Save the updated files.

3.1 Global Configuration Options

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

Table 3-1 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.

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.serviceAccountName`	This is an optional parameter. Name of service account. If this field is kept empty then a default service account '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.
`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.

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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.
`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.
`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.

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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
`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

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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
`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.

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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.containers LogStorage **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.containers CrititcalStorage
`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.
`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:**

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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.k8sResources.pdb.supportedVersions`	This is an optional parameter. List of supported Kubernetes apiVersion for PodDisruptionBudget out of which the priority is given to the latest.	**DataType:** List String **Range:**It takes resources and its version in the form of <resource_name>/<max_version_supportedbyNF> **Default Value:** policy/v1
`global.extraContainers`	This is a mandatory parameter. To enable or disable the debug tools container	**DataType:** enum **Range:** DISABLED, ENABLED **Default Value:**
`global.debugToolContainerMemoryLimit`	This is a mandatory parameter This field describes the memory size (request and limit) and emptyDir volume size for the Debug-tool container.	**DataType:** String **Range:** Valid Integer value followed by Mi/Gi **Default Value:** 4Gi
`global.extraContainersVolumesTpl.name`	This is a mandatory parameter Name of the emptyDir volume.	**DataType:** String **Range:** 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:** debug-tools-dir
`global.extraContainersVolumesTpl.emptyDir.medium`	This is a mandatory parameter Describes where emptyDir volumes are stored	**DataType:** String **Range:** **Default Value:** Memory
`global.extraContainersVolumesTpl.emptyDir.sizeLimit`	This is a mandatory parameter The size of the emptyDir volume.	**DataType:** String **Range:** Valid Integer value followed by Mi/Gi **Default Value:** 4Gi

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

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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
`global.extraContainersTpl.name`	This is a mandatory parameter. Name of the container	**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.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:**512Mi
`global.extraContainersTpl.resources.limits.cpu`	The maximum amount of cpu allowed	**DataType:** String **Range:** Valid floating point value between 0 and 1 **Default Value:**0.5
`global.extraContainersTpl.resources.limits.memory`	This is a mandatory parameter. The maximum amount of memory allowed	**DataType:** String **Range:** Valid Integer value followed by Mi/Gi etc. Values will be set by global.debugToolContainerMemoryLimit **Default Value:**4Gi
`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:** 512Mi
`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. Values will be set by global.debugToolContainerMemoryLimit **Default Value:**4Gi

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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

`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.runAsUser`	This is an optional parameter. The UID to run the entrypoint of the container process.	**DataType:** Integer **Range:**Integer Value
`global.extraContainersTpl.volumeMounts.mountPath`	This is a mandatory parameter. Path where the emptyDir volume has to be mounted inside the debug-tool container.	**DataType:** String **Range:**Valid path **Default Value:** /tmp/tools
`global.extraContainersTpl.volumeMounts.name`	This is a mandatory parameter.	**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:** debug-tools-dir
`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

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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
`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.useClusterIpForLbServices`	This is an optional parameter. Set this flag to true to assign ClusterIP service type for LoadBalancer (LB) services. This flag is required for deployment, that manages their LB services as ClusterIP alternatively. LoadBalancer in such cases is generally rely on service annotation to discover LB services.	**DataType:**Boolean **Range:** True or False **Default Value:** False

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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.iamUserName`	This is a mandatory parameter. It is the name of cncc-iam user as given by a user. Example: admin	**DataType:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. **Default Value:** Admin
`global.isMultiClusterDeployment`	This is an optional parameter. This flag indicates single cluster deployment or multicluster deployment. By default single cluster deployment is configured, flag must be set to true for multicluster deployment.	**DataType:** Boolean **Range:** True or False **Default Value:** False
`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 multicluster 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 multicluster deployment	**DataType:** Boolean **Range:** True or False **Default Value:** True

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`global.self.cnccId`	This is a mandatory parameter. It is the ID of deployment in multicluster 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, see CNCConsole Multi Cluster and Multi Instance Configurations.
		DataType: List[String]
		Range:
		Example:
		`drop:`
		`-all`
		Default Value:

`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, CNC Console 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, CNC Console 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:**

Table 3-1 (Cont.) Global Configuration Options

Parameter	Description	Details
`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:**
`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 3-1 (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:**

3.2 CNC Console IAM Configuration Parameters

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

3.2.1 Global Parameters

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

Table 3-2 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.iamSettingEnabled`	This is an optional parameter. It enables additional settings for M-CNCC IAM.	**DataType:**Boolean **Range**: True or False **Default Value:** False
`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

Table 3-2 (Cont.) Global Parameters

Parameter	Description	Details
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
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 3-2 (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
`cncc-iam.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:**NA **Default Value:** NA
`cncc-iam.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 key will be picked)	**DataType:**String **Range:**NA **Default Value:** NA

3.2.2 IAM Backend Parameters

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

Table 3-3 IAM Backend Parameters

Parameter	Description	Details
`cncc-iam.kc.preferIpv6Stack.enabled`	This is an optional parameter. The flag to enable or disable CNC Console deployment on an IPv6 setup.	**DataType:**Boolean **Range:**True or False **Default Value:** False
`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

Table 3-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
`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
`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

Table 3-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
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
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

Table 3-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
`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
`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

Table 3-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
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:**
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

Table 3-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
`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>
`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

Table 3-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
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
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.

Table 3-3 (Cont.) IAM Backend Parameters

Parameter	Description	Details
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.
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.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:** NA
cncc-iam.kc.keycloak.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 key will be picked)	**DataType:**String **Range:** **Default Value:** NA

3.2.3 Ingress Gateway Parameters

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

Table 3-4 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

Table 3-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
`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>
`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.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
`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

Table 3-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
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
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

Table 3-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
`cncc-iam.ingress-gateway.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
`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:**{}
`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

Table 3-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
`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
`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

Table 3-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
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
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.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

Table 3-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
`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.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
`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

Table 3-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
`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
`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_WIT H_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_AE S_256_GCM_ SHA384 TLS_ECDHE_ ECDSA_WIT H_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 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:**

Table 3-4 (Cont.) Ingress Gateway Parameters

Parameter	Description	Details
`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
`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:**

3.3 M-CNCC Core Configuration Options

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

> **Note**
>
> The options mentioned in the following tables can be read as *mcncc-core.<field Name>*.
>
> Example: The *global.dbName* can be read as *mcncc-core.global.dbName* or *acncc-core.global.dbName*.

3.3.1 Global Parameters

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

Table 3-5 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.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 aserviceAccountNamend 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:**
`global.dbName`	This is a mandatory parameter. It is the name of the database used for M-CNCC Core user should create DB with unique name before deploying M-CNCC Core	**DataType:**String **Range:**Valid string **Default Value:** • mcncccommonconfig in case of mcncc-core
`global.httpsEnabled`	This is an optional parameter. To Initialize SSL related infrastructure in init 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

Table 3-5 (Cont.) Global Parameters

Parameter	Description	Details
`global.enableIncomingHttps`	This is an optional parameter. Server Configuration for http and https support.	**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

3.3.2 Core Backend Parameters

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

Table 3-6 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 tag 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:** It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy **Default Value:** IfNotPresent

Table 3-6 (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:** Values will be set by global.secretName Default value is cncc-db-secret **Default Value:**
`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

Table 3-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
`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
`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

Table 3-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
`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: 23.4.4	**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>
`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.nodeSelector.nodeKey`	This is an optional parameter. Node Selector Key. (note this will be looked first and then if not present global node key will be picked).	**DataType:**Boolean **Range:**True or False **Default Value:**False
`cmservice.deployment.nodeSelector.nodeValue`	This is an optional parameter. Node Selector value.(note this will be looked first and then if not present global node key will be picked)	**DataType:**Integer **Range:** **Default Value:** zone.

Table 3-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
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
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.startupProbe.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

Table 3-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
cmservice.deployment.startupProbe.periodSeconds	This is an optional parameter. It specifies that the kubelet should perform a liveness probe every xx second	**DataType:**Integer **Range:** 0-65535. **Default Value:** 3
cmservice.deployment.startupProbe.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.startupProbe.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.startupProbe.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:**
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

Table 3-6 (Cont.) Core Backend Parameters

Attribute Name	Description	Details
`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:**

3.3.3 Ingress Gateway Parameters

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

Table 3-7 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>

Table 3-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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>
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.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 3-7 (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
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

Table 3-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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
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

Table 3-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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
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

Table 3-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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
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

Table 3-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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.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.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.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.

Table 3-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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:**
ingress-gateway.cipherSuites	Allowed CipherSuites for TLS1.2, if ingressgateway.enableIncomingHttps is true	**DataType:**List[String] **Range:** TLS_ECDHE_ ECDSA_WIT H_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_AE S_256_GCM_ SHA384 TLS_ECDHE_ ECDSA_WIT H_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 container	**DataType:**Boolean **Range:**True or False **Default Value:** Values will be set by mcncc-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 q
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

Table 3-7 (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
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
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

Table 3-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
`ingress-gateway.dbConfig.dbName`	This is an optional parameter. • It is the name of the database used for M-CNCC Core, user should create DB with unique name before deploying M-CNCC Example: – DB Name in case M-CNCC Core mcncccommonconfig	**DataType:**String **Range:**Valid String **Default Value:** Values will be set by mcncc-core.global.dbName in case of mcncc-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
`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

Table 3-7 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
`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.	**DataType:**Integer **Range:**300-7200 **Default Value:**1800

3.4 A-CNCC Core Configuration Options

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

> **Note**
>
> 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*.

3.4.1 Global Parameters

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

Table 3-8 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

Table 3-8 (Cont.) Global Parameters

Parameter	Description	Details
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 aserviceAccountNamend 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:**
global.httpsEnabled	This is an optional parameter. To Initialize SSL related infrastructure in init 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
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

3.4.2 Ingress Gateway Parameters

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

Table 3-9 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>
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

Table 3-9 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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
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

Table 3-9 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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
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

Table 3-9 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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
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

Table 3-9 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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
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

Table 3-9 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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
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.commonHooksCpu	This is an optional parameter. common Hooks Cpu limit	**DataType:**String **Range:** **Default Value:** 1

Table 3-9 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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.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.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.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 3-9 (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_WIT H_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_AE S_256_GCM_ SHA384 TLS_ECDHE_ ECDSA_WIT H_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 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.enableIncomingHttp q
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 3-9 (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
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

Table 3-9 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
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
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>

Table 3-9 (Cont.) Ingress Gateway Parameters

Attribute Name	Description	Details
`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.	**DataType:**Integer **Range:**300-7200 **Default Value:**1800

3.5 CNC Console Instances Configurations

This section explains about the CNC Console 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, DD-UI, DD-API, PROVGW, NWDAF, DBTIER, OCCM.
> - apiPrefix is must for type CS (OCCNE Common Services), DD (DD-UI and DD-API), NWDAF (NWDAF-UI and NWDAF-API).
> - Supported common services are grafana, kibana, jaeger, prometheus, alertmanager, promxy, opensearch, and jaeger-es.

CNC Console Supported OCCNE Common Services/NFs with expected apiPrefix format.

Table 3-10 CNC Console Supported OCCNE Common Services/NFs 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	/$OCCNE_CLUSTER/alertmanager	/mycne-cluster/alertmanager
Promxy	/$OCCNE_CLUSTER/promxy	/mycne-cluster/promxy
OpenSearch	/$OCCNE_CLUSTER/dashboard	/mycne-cluster/dashboard
Jaeger-ES	/$OCCNE_CLUSTER/esjaeger	/mycne-cluster/esjaeger
Data Director (DD-UI)	/$OCCNE_CLUSTER/ocnadd	/mycne-cluster/ocnadd

Table 3-10 (Cont.) CNC Console Supported OCCNE Common Services/NFs with expected apiPrefix format

Common Service	apiPrefix format	Example
Data Director (DD-API)	/$OCCNE_CLUSTER/ocnaddapi	/mycne-cluster/ocnaddapi
Network Data Analytics Function (NWDAF-UI)	/$OCCNE_CLUSTER/ocnwdaf	/mycne-cluster/ocnwdaf
Network Data Analytics Function (NWDAF-API)	/$OCCNE_CLUSTER/ocnwdafapi	/mycne-cluster/ocnwdafapi

CNC Console supports common services, Data Director, and Network Data Analytics Function NF 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.

3.5.1 CNC Console Instances Configuration Options

This section describes the parameters that are configured in the CNC Console 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: Cluster1	**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

Attribute Name	SubType	Description	Details

	ip	This is a mandatory parameter (if ip is not provided).
		M-CNCC IAM URI IP
		Note:CNC Console supports configuration through IPv4 and IPv6 family IPs.
		IPv6 Sample configuration:
		`mCnccIams:`
		`- id: Cluster1`
		`ip: "[1000:xxxx:xxxx:xxx::yy]"`
		IPv4 Sample Configuration:
		`mCnccIams:`
		`- id: Cluster1`
		`ip: 10.xx.xx.xx`
			DataType: String

	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

Attribute Name	SubType	Description	Details
	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
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

Attribute Name	SubType	Description	Details
global.instances.[]	id	This is a mandatory parameter. ID to uniquely identify the NF Instance or OCCNE 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, DD-UI, DD-API, PROVGW, NWDAF-UI, NWDAF-API, DBTIER, OCCM.	**DataType:** String **Range:**It can take one of these values: BSF, NRF, NSSF, POLICY, SCP, SEPP, UDR-CONFIG, UDR-PROV, CS, DD-UI, DD-API, PROVGW, NWDAF-UI, NWDAF-API, DBTIER, OCCM.
	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 Services, Data Data Director, and NWDAF and used for routing OCCNE Common Services, Data Director, and NWDAF instances. Note: ApiPrefix is not required in case of NF instances (except DD and NWDAF) Example : /<Cluster Prefix>/grafana	**DataType:** String

4
Accessing CNC Console

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

4.1 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/
   ```

4.2 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.

5
Upgrading CNC Console

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

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

> **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.

> **Note**
>
> A-CNCC DB dependency has been removed from CNC Console release 23.2.0. Only CNCC IAM and M-CNCC Core have DB dependency.

The following steps must be followed while performing the upgrade:

1. CNCC IAM DB Backup
2. A-CNCC Core DB Backup
3. CNCC Upgrade
4. Removing A-CNCC Core DB

This section explains about the CNC Console upgrade 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	References	Applicable for CDCS	Applicable for CLI
CNCC IAM DB Backup	CNCC IAM DB Backup	Yes	Yes
A-CNCC Core DB Backup	A-CNCC Core DB Backup	Yes	Yes
CNCC Upgrade	CNCC Upgrade	See Oracle Communications CD Control Server Installation and Upgrade Guide	Yes
Removing A-CNCC Core DB	Removing A-CNCC Core DB	Yes	Yes

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

5.1 Supported Upgrade Paths

This section describes the supported upgrade paths for CNCC

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

The following table lists the supported upgrade paths for Console:

Table 5-1 CNC Console Upgrade Sequence

Deployment Mode	Source Version	Target Version	Upgrade Sequence	Comments
Single Cluster	23.2.x or 23.3.x	23.4.x	**Console Upgrade** Upgrade CNCC **NF Upgrade** Upgrade Instances (NF or CNE/OSO Common Services) **cnDbtier Upgrade**	• CNC Console can be upgraded first as 22.2.0 onwards NF includes menu api to support Console first upgrade functionality. • A-CNCC DB dependency has been removed from CNC Console release 23.2.0. Only CNCC IAM and M-CNCC Core have DB dependency. For more information see the preupgrade and postupgrade tasks. • Helm upgrade CNCC using existing release name (cncc-iam or cncc).

Table 5-1 (Cont.) CNC Console Upgrade Sequence

Deployment Mode	Source Version	Target Version	Upgrade Sequence	Comments
Multi Cluster	23.2.x or 23.3.x	23.4.x	**Console Manager Upgrade** Upgrade CNCC **Console Agent Upgrade** Upgrade A-CNCC Core **NF Upgrade** Upgrade Instances (NF or CNE/OSO Common Services) **cnDbtier Upgrade**	• CNC Console can be upgraded first as 22.2.0 onwards NF includes menu API to support Console first upgrade functionality. • Manager Clusters: – A-CNCC DB dependency has been removed from CNC Console release 23.2.0. Only CNCC IAM and M-CNCC Core have DB dependency. For more information see the preupgrade and postupgrade tasks. – Helm upgrade CNCC using existing release name(cncc-iam or cncc). • Agent Clusters: – A-CNCC DB dependency has been removed from CNC Console release 23.2.0. Only CNCC IAM and M-CNCC Core have DB dependency. For more information see the preupgrade and postupgrade tasks. – Helm upgrade A-CNCC using existing release name (cncc-core or cncc). This will upgrade A-CNCC Core.

Note

- Console should be upgraded first followed by NF for all the supported NFs.
- cnDBTier only supports APIs from 23.3.x onwards. Users with a combination of cnDBTier 23.3.x or older, and NF version 23.4.x or newer won't have access to the data in cnDBTier configuration items.

5.2 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.
3. Install or upgrade the network policies, if applicable. For more information, see Configuring Network Policies for CNC Console.

5.3 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, login to MCM client on one of the SQL node of the cluster. Execute the following command to check the node status:

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

- 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 --no-create-info -h 127.0.0.1 -u <username> -
  p <database-name> | gzip > <backup_filename>.sql.gz
  ```

  Example:

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

5.4 A-CNCC Core DB Backup

This section describes the procedure to take the DB backup of A-CNCC Core.

> **Note**
>
> This step is applicable when upgrading from 23.1.x or older version as CNCC 23.2.0 onwards A-CNCC DB dependency has been taken out. Only CNC Console IAM and M-CNCC Core has DB dependency.

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 DBTier 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;
  ```

- 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 ndbappmysqld-0 -- mysqldump --single-
  transaction --no-tablespaces -h 127.0.0.1 -u cnccusr -p acncccommonconfig
  | gzip >
      acncccommonconfigBackup.sql.gz
  ```

5.5 CNCC Upgrade

This section describes the procedure to upgrade CNCC.

> **Note**
>
> - Existing release name must be used for upgrade.
> - mCnccIams port is assumed as 80. The mCnccIams port configuration must be added only if port is other than 80.

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 <cncc_iam_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 <cncc_iam_release_name> -n <namespace>
   ```

   Example:

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

5.6 Post Upgrade Tasks

This section describes the tasks to be performed after an upgrade.

5.6.1 Removing A-CNCC Core DB

> **Note**
>
> This step is applicable when upgrading from 23.1.x or older version as CNCC 23.2.0 onwards A-CNCC DB dependency has been taken out. Only CNC Console IAM and M-CNCC Core has DB dependency.

Perform the following steps to remove A-CNCC Core DB:

- Log in to a server or machine that can access the SQL nodes of the NDB cluster.

- Connect to the SQL node of the NDB cluster or connect to the cnDBTier.
  To connect to cnDBTier, run the following command:

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

  Example:

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

- Login to the MySQL prompt using a root permission or user that can create users with permissions.

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

- Drop the existing acncccommonconfig database.

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

5.7 Parameters and Definitions During CNC Console Upgrade

Parameters and Definitions during CNCC Upgrade

Table 5-2 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.
<occncc_rollback_iam_schema_<version>.sql>	CNCC DB schema file used for rollback

6
Rolling Back CNC Console

This section provides information about rolling back Cloud Native Configuration Console deployment to the previous release.

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

The following steps must be followed while performing the rollback:

1. CNCC IAM DB Rollback/Restore
2. A-CNCC Core DB Rollback or Restore
3. CNCC Rollback

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

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

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

6.1 Supported Rollback Paths

This section describes the supported upgrade paths for CNC Console

Table 6-1 CNC Console Rollback Sequence

Deployment Mode	Source Version	Target Version	Rollback Sequence	Comments
Single Cluster	23.4.x	23.3.x or 23.2.x	• **cnDbtier Rollback** • **NF Rollback** Rollback Instances (NF or CNE/OSO Common Services) • **Console Rollback** Rollback CNC Console	• CNC Console 23.2.0 onwards A-CNCC DB dependency has been taken out. Only CNC Console IAM and M-CNCC Core has DB dependency. Refer pre rollback tasks section for additional steps before perming rollback. • Helm rollback CNC Console using existing release name (cncc-iam or cncc). This will rollback M-CNCC IAM M-CNCC Core and A-CNCC Core if enabled.
Multi Cluster	23.4.x	23.3.x or 23.2.x	• **cnDbtier Rollback** • **NF Rollback** Rollback Instances (NF or CNE/OSO Common Services) • **Console Agent Rollback** Rollback A-CNCC Core • **Console Manager Rollback** Rollback CNC Console	• Manager Clusters : – CNC Console 23.2.0 onwards A-CNCC DB dependency has been taken out. Refer pre rollback tasks section for additional steps before perming rollback. – Helm rollback CNC Console 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: – CNC Console 23.2.0 onwards A-CNCC DB dependency has been taken out. Refer pre rollback tasks section for additional steps before perming rollback. – Helm rollback cncc-core using existing release name (cncc-core or cncc). This will rollback A-CNCC Core.

6.2 Pre-rollback Tasks

This section describes the tasks to be performed before a rollback.

6.2.1 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.
> - This entire section of Pre Rollback tasks must be done in one go. If any operational delay or error is encountered after starting, the whole rollback process must be repeated from the start.
> - Ensure that all the necessary files mentioned in the section are ready before you begin the Pre Rollback tasks.

To perform a CNC Console IAM DB rollback or Restore:

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 (*<occncc_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;
   ```

   > **Note**
   >
   > You must take the `<occncc_rollback_iam_schema_<version>.sql>` file from the CNC Console version from which you are performing the rollback. For example, if you are performing rollback from version N to version N-2 or N-1, you must use the `<occncc_rollback_iam_schema_<version>.sql>` file given in the CNC Console package of version N.

2. Copy the DB Schema file provided as part of package into the MYSQl pod (occncc_rollback_iam_schema_<version>.sql).
   Before copying the file into MYSQL pod replace string "InnoDB" with "ndbcluster" using following command:

   ```
   sed -i 's/InnoDB/ndbcluster/g' occncc_rollback_iam_schema_<version>.sql
   ```

   For example:

   ```
   sed -i 's/InnoDB/ndbcluster/g' occncc_rollback_iam_schema_23.4.4.sql
   ```

Run the following command to copy the DB Schema file:

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

Example:

```
kubectl cp occncc_rollback_iam_schema_23.4.4.sql cndbtier1/ndbappmysqld-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 ndbappmysqld-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>
```

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/ndbappmysqld-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 ndbappmysqld-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
    ```

6.2.2 A-CNCC Core DB Rollback or Restore

This section provides details of A-CNCC Core Rollback. In case of CNC Console upgrade failure, roll back A-CNCC Core DB to previous version by following the below steps.

> **Note**
>
> - The latest backup must be used for rollback.
> - This step is applicable when rolling back to 23.1.x or older version as CNC Console 23.2.0 onwards A-CNCC DB dependency has been taken out. Only CNC Console IAM and M-CNCC Core has DB dependency.

To perform A-CNCC Core DB rollback or restore:

1. Log in to the deployment cluster, and run the following command in the mysql pod to drop the existing database and create a new database.

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

   Example:

   ```
   DROP DATABASE acncccommonconfig;
   CREATE DATABASE IF NOT EXISTS acncccommonconfig;
   ```

2. Convert the mysqldump file which was taken as a backup (sql.gz file) to a sql file.

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

   Example:

   ```
   gunzip -d acncccommonconfigBackup.sql.gz
   ```

3. Copy the existing file with the following command:

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

   Example:

   ```
   kubectl cp acncccommonconfigBackup.sql cndbtier1/ndbappmysqld-0:/home/mysql
   ```

4. 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 ndbappmysqld-0 -c mysqlndbcluster -- bash
   ```

5. Populate the DB with data using the file which you have copied into the pod after filtering the sqldump file using the following command:

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

   Example:

   ```
   mysql -h 127.0.0.1 -u root -p acncccommonconfig <
   acncccommonconfigBackup.sql
   ```

6. Log in to the MySQL prompt and confirm that the databases are restored.
7. 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 <backup_filename>
   ```

   Example:

   ```
   rm -rf acncccommonconfigBackup.sql
   ```

6.3 CNCC Rollback

This section describes the procedure to Rollback CNCC.

> **Note**
>
> Existing release name must be used for 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 rollback <release_name> <revision_number> -n <namespace>
   ```

   Example:

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

7
Uninstalling CNC Console

This chapter provides information about uninstalling Cloud Native Configuration Console.

> **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.

7.1 Uninstalling CNC Console Using Helm

To uninstall CNC Console, run the following command:

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

Where,

<release_name> is a name provided by the user to identify the helm deployment.

<namespace_name> is a name provided by the user to identify the namespace of CNC Console deployment.

For example:

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

Helm keeps a record of its releases, so you can still reactivate the release after uninstalling it.

To completely remove a release from the cluster, add the `--purge` parameter to `helm delete` command:

```
helm delete --purge release_name
```

For example:

```
helm delete --purge occncc
```

7.2 Deleting Kubernetes Namespace

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
   ```

7.3 Deleting the CNC Console MySQL details

Perform the following procedure to delete the 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;
```

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.

7.4 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. You must perform this procedure before running helm upgrade again.

Following resources require cleanup during upgrade failure:

- Roles
- RoleBindings
- ServiceAccounts
- ConfigMaps
- PodDisruptionBudgets

- Services

> **Note**
>
> While upgrading from older CNCC version to CNCC 23.4.4, 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-acore-ingressgateway-role cncc-mcore-
   ingressgateway-role cncc-iam-ingressgateway-role cncc-helmtest-role
   ```

- **RoleBindings**

1. Run the following command to check whether the RoleBindings 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-acore-ingressgateway-
rolebinding-v1
      cncc-mcore-ingressgateway-rolebinding-v1 cncc-iam-ingressgateway-
rolebinding-v1
      cncc-helmtest-rolebinding
```

- **ServiceAccounts**

1. Run the following command to check whether the ServiceAccounts 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-acore-ingress-gateway cncc-helmtest-
serviceaccount
      cncc-mcore-ingress-gateway cncc-iam-ingress-gateway
```

- **ConfigMaps**

1. Run the following command to check whether the ConfigMaps 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-ingress-
gateway
                                              <release_name>-mcore-
cmservice
                                              <release_name>-mcore-
```

```
ingress-gateway
                                                    <release_name>-iam-ingress-
gateway
```

Example:

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

- **PodDisruptionBudgets**

1. Run the following command to check whether the PodDisruptionBudgets 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-ingress-gateway-podDisruptionBudget
                                                    <release_name>-
mcore-cmservice-podDisruptionBudget
                                                    <release_name>-
mcore-ingress-gateway-podDisruptionBudget
                                                    <release_name>-
iam-ingress-gateway-podDisruptionBudget
```

Example:

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

- **Services**

1. Run the following command to check whether the Services 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-igw-cache
                                         <release_name>-acore-ingress-
gateway
```

```
                                                <release_name>-mcore-cmservice
                                                <release_name>-mcore-igw-cache
                                                <release_name>-mcore-ingress-
gateway
                                                <release_name>-iam-igw-cache
                                                <release_name>-iam-ingress-gateway
                                                <release_name>-iam-kc-headless
                                                <release_name>-iam-kc-http
```

Example:

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

7.5 CNC Console Helm Test Cleanup

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

8
Fault Recovery

This chapter provides information about fault recovery for Cloud Native Configuration Console deployment.

8.1 Overview

You must take database backup and restore it either on the same or a different cluster. It uses the CNCC database to run any command or follow instructions.

> **Note**
>
> This chapter describes fault recovery scenarios of CNC Console and how to recover from those scen

8.2 Impacted Area

The following table describes scenarios about the impacted areas during CNC Console fault recovery:

Table 8-1 Fault Recovery Scenarios Impact Information

Scenario	Requires Fault Recovery or re-install of CNE?	Requires Fault Recovery or re-install of cnDBTier?	Requires Fault Recovery or re-install of CNC Console?	Comments
Scenario 1: Complete Site Failure	Yes	Yes	Yes	
Scenario 2: cnDBTier Corruption	No	Yes	No	DBTier must be restored from backup. Re-install of CnDBTier can be considered, if restoring from backup is not possible. Use helm upgrade if DB configuration is changed

Table 8-1 (Cont.) Fault Recovery Scenarios Impact Information

Scenario	Requires Fault Recovery or re-install of CNE?	Requires Fault Recovery or re-install of cnDBTier?	Requires Fault Recovery or re-install of CNC Console?	Comments
Scenario 3: Console Configuration Database Corruption	No	No	No	Backup and restore of configuration database is required on the impacted site. This needs periodic backup. (Applicable for M-CNCC IAM)
Scenario 4: Deployment Failure	No	No	Yes	CNC Console DB is not restored. Only helm uninstall/install is done. (Applicable for M-CNCC IAM, M-CNCC Core and A-CNCC Core)
Scenario 5: NF Instance Failure	No	No	Yes	The NF endpoints must be updated in the custom values.yaml file and helm upgrade must be performed to incorporate the latest endpoints (Applicable at M-CNCC Core and A-CNCC Core).

8.3 Prerequisites

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

- DBTier should be in healthy state and available on multiple sites along with CNC Console.
- On demand backup should be enabled on DBTier. Scheduled regular backups help to:
  - Restore stable version of the CNC Console configuration database.
  - Minimize significant loss of data due to upgrades or roll back failures.
  - Minimize loss of data due to system failure.
  - Minimize loss of data due to data corruption or deletion due to external input.
  - Migrate Console database information from one site to another.
- Custom values file used at the time of Console deployment is retained. If the `custom_values.yaml` file is not retained, then regenerate it manually. This task increases the overall fault recovery time.
- Docker images used during the last installation or upgrade must be retained in the external data source.
  For the CNC Console database backup and restore prerequisites, see Fault Recovery Procedures - DB Backup and Restore section.

8.4 Fault Recovery Scenarios

This section describes the fault recovery procedures for various scenarios.

8.4.1 Scenario 1: Complete Site Failure

This section describes how to perform fault recovery when either one, many, or all of the sites have software failure.

The following are site failure scenarios:

- Scenario 1A: Single or Multiple Site Failure
- Scenario 1B: All Sites Failure

8.4.1.1 Scenario 1A: Single or Multiple Site Failure

This scenario applies when one or more sites, and not all sites, have failed and there is a requirement to perform fault recovery. It is assumed that the user has DBTier and Oracle Communications CNC Console installed on multiple sites with automatic data replication and backup enabled.

To recover the failed sites:

1. Run the Cloud Native Environment (CNE) installation procedure to install a new cluster. For more information, see *Oracle Communications Cloud Native Environment (OCCNE) Installation Guide.*
2. For CnDBTier fault recovery:
   a. Take on-demand backup from the mate site that has health replication with the failed site or sites.
   b. Use the backup data from the mate site to restore the database.
3. Install CNC Console helm chart. For more information about installing CNC Console, see the Installing CNC Console chapter in the *Oracle Communications Cloud Native Configuration ConsoleInstallation and Upgrade Guide.*

8.4.1.2 Scenario 1B: All Sites Failure

This scenario applies when all sites have failed and there is a requirement to perform fault recovery. It is assumed that the user has DBTier and Oracle Communications CNC Console installed on multiple sites with automatic data replication and backup enabled.

To recover all the failed sites:

1. Run the Cloud Native Environment (CNE) installation procedure to install a new cluster. For more information, see *Oracle Communications Cloud Native Environment (OCCNE) Installation Guide.*
2. Use an on-demand backup file to restore the database from previous data backup.

> **Note**
>
> - The auto-data backup file is one that is built from scheduled automatic backup.
> - The Restore Georeplication Failure chapter contains a procedure for two sites where one of the clusters has a fatal error. You can perform the same procedure for all the sites in a multiple site setup.

3. Install CNC Console helm chart. For more information about installing CNC Console, see the Installing CNC Console chapter in the *Oracle Communications Cloud Native Configuration Console Installation and Upgrade Guide.*

8.4.2 Scenario 2: cnDBTier Corruption

This section describes how to recover a database when the data replication has failed due to database corruption and cnDBTier has failed in single, multiple sites, or all sites.

When the database gets corrupted, the database on all the other sites can also get corrupted due to data replication. It depends on the replication status after the corruption has occurred. If the data replication fails due to database corruption, then DBTier fails in either single or multiple sites (not all sites). If the data replication is successful, then database corruption replicates to all the cnDBTier sites and cnDBTier fails in all sites.

The following are cnDBTier failure scenarios:

If corrupted database is replicated to mated sites, follow:

- Scenario 2A: When DBTier fails in Single or Multiple (but not all) Sites

If corrupted database causes replication failure and hence local to a site, follow:

- Scenario 2B: When CnDBTier failed in all Sites

> **Note**
>
> This impacts all the NFs using the corrupted cnDBTier. All the NFs sharing cnDBTier needs to do a fault recovery as cnDBTier is corrupted.

8.4.2.1 Scenario 2A: When DBTier fails in Single or Multiple (but not all) Sites

This section describes how to recover database when the data replication has failed due to database corruption and DBTier has failed in either single or multiple sites (not all sites).

To recover database:

1. Uninstall CNC Console helm chart. For information about uninstalling CNC Console, see the Uninstalling CNC Console chapter in the *Oracle Communications Cloud Native Configuration ConsoleInstallation and Upgrade Guide.*
2. For DBTier fault recovery:
   a. Create on-demand backup from mated site that has health replication with failed site.
   b. Use the backup data from mate site for restoration.

3. Install CNC Console helm chart. For more information about installing CNC Console, see the Installing CNC Console chapter in the Oracle Communication Cloud Native Configuration ConsoleInstallation and Upgrade Guide.

8.4.2.2 Scenario 2B: When CnDBTier failed in all Sites

This section describes how to recover database when successful data replication corrupts all the DBTier sites.

To recover database:

1. Uninstall CNC Console helm charts. For more information about uninstalling CNC Console, see the Uninstalling CNC Console chapter in the *Oracle Communications Cloud Native Configuration ConsoleInstallation and Upgrade Guide.*
2. For CnDBTier fault recovery:
   a. Use an on-demand backup file to restore the database from the previous data backup.

> **Note**
>
> The Restore Georeplication Failure chapter has a procedure for two sites where one of the cluster has fatal error. You can perform that procedure for all the sites in a multiple site setup.

3. Install CNC Console helm charts. For more information about installing CNC Console, see the Installing CNC Console chapter in the *Oracle Communication Cloud Native Configuration Console Installation and Upgrade Guide.*

8.4.3 Scenario 3: Console Configuration Database Corruption

This section describes how to recover when the CNC Console configuration database is corrupted.

For recovery and restore procedure, see the Fault Recovery Procedures - DB Backup and Restore section.

8.4.4 Scenario 4: Deployment Failure

This section describes how to recover CNC Console when its deployment fails.

For recovery and restore procedure, see Restoring CNC Console .

8.4.5 Scenario 5: NF Instance Failure

Perform this procedure to recover from NF instance failure.

1. Refer to the procedures in the specific NF Disaster Recovery Guide to find the necessary action required to be taken.
2. Check whether the NF endpoints are the same.
   a. If the NF endpoints are the same, no change is required in CNC Console side.
   b. If the NF endpoints are different, update the NF IP and Port in the *occncc_custom_values_<version>.yaml* file and perform a helm upgrade operation to incorporate the new NF URL.

A.1 CNC Console Microservices

This section provides the details of CNC Console microservices.

A.1.1 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 2 CNC Console IAM Microservices

NAME	TYPE	PORT(S)
cncc-iam-kc-headless	ClusterIP	8285/TCP
cncc-iam-kc-http	ClusterIP	8285/TCP
cncc-iam-ingress-gateway	LoadBalancer	8080:30346/TCP
cncc-iam-cncc-iam-igw-cache	ClusterIP	8000/TCP

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

A.1.2 M-CNCC Core Microservices

M-CNCC Core has two microservices:

1. **cncc-mcore-ingress-gateway**: cncc-mcore-ingress-gateway is responsible to forward the request either to the Agent CNCC or to CNCC GUI.
2. **cncc-mcore_cmservice**: cncc-mcore_cmservice is responsible for displaying CNCC GUI.

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

Table 3 M-CNCC Core Microservices

NAME	TYPE	PORT(S)
cncc-mcore-cmservice	ClusterIP	8442/TCP
cncc-mcore-igw-cache	ClusterIP	8000/TCP
cncc-mcore-ingress-gateway	LoadBalancer	8080:31417/TCP

> **Note**
>
> `cncc-iam-cncc-iam-igw-cache` is an Ingress Gateway cache service which is enabled by default by Ingress Gateway chart. CNC Console core does not use this feature.

A.1.3 A-CNCC Core Microservices

A-CNCC Core has the "cncc-acore-ingress-gateway" microservice . It is cncc-acore-ingress-gateway is responsible for forwarding the request to the NF.

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

Table 4 A-CNCC Core Microservices

NAME	TYPE	PORT(S)
cncc-acore-cncc-core-igw-cache	ClusterIP	8000/TCP
cncc-acore-ingress-gateway	ClusterIP	8080/TCP

> **Note**
>
> `cncc-acore-igw-cache` is an Ingress Gateway cache service which is enabled by default by Ingress Gateway helm chart. CNC Console Core does not use this feature.

B.1 CNC Console Microservices to Port Mapping

This section contains CNC Console microservices to port mapping details.

Table 5 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
cncc-iam-http	cncc-iam-kc-0	service	Thirdparty SAML service	External				saml	Depends on Customer SAML Provider

Table 5 (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-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	
metrics	worker node	pod	cncc-mcore-ingress-gateway	Internal	Internal	Internal / K8s	9090	http	GET /actuator/health

C.1 CNC Console Validation-hook Error Codes

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

Validation hook pod is auto deleted if validation succeeds. In case of failure, pod status is displayed as" *Error*.

In case of Validation-hook errors, refer to the following error codes:

Table 6 Validation-hook Error Codes

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> • DD instances must have one of these subtypes <ocnadd, ocnaddapi> • NWDAF instances must have one of these subtypes <ocnwdaf, ocnwdafapi> • Both ip and fqdn cannot be provided. • 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 • there must be 2 DD instances with same instance id (one	Invalid value. Resource: mCncclam, 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 mCncclam 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. Invalid value. Resource: DD, ID: Cluster1-dd-instance1, Attribute: N/A. More than one DD instance

Table 6 (Cont.) Validation-hook Error Codes

Error Code	Error Message Format	Error Scenarios	Sample Error Messages
		with type DD-UI and the other with DD-API) • there must be 2 NWDAF instances with same instance id (one with type NWDAF-UI and the other with NWDAF-API) • NF type should be present along with cnDBTier instance • Each NF can have a maximum of 2 instances with same instance ID, one of which must have type cnDBTier	with same id and type is not allowed Invalid value. Resource: instance, ID: Cluster1-dd-instance1, Attribute: apiPrefix. For type DD valid Api Prefixes: [/ocnadd, /ocnaddapi] Invalid value. Resource: NF, ID: Cluster1-instance1, Attribute: N/A. NF type should be present along with cnDBTier instance
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/DD/NWDAF subtype	• Duplicate value(s). Resource: aCncc, ID: [Cluster3], Attribute: id. • Duplicate value(s). Resource: instance, ID: N/A, Attribute: apiPrefix. Duplicate Attribute values: [/occne12-ip-cluster/ocnadd] for owner: Kolkata
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.
1004	Missing value. Resource: <Configuration Name>, ID: <ID>, Attribute: <Attribute>. <More Info>	• Missing apiPrefix parameter for type CS, DD-UI, DD-API, NWDAF-UI, or NWDAF-API • 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.

C.1.1 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 Validation-hook 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 Validation-hook 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"
}
```

D.1 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".
> - To enable cnDBTier menu for NFs, see the NF Instance Configuration With cnDBTier Menu Enabled section.

Support for Instance Configuration over IPV6 Family IPs

- CNC Console supports configuration of NF and Common Service Instances over IPv6 Family IPs as well. For configuring an IPv6 IP in the Instances:
  - IP must be enclosed in square brackets "[]" as per IPv6 standards
  - The complete IP String must be enclosed in double quotes (""). For example, "[2606:b400:605:b82e::]"

Sample Configuration Using IPV6

```
global:

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

  isMultiClusterDeployment: false
   # Automatic route generation for CNCC Manager & Agent Deployment
  self:
    cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: "[2606:b400:605:b82e::]"
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.con6.svc.trypticon.lab.us.oracle.com
      port: 80
  instances:
    - id : Cluster1-bsf-instance1
      type: BSF
      owner: Cluster1
```

```
    ip: "[2606:b400:605:b82e::1]"
    port: 80
  - id: Cluster1-kibana
    type: CS
    owner: Cluster1
    ip: "[2606:b400:605:b82e::3]"
    apiPrefix: /jazz/kibana
```

D.1.1 BSF Instances Configuration Examples

D.1.1.1 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
```

D.1.1.2 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
```

D.1.2 DD Instance Configuration Examples

D.1.2.1 DD 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.occne-12ipcluster
      port: 80
  instances:
    - id : Cluster1-dd-instance1
      type: DD-UI
      owner: Cluster1
      fqdn: ocnaddgui.ocnadd-deploy.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnadd
    - id : Cluster1-dd-instance1
      type: DD-API
      owner: Cluster1
      fqdn: ocnaddbackendrouter.ocnadd-deploy.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnaddapi
```

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.occne-12ipcluster
      port: 443
  instances:
    - id : Cluster1-dd-instance1
      type: DD-UI
      owner: Cluster1
      fqdn: ocnaddgui.ocnadd-deploy.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnadd
    - id : Cluster1-dd-instance1
      type: DD-API
      owner: Cluster1
      fqdn: ocnaddbackendrouter.ocnadd-deploy.svc
       port: 80
      apiPrefix: /occne-12ipcluster/ocnaddapi
```

D.1.2.2 DD Multicluster Deployment Instances Configuration Examples

M-CNCC Core on Cluster1, A-CNCC Core on Cluster1 and A-CNCC Core on Cluster2

CNC 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.occne-12ipcluster
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id : Cluster1-dd-instance1
      type: DD-UI
      owner: Cluster1
      fqdn: ocnaddgui.ocnadd-deploy.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnadd
    - id : Cluster1-dd-instance1
      type: DD-API
      owner: Cluster1
      fqdn: ocnaddbackendrouter.ocnadd-deploy.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnaddapi
    - id : Cluster2-dd-instance1
      type: DD-UI
      owner: Cluster2
      fqdn: ocnaddgui.ocnadd-deploy2.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnadd
    - id : Cluster2-dd-instance1
      type: DD-API
      owner: Cluster2
      fqdn: ocnaddbackendrouter.ocnadd-deploy2.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnaddapi
```

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.occne-12ipcluster
    port: 443
  - id: Cluster2
    role: Cluster2
    scheme: https
    ip: 10.xx.xx.xx
    port: 443
instances:
  - id : Cluster1-dd-instance1
    type: DD-UI
    owner: Cluster1
    fqdn: ocnaddgui.ocnadd-deploy.svc
    port: 80
    apiPrefix: /occne-12ipcluster/ocnadd
  - id : Cluster1-dd-instance1
    type: DD-API
    owner: Cluster1
    fqdn: ocnaddbackendrouter.ocnadd-deploy.svc
    port: 80
    apiPrefix: /occne-12ipcluster/ocnaddapi
  - id : Cluster2-dd-instance1
    type: DD-UI
    owner: Cluster2
    fqdn: ocnaddgui.ocnadd-deploy2.svc
    port: 80
    apiPrefix: /occne-12ipcluster/ocnadd
  - id : Cluster2-dd-instance1
    type: DD-API
    owner: Cluster2
    fqdn: ocnaddbackendrouter.ocnadd-deploy2.svc
    port: 80
    apiPrefix: /occne-12ipcluster/ocnaddapi
```

> **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-dd-instance1
      type: DD-UI
      owner: Cluster2
      fqdn: ocnaddgui.ocnadd-deploy2.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnadd
    - id : Cluster2-dd-instance1
      type: DD-API
      owner: Cluster2
      fqdn: ocnaddbackendrouter.ocnadd-deploy2.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnaddapi
```

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-dd-instance1
      type: DD-UI
      owner: Cluster2
      fqdn: ocnaddgui.ocnadd-deploy.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnadd
    - id : Cluster2-dd-instance1
      type: DD-API
      owner: Cluster2
      fqdn: ocnaddbackendrouter.ocnadd-deploy2.svc
      port: 80
      apiPrefix: /occne-12ipcluster/ocnaddapi
```

D.1.3 NRF Instances Configuration Examples

D.1.3.1 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
```

D.1.3.2 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
```

D.1.4 NSSF Instances Configuration Examples

D.1.4.1 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
```

D.1.4.2 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
```

D.1.5 NWDAF Instance Configuration Examples

D.1.5.1 NWDAF 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.occne-223cluster
      port: 80
  instances:
    - id : Cluster1-nwdaf-instance1
      type: NWDAF-UI
      owner: Cluster1
      fqdn: nwdafgui.nwdaf-gui.svc
      port: 80
      apiPrefix: /occne-223cluster/nwdaf-gui/ocnwdaf
    - id : Cluster1-nwdaf-instance1
      type: NWDAF-API
      owner: Cluster1
      fqdn: nwdaf-portal-service.nwdaf-gui.svc
      port: 80
      apiPrefix: /occne-223cluster/nwdaf-gui/ocnwdafapi
```

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.occne-223cluster
      port: 443
  instances:
    - id : Cluster1-nwdaf-instance1
      type: NWDAF-UI
      owner: Cluster1
      fqdn: nwdafgui.nwdaf-gui.svc
      port: 80
      apiPrefix: /occne-223cluster/nwdaf-gui/ocnwdaf
    - id : Cluster1-nwdaf-instance1
      type: NWDAF-API
      owner: Cluster1
      fqdn: nwdaf-portal-service.nwdaf-gui.svc
      port: 80
      apiPrefix: /occne-223cluster/nwdaf-gui/ocnwdafapi
```

D.1.5.2 NWDAF Multicluster Deployment Instances Configuration Examples

M-CNCC Core on Cluster1, A-CNCC Core on Cluster1 and A-CNCC Core on Cluster2

CNC 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.occne-223cluster
      port: 80
    - id: Cluster2
      role: Cluster2
      ip: 10.xx.xx.xx
      port: 80
  instances:
    - id : Cluster1-nwdaf-instance1
      type: NWDAF-UI
      owner: Cluster1
      fqdn: nwdafgui.nwdaf-gui.svc
      port: 80
      apiPrefix: /occne-223cluster/nwdaf-gui/ocnwdaf
    - id : Cluster1-nwdaf-instance1
      type: NWDAF-API
      owner: Cluster1
      fqdn: nwdaf-portal-service.nwdaf-gui.svc
      port: 80
      apiPrefix: /occne-223cluster/nwdaf-gui/ocnwdafapi
    - id : Cluster2-nwdaf-instance1
      type: NWDAF-UI
      owner: Cluster2
      ip: 10.xx.xx.xx
      port: 80
      apiPrefix: /occne-224cluster/nwdaf-gui/ocnwdaf
    - id : Cluster2-nwdaf-instance1
      type: NWDAF-API
      owner: Cluster2
      ip: 10.xx.xx.xx
      port: 80
      apiPrefix: /occne-224cluster/nwdaf-gui/ocnwdafapi
```

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.occne-223cluster
    port: 443
  - id: Cluster2
    role: Cluster2
    scheme: https
    ip: 10.xx.xx.xx
    port: 443
instances:
  - id : Cluster1-nwdaf-instance1
    type: NWDAF-UI
    owner: Cluster1
    fqdn: nwdafgui.nwdaf-gui.svc
    port: 80
    apiPrefix: /occne-223cluster/nwdaf-gui/ocnwdaf
  - id : Cluster1-nwdaf-instance1
    type: NWDAF-API
    owner: Cluster1
    fqdn: nwdaf-portal-service.nwdaf-gui.svc
    port: 80
    apiPrefix: /occne-223cluster/nwdaf-gui/ocnwdafapi
  - id : Cluster2-nwdaf-instance1
    type: NWDAF-UI
    owner: Cluster2
    ip: 10.xx.xx.xx
    port: 80
    apiPrefix: /occne-224cluster/nwdaf-gui/ocnwdaf
  - id : Cluster2-nwdaf-instance1
    type: NWDAF-API
    owner: Cluster2
    ip: 10.xx.xx.xx
    port: 80
    apiPrefix:  /occne-224cluster/nwdaf-gui/ocnwdafapi
```

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-nwdaf-instance1
      type: NWDAF-UI
      owner: Cluster2
      ip: 10.xx.xx.xx
      port: 80
      apiPrefix: /occne-224cluster/nwdaf-gui/ocnwdaf
     - id : Cluster2-nwdaf-instance1
      type: NWDAF-API
      owner: Cluster2
      ip: 10.xx.xx.xx
      port: 80
      apiPrefix: /occne-224cluster/nwdaf-gui/ocnwdafapi
```

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 mCnccCores:
  - id: Cluster1
aCnccs:
  - id: Cluster2
    role: Cluster2
instances:

  - id : Cluster2-nwdaf-instance1
    type: NWDAF-UI
    owner: Cluster2
    ip: 10.xx.xx.xx
    port: 80
    apiPrefix: /occne-224cluster/nwdaf-gui/ocnwdaf
  - id : Cluster2-nwdaf-instance1
    type: NWDAF-API
    owner: Cluster2
    ip: 10.xx.xx.xx
    port: 80
    apiPrefix: /occne-224cluster/nwdaf-gui/ocnwdafapi
```

D.1.6 Policy Instances Configuration Examples

D.1.6.1 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
```

D.1.6.2 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
```

D.1.7 PROVGW Instances Configuration Examples

D.1.7.1 PROVGW 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-provgw-instance1
      type: PROVGW
      owner: Cluster1
```

```
    fqdn: provgw1-provgw-config.provgw.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-provgw-instance1
      type: PROVGW
      owner: Cluster1
      fqdn: provgw1-provgw-config.provgw.svc.cluster.local
      port: 80
```

D.1.7.2 PROVGW Multicluster Deployment Instances Configuration Examples

M-CNCC Core on Cluster1, A-CNCC Core on Cluster1 and A-CNCC Core on Cluster2

CNC 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-provgw-instance1
      type: PROVGW
      owner: Cluster1
      fqdn: provgw1-provgw-config.provgw.svc.cluster.local
      port: 80
    - id : Cluster2-provgw-instance1
      type: PROVGW
      owner: Cluster2
     fqdn: provgw1-provgw-config.provgw.svc.cluster.local
      port: 80
    - id : Cluster2-provgw-instance2
      type: PROVGW
      owner: Cluster2
      fqdn: provgw1-provgw-config.provgw2.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-provgw-instance1
      type: PROVGW
      owner: Cluster1
      fqdn: provgw1-provgw-config.provgw.svc.cluster.local
      port: 80
    - id : Cluster2-provgw-instance1
      type: PROVGW
      owner: Cluster2
      fqdn: provgw1-provgw-config.provgw.svc.cluster.local
      port: 80
    - id : Cluster2-provgw-instance2
      type: PROVGW
      owner: Cluster2
      fqdn: provgw1-provgw-config.provgw2.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-provgw-instance1
    type: PROVGW
    owner: Cluster2
    fqdn: provgw1-provgw-config.provgw.svc.cluster.local
    port: 80
  - id : Cluster2-provgw-instance2
    type: PROVGW
    owner: Cluster2
    fqdn: provgw1-provgw-config.provgw2.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-provgw-instance1
      type: PROVGW
```

```
    owner: Cluster2
    fqdn: provgw1-provgw-config.provgw.svc.cluster.local
    port: 80
  - id : Cluster2-provgw-instance2
    type: PROVGW
    owner: Cluster2
    fqdn: provgw1-provgw-config.provgw2.svc.cluster.local
    port: 80
```

D.1.8 SCP Instances Configuration Examples

D.1.8.1 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
```

D.1.8.2 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
```

D.1.9 SEPP Instances Configuration Examples

D.1.9.1 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
```

D.1.9.2 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
```

D.1.10 UDR Instances Configuration Examples

D.1.10.1 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
```

D.1.10.2 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
```

D.1.11 Common Service Instances Configuration Examples

D.1.11.1 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
  - id: Cluster1-promxy
    type: CS
    owner: Cluster1
    fqdn: occne-promxy-apigw-nginx.occne-infra.svc
    apiPrefix: /jazz/promxy
  - id: Cluster1-opensearch
    type: CS
    owner: Cluster1
    fqdn: occne-opensearch-dashboards.occne-infra.svc
    apiPrefix: /jazz/dashboard
  - id: Cluster1-jaegeres
    type: CS
    owner: Cluster1
    fqdn: occne-tracer-elasticsearch-jaeger-query.occne-infra.svc
    apiPrefix: /jazz/esjaeger
```

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
  - id: Cluster1-promxy
    type: CS
    owner: Cluster1
    fqdn: occne-promxy-apigw-nginx.occne-infra.svc
    apiPrefix: /jazz/promxy
  - id: Cluster1-opensearch
    type: CS
    owner: Cluster1
    fqdn: occne-opensearch-dashboards.occne-infra.svc
    apiPrefix: /jazz/dashboard
  - id: Cluster1-jaegeres
    type: CS
    owner: Cluster1
    fqdn: occne-tracer-elasticsearch-jaeger-query.occne-infra.svc
    apiPrefix: /jazz/esjaeger
```

D.1.11.2 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: Cluster1-promxy
```

```
    type: CS
    owner: Cluster1
    fqdn: occne-promxy-apigw-nginx.occne-infra.svc
    apiPrefix: /jazz/promxy
  - id: Cluster1-opensearch
    type: CS
    owner: Cluster1
    fqdn: occne-opensearch-dashboards.occne-infra.svc
    apiPrefix: /jazz/dashboard
  - id: Cluster1-jaegeres
    type: CS
    owner: Cluster1
    fqdn: occne-tracer-elasticsearch-jaeger-query.occne-infra.svc
    apiPrefix: /jazz/esjaeger
  - 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
  - id: Cluster2-promxy
    type: CS
    owner: Cluster2
    fqdn: occne-promxy-apigw-nginx.occne-infra.svc
    apiPrefix: /rivendell-2210/promxy
  - id: Cluster2-opensearch
    type: CS
    owner: Cluster2
    fqdn: occne-opensearch-dashboards.occne-infra.svc
    apiPrefix: /rivendell-2210/dashboard
  - id: Cluster2-jaegeres
    type: CS
    owner: Cluster2
    fqdn: occne-tracer-elasticsearch-jaeger-query.occne-infra.svc
    apiPrefix: /rivendell-2210/esjaeger
```

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.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: Cluster1-promxy
    type: CS
    owner: Cluster1
    fqdn: occne-promxy-apigw-nginx.occne-infra.svc
    apiPrefix: /jazz/promxy
  - id: Cluster1-opensearch
    type: CS
    owner: Cluster1
    fqdn: occne-opensearch-dashboards.occne-infra.svc
    apiPrefix: /jazz/dashboard
  - id: Cluster1-jaegeres
    type: CS
    owner: Cluster1
    fqdn: occne-tracer-elasticsearch-jaeger-query.occne-infra.svc
    apiPrefix: /jazz/esjaeger
  - 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
- id: Cluster2-promxy
  type: CS
  owner: Cluster2
  fqdn: occne-promxy-apigw-nginx.occne-infra.svc
  apiPrefix: /rivendell-2210/promxy
- id: Cluster2-opensearch
  type: CS
  owner: Cluster2
```

```
    fqdn: occne-opensearch-dashboards.occne-infra.svc
    apiPrefix: /rivendell-2210/dashboard
  - id: Cluster2-jaegeres
    type: CS
    owner: Cluster2
    fqdn: occne-tracer-elasticsearch-jaeger-query.occne-infra.svc
    apiPrefix: /rivendell-2210/esjaeger
```

> **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
  - id: Cluster2-promxy
    type: CS
    owner: Cluster2
    fqdn: occne-promxy-apigw-nginx.occne-infra.svc
    apiPrefix: /rivendell-2210/promxy
  - id: Cluster2-opensearch
    type: CS
    owner: Cluster2
    fqdn: occne-opensearch-dashboards.occne-infra.svc
    apiPrefix: /rivendell-2210/dashboard
  - id: Cluster2-jaegeres
    type: CS
    owner: Cluster2
    fqdn: occne-tracer-elasticsearch-jaeger-query.occne-infra.svc
    apiPrefix: /rivendell-2210/esjaeger
```

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
  - id: Cluster2-promxy
    type: CS
    owner: Cluster2
    fqdn: occne-promxy-apigw-nginx.occne-infra.svc
    apiPrefix: /rivendell-2210/promxy
  - id: Cluster2-opensearch
    type: CS
    owner: Cluster2
    fqdn: occne-opensearch-dashboards.occne-infra.svc
    apiPrefix: /rivendell-2210/dashboard
  - id: Cluster2-jaegeres
    type: CS
    owner: Cluster2
    fqdn: occne-tracer-elasticsearch-jaeger-query.occne-infra.svc
    apiPrefix: /rivendell-2210/esjaeger
```

D.1.12 CNC Console Instances Configuration With cnDBTier Menu Enabled

D.1.12.1 CNC Console Single-Cluster Configuration with cnDBTier Menu Enabled

CNC Console 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
      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-cncc-instance1
      type: CNCC
      owner: Cluster1
    - id : Cluster1-cncc-instance1
      type: DBTIER
      owner: Cluster1
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.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

CNC Console 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-cncc-instance1
      type: CNCC
      owner: Cluster1
    - id : Cluster1-cncc-instance1
      type: DBTIER
      owner: Cluster1
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.svc.cluster.local
      port: 80
```

D.1.12.2 CNCC Multicluster Configuration with cnDBTier Menu Enabled

(M-CNCC Core on Cluster1, A-CNCC Core on Cluster1, and A-CNCC Core on Cluster2)

CNC Console Core (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-cncc-instance1
      type: CNCC
      owner: Cluster1
    - id : Cluster1-cncc-instance1
      type: DBTIER
      owner: Cluster1
```

```
    fqdn: mysql-cluster-db-monitor-svc.cndbtier.svc.cluster.local
    port: 80
  - id : Cluster2-cncc-instance1
    type: CNCC
    owner: Cluster2
  - id : Cluster2-cncc-instance1
    type: DBTIER
    owner: Cluster2
    fqdn: mysql-cluster-db-monitor-svc.cndbtier.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

CNC Console 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-cncc-instance1
      type: CNCC
      owner: Cluster1
    - id : Cluster1-cncc-instance1
      type: DBTIER
      owner: Cluster1
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.svc.cluster.local
```

```
    port: 80
  - id : Cluster2-cncc-instance1
    type: CNCC
    owner: Cluster2
  - id : Cluster2-cncc-instance1
    type: DBTIER
    owner: Cluster2
    fqdn: mysql-cluster-db-monitor-svc.cndbtier.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.

CNC Console 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-cncc-instance1
      type: CNCC
      owner: Cluster2
    - id : Cluster2-cncc-instance1
      type: DBTIER
      owner: Cluster2
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.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

CNC Console 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-cncc-instance1
      type: CNCC
      owner: Cluster2
    - id : Cluster2-cncc-instance1
      type: DBTIER
      owner: Cluster2
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.svc.cluster.local
      port: 80
```

D.1.13 OCCM Instance Configuration Examples

D.1.13.1 OCCM Single Cluster Configuration

OCCM Single Cluster Configuration

CNC Console 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-occm-instance1
      type: OCCM
      owner: Cluster1
      fqdn: occm.occm.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

CNC Console 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-occm-instance1
      type: OCCM
      owner: Cluster1
```

```
    fqdn: occm.occm.svc.cluster.local
    port: 80
```

D.1.13.2 OCCM Multicluster Configuration

(M-CNCC Core on Cluster1 , A-CNCC Core on Cluster1, and A-CNCC Core on Cluster2)

CNC Console 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-occm-instance1
      type: OCCM
      owner: Cluster1
      fqdn: occm.occm.svc.cluster.local
      port: 80
    - id : Cluster2-occm-instance1
      type: OCCM
      owner: Cluster2
      fqdn: occm.occm.svc.cluster.local
      port: 80
    - id : Cluster2-occm-instance2
      type: OCCM
      owner: Cluster2
      fqdn: occm.occm.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-occm-instance1
      type: OCCM
      owner: Cluster1
      fqdn: occm.occm.svc.cluster.local
      port: 80
    - id : Cluster2-ocm-instance1
      type: OCCM
      owner: Cluster2
      fqdn: occm.occm.svc.cluster.local
      port: 80
    - id : Cluster2-occm-instance2
      type: OCCM
      owner: Cluster2
      fqdn: occm.occm.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-occm-instance1
      type: OCCM
      owner: Cluster2
      fqdn: occm.occm.svc.cluster.local
      port: 80
    - id : Cluster2-occm-instance2
      type: OCCM
      owner: Cluster2
      fqdn: occm.occm.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-occm-instance1
    type: OCCM
    owner: Cluster2
    fqdn: occm.occm.svc.cluster.local
    port: 80
  - id : Cluster2-occm-instance2
    type: OCCM
    owner: Cluster2
    fqdn: occm.occm.svc.cluster.local
    port: 80
```

D.1.14 NF Instance Configuration With cnDBTier Menu Enabled

D.1.14.1 NF Single Cluster Configuration With cnDBTier Menu Enabled

NF instance configuration should include DBTIER instance configuration to enable DBTIER menu items for NF.

Here is the example for SCP instance configuration. Similar configuration should be performed for other NFs.

CNC Console 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.ocscp.svc.cluster.local
      port: 80
    - id : Cluster1-scp-instance1
      type: DBTIER
      owner: Cluster1
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.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

CNC Console 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.ocscp.svc.cluster.local
    port: 80
  - id : Cluster1-scp-instance1
    type: DBTIER
    owner: Cluster1
    fqdn: mysql-cluster-db-monitor-svc.cndbtier.svc.cluster.local
    port: 80
```

D.1.14.2 NF Multicluster Configuration With cnDBTier Menu Enabled

(M-CNCC Core on Cluster1, A-CNCC Core on Cluster1, and A-CNCC Core on Cluster2)

CNC Console Core (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-scp-instance1
      type: SCP
      owner: Cluster1
      fqdn: ocscp-scpc-configuration.ocscp.svc.cluster.local
      port: 80
    - id : Cluster1-scp-instance1
      type: DBTIER
      owner: Cluster1
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.svc.cluster.local
      port: 80
    - id : Cluster2-scp-instance1
      type: SCP
```

```
    owner: Cluster2
    fqdn: ocscp-scpc-configuration.ocscp.svc.cluster.local
    port: 80
  - id : Cluster2-scp-instance1
    type: DBTIER
    owner: Cluster2
    fqdn: mysql-cluster-db-monitor-svc.cndbtier.svc.cluster.local
    port: 80
```

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

CNC Console 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-cncc-instance1
      type: CNCC
      owner: Cluster1
    - id : Cluster1-cncc-instance1
      type: DBTIER
      owner: Cluster1
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.svc.cluster.local
      port: 80
    - id : Cluster2-cncc-instance1
```

```
    type: CNCC
    owner: Cluster2
  - id : Cluster2-cncc-instance1
    type: DBTIER
    owner: Cluster2
    fqdn: mysql-cluster-db-monitor-svc.cndbtier.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.

CNC Console 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-cncc-instance1
      type: CNCC
      owner: Cluster2
    - id : Cluster2-cncc-instance1
      type: DBTIER
      owner: Cluster2
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.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

CNC Console 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-cncc-instance1
      type: CNCC
      owner: Cluster2
    - id : Cluster2-cncc-instance1
      type: DBTIER
      owner: Cluster2
      fqdn: mysql-cluster-db-monitor-svc.cndbtier.svc.cluster.local
      port: 80
```

E.1 cnDBTier Profile

This section includes information about cnDBTier profiles.

cnDBTier Profile

Table 7 cnDBTier Profile

Service/Component	Per Pod						Side Cars:db-backup-exec-service (DB Tier Sidecar)Service Mesh Sidecar (ASM, Istio/ Envoy)				Number of Pods (Replica)	Total					
	CPU		Memory		Storage		CPU		Memory			CPU		Memory		Storage	
	Req	Limit	Req	Limit	PVC	Count	Req	Limit	Req	Limit		Req	Limit	Req	Limit	PVC	Count
Management (MGMT STS) • k8 Resource Type: StatefulSet • Sidecar name: NA • svc: ndbmgmdsvc	2	2	1	1.25	2	1	0.2	0.2	0.256	0.256	**2**(mgmReplicaCount)	4	4	2	2.5	4	2
NDB Data Store (NDB STS) • k8 Resource Type: StatefulSet • Sidecar name: db-executor-svc • svc : ndbmtdsvc	2	2	4	4	64	11	1.2	1.2	2.256	2.256	**2** (ndbReplicaCount)	6.4	6.4	12.512	12.512	128	22
Replication (SQL STS - mysqld) • k8 Resource Type: StatefulSet • Sidecar name: init-sidecar • svc : ndbmysqldsvc	2	2	1	1	4	1	0.3	0.3	0.512	0.512	**2** 2/4/6 (Depending on GR setup of 2/3/4 site) (apiReplicaCount)	4.6	4.6	3.024	3.024	8	2
App Access (SQL STS - appmysqld) • k8 Resource Type StatefulSet • Sidecar name: init-sidecar • svc : ndbappmysqldsvc	2	2	1	1	4	1	0.1	0.1	0.256	0.256	**2** (ndbappReplicaCount)	4.2	4.2	2.512	2.512	8	2

Table 7 (Cont.) cnDBTier Profile

Monitor Service • Sidecar name: NA • k8 Resource Type: Deployment • svc: db-monitor-service	1	1	0.5	0.5	0	0	NA	NA	NA	NA	**1**	1	1	0.5	0.5	0	0
Replication Service • Sidecar name: init-discover-sql-ips • k8 Resource Type: Deployment • svc: db-replication-service	0.6	1	1	2	2	1	0.2	0.2	0.5	0.5	**1**	0.8	1.2	2.5	2.5	2	1
GR Recovery Resources • Used for performing Disaster Recovery	2	2	4	4	0	0	0.2	0.2	0.5	0.5		2.2	2.2	4.5	4.5	0	0
Backup Manager Service • Sidecar name: NA • k8 Resource Type: Deployment • svc: db-backup-manager-svc	0.1	0.1	0.128	0.128	0	0	NA	NA	NA	NA	**1**	0.1	0.1	0.128	0.128	0	0
postUpgradeJob	0.1	0.1	0.256	0.256	0	0						0.1	0.1	0.256	0.256	0	0
preUpgradeJob	0.1	0.1	0.256	0.256	0	0						0.1	0.1	0.256	0.256	0	0
helm test • Sidecar name: NA • k8 Resource Type: Job *Note: This will be created only when "helm test" is performed*	0.1	0.1	0.256	0.256	0	0						0.1	0.1	0.256	0.256	0	0
											12	23.6	24.0	28.444	28.944	42	11

Table 8 cnDBTier Profile Ephemeral Storage

Service / Component	Per Pod		Side Cars:db-backup-exec-service (DB Tier Sidecar)Service Mesh Sidecar (ASM, Istio/Envoy)		Number of Pods (Replica)	Total	
	Ephemeral Storage		Ephemeral Storage			Ephemeral Storage	
	Req	Limit	Req	Limit		Req	Limit
Management (MGMT STS) • k8 Resource Type: StatefulSet • Sidecar name: NA • svc: ndbmgmdsvc	90Mi	100Mi	90Mi	100Mi	**2**(mgmReplicaCount)	360Mi	400Mi
NDB Data Store (NDB STS) • k8 Resource Type: StatefulSet • Sidecar name: db-executor-svc • svc : ndbmtdsvc	90Mi	100Mi	180Mi	1Gi	**2** (ndbReplicaCount)	540Mi	2200Mi
Replication (SQL STS - mysqld) • k8 Resource Type: StatefulSet • Sidecar name: init-sidecar • svc : ndbmysqldsvc	90Mi	100Mi	180Mi	200Mi	**2** 2/4/6 (Depending on GR setup of 2/3/4 site) (apiReplicaCount)	540Mi	600Mi
App Access (SQL STS - appmysqld) • k8 Resource Type StatefulSet • Sidecar name: init-sidecar • svc : ndbappmysqldsvc	90Mi	100Mi	90Mi	100Mi	**2** (ndbappReplicaCount)	360Mi	400Mi
Monitor Service • Sidecar name: NA • k8 Resource Type: Deployment • svc: db-monitor-service	90Mi	100Mi	NA	NA	**1**	90Mi	100Mi
Replication Service • Sidecar name: init-discover-sql-ips • k8 Resource Type: Deployment • svc: db-replication-service	90Mi	1000Mi	90Mi	100Mi	**1**	180Mi	1100Mi
GR Recovery Resources • Used for performing Disaster Recovery	90Mi	1000Mi	90Mi	100Mi		180Mi	1100Mi
Backup Manager Service • Sidecar name: NA • k8 Resource Type: Deployment • svc: db-backup-manager-svc	90Mi	100Mi	NA	NA	**1**	90Mi	100Mi
postUpgradeJob						90Mi	100Mi
preUpgradeJob							

Table 8 (Cont.) cnDBTier Profile Ephemeral Storage

helm test • Sidecar name: NA • k8 Resource Type: Job **Note:** This will be created only when "helm test" is performed							
Totals					11	2430Mi	6100Mi

> **Note**
>
> - In case of shared cnDBTier between Console and NF, Console DB Profile sizing needs to be included in NF DB Profile sizing.
> - cnDBTier custom values.yaml file has been included along with Console package from 23.3.0 release. The yaml file reflects the above cnDBTier profile and is packaged along with the Console custom values.yaml file used for installation. For installing the DBTier, Please refer to the *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide.*

F.1 Support for IPv4 or IPV6 Configuration for CNC Console

CNC Console supports following deployment modes:

1. Single Stack with IPv4 only
2. Single Stack with IPv6 only
3. DualStack with IPv4 Preferred
4. DualStack with IPv6 Preferred

Procedure

To deploy the CNC Console in any of these supported modes:

1. Before CNC Console deployment, update *occncc_custom_values<version>.yaml* file and set the **cncc-iam.kc.preferIpv6Stack.enabled** flag to desired value as mentioned in the below table.
2. After CNC Console is deployed, the following CNC Console services must be backed up, deleted and reapplied with preferred IP_Family configuration as mentioned in the following table:

> **Note**
>
> Below services have 'cncc' as the release name cncc-acore-ingress-gateway cncc-iam-ingress-gateway cncc-iam-kc-http cncc-mcore-cmservice cncc-mcore-ingress-gateway
> These services do not have any clusterIP assigned, and hence can be ignored. cncc-acore-igw-cache cncc-mcore-igw-cache cncc-iam-igw-cache cncc-iam-kc-headless.

The following table outlines the changes to be made before and after deploying the CNC Console for each deployment mode:

Table 9 Single or Dual Stack IP Configuration

Cluster Deployment Mode	CNC Console Deployment Mode	CNC Console custom values.yaml (PreInstallation Step)	Service configuration with preferred IP Family (PostInstallation Step)

Single Stack with IPv4 only	NA	`cncc-iam:`
		`  kc:`
		
		`preferIpv6Stack:`
		`  enabled:`
		`false`
			This will lead to no changes in the service file. Depending on the cluster, IPs will be assigned.

Table 9 (Cont.) Single or Dual Stack IP Configuration

Cluster Deployment Mode	CNC Console Deployment Mode	CNC Console custom values.yaml (PreInstallation Step)	Service configuration with preferred IP Family (PostInstallation Step)

Single Stack with IPv6 only	NA	`cncc-iam:`
		`  kc:`
		
		`preferIpv6Stack:`
		`  enabled:`
		`true`
			This will lead to no changes in the service file. Depending on the cluster, IPs will be assigned.

DualStack with IPv4 Preferred	IPv4 Preferred	`cncc-iam:`
		`  kc:`
		
		`preferIpv6Stack:`
		`  enabled:`
		`false`
			This will lead to no changes in the service file. Depending on the cluster, IPs will be assigned.

	IPv6 Preferred	`cncc-iam:`
		`  kc:`
		
		`preferIpv6Stack:`
		`  enabled:`
		`true`
			`spec:`
			
			`ipFamilyPolicy:`
			`RequireDualStack`
			`  ipFamilies:`
			`  - IPv6`
			`  - IPv4`

DualStack with IPv6 Preferred	IPv4 Preferred	`cncc-iam:`
		`  kc:`
		
		`preferIpv6Stack:`
		`  enabled:`
		`false`
			`spec:`
			
			`ipFamilyPolicy:`
			`RequireDualStack`
			`  ipFamilies:`
			`  - IPv4`
			`  - IPv6`

	IPv6 Preferred	`cncc-iam:`
		`  kc:`
		
		`preferIpv6Stack:`
		`  enabled:`
		`true`
			This will lead to no changes in the service file. Depending on the cluster, IPs will be assigned.

> **Note**
>
> Run the following commands to edit the service:
>
> ```
> kubectl edit svc -n <cncc_namespace> <cncc_service>
> ```
>
> For example:
>
> ```
> kubectl edit svc -n cncc -n cncc-iam-ingress-gateway
> ```

Sample CNC Deployment with IPv6 on DualStack with IPv4 Preferred Setup

Consider an example where you want to deploy CNC Console preferred with IPv6 on DualStack with IPv4 Preferred setup. This example describes the configuration changes required for single service of CNC Console after the deployment. You must follow a similar procedure for rest of the CNC Console services. Here:

- CNC Console deployment namespace: **cncc**
- CNC Console release name: **cncc**
- CNC Console service to be edited: **cncc-iam-ingress-gateway**
- Infrastructure Deployment Mode: **DualStack with IPv4 Preferred**
- CNC Console Deployment Mode: **IPv6 Preferred**

Procedure:

1. Run the following command to verify that cncc-iam-ingress-gateway service is set to IPv4 address:

   ```
   kubectl get svc cncc-iam-ingress-gateway -n cncc
   ```

   Output:

   ```
   NAME                                 TYPE           CLUSTER-IP
   EXTERNAL-IP     PORT(S)             AGE
   cncc-iam-ingress-gateway             LoadBalancer   1x.xxx.xx.xx
   1x.xxx.xx.xx    30085:32556/TCP     5h8m
   ```

2. Run the following command to take a backup of cncc-iam-ingress-gateway service and duplicate it for recovery purpose:

   ```
   kubectl get svc cncc-iam-ingress-gateway -n cncc -o yaml > cncc-iam-
   ingress-gateway.yaml;

   cp cncc-iam-ingress-gateway.yaml cncc-iam-ingress-gateway_orig.yaml;
   ```

3. Run the following command to delete the original cncc-iam-ingress-gateway service:

   ```
   kubectl delete svc cncc-iam-ingress-gateway -n cncc
   ```

4. Run the following command to edit the cncc-iam-ingress-gateway.yaml file generated as part of step 2:

   ```
   vim cncc-iam-ingress-gateway.yaml
   ```

5. Update following in the cncc-iam-ingress-gateway.yaml:

   a. Delete **clusterIPs** and **clusterIP** fields completely.

   ```
   # Remove the clusterIP and clusterIPs field
   ...
   spec:
     ...
     clusterIP: 1x.xxx.xx.xx
     clusterIPs:
     - 1x.xxx.xx.xx
   ...
   ```

   b. Under **spec.ipFamilyPolicy** set **RequiredDualStack** and under **spec.ipFamilies** set **[- IPv6, - IPv4]** as follows:

   ```
   apiVersion: v1
   kind: Service
   metadata:
     ...
     name: cncc-iam-ingress-gateway
     namespace: cncc
   spec:
     allocateLoadBalancerNodePorts: true
     ...
     ...
     ipFamilies:
     - IPv6
     - IPv4
     ipFamilyPolicy: RequireDualStack
     ...
     ...
     sessionAffinity: None
     type: LoadBalancer
   status:
     loadBalancer: {}
   ```

6. Run the following command to apply the cncc-iam-ingress-gateway.yaml file:

   ```
   kubectl apply -f cncc-iam-ingress-gateway.yaml -n cncc
   ```

7. Run the following command to check the service output. You should find the service with updated the IPv6 address:

   ```
   kubectl get svc cncc-iam-ingress-gateway -n cncc
   ```

   Output:

   ```
   NAME                                TYPE           CLUSTER-IP
   EXTERNAL-IP             PORT(S)             AGE
   ```

```
cncc-iam-ingress-gateway             LoadBalancer   xxxx:0:0:2::xxxx
xxxx:b400:605:xxx::3     30085:32556/TCP     5h8m
```

G.1 Fault Recovery Procedures - DB Backup and Restore

Introduction

This section describes the backup and restore procedures specific to Console. The instructions are for the operator to take Console database's backup and restore the database on a different cluster. These instructions can be used for Fault Recovery of Console and are part of the Fault Recovery guide.

The commands used for these procedures are given by the MySQL NDB Cluster.

Prerequisite

See the **Preupgrade Tasks** and Pre-rollback Tasks sections before performing backup or restore procedure.

Backup Procedure

CNC Console Database Backup
If the Console database backup is required, do the following:

Periodically keep backup of CNC Console Database using the DB backup procedure and store it. For more information see CNCC IAM DB Backup .

CNC Console Deployment Backup

- Keep a backup of the *occncc_custom_values_<version>.yaml* file that was used for last installation or upgrade of Console.
- Docker images used during the last installation or upgrade must be retained in the external data source.

Restore Procedure

CNC Console Database Restore

> **Note**
>
> Not applicable for OCI deployment.

Follow the restore procedure decsribed in the CNCC IAM DB Rollback or Restore section.

CNC Console Deployment Restore

To restore the CNC Console Deployment:

- Run the following command to uninstall the existing CNC Console deployment (if needed)

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

  For example:

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

- Use the backed up copy of *occncc_custom_values_<version>.yaml* file taken as part of CNC Console custom_values.yaml Backup to install the Console.

H.1 Restoring CNC Console

Perform this procedure to restore CNC Console.

Prerequisites:

- Take a backup of the `occncc_custom_values_<version>.yaml` file that was used for installing CNC Console.
- Take a backup of the CNC Console database and restore the database as described in Fault Recovery Procedures - DB Backup and Restore section.

Corruption in CNC Console Deployment

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

Console Deployment Corruption

If Console deployment is corrupted, do the following:

1. Run the following command to uninstall the corrupted CNCC deployment deployment:

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

   Where,

   - `<deployment name>` is a name used to track this installation instance.

   Example:

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

2. Install Console using the backed up copy of the `occncc_custom_values_<version>.yaml` file.
   For information about installing CNC Console using Helm, see *Oracle Communication Cloud Native Configuration Console Installation and Upgrade Guide*.