Oracle® Communications

Cloud Native Core Console Installation Guide

Release 1.3.0
F35169-01
September 2020

Oracle Communications Cloud Native Core Console Installation Guide, Release 1.3.0

F35169-01

Contents

My Oracle Support

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

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

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

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

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

What's New in This Guide

This section introduces the documentation updates for Release 1.3.0 in Cloud Native Core Console.

New and Updated Features in Release 1.3.0:

- Helm test
- Supports latest version of NFs

New and Updated Features in Release 1.2.1:

- CNCC Core and CNC IAM Configuration for Aspen Service Mesh (ASM)
- CNCC Core and CNC IAM Configuration for Operations Services Overlay (OSO)

1 Installation Overview

This document contains procedures to install the Cloud Native Core Console (CNCC). The CNCC provides user interface for the configuration parameters of Service Communication Proxy (SCP), Network Repository Function (NRF), Policy Control Function (PCF), Cloud Native Policy and Charging Rules Function (CNPCRF) and Unified Data Repository (UDR) network functions.

Reference

Refer the following documents for more information:

- Service Communication Proxy (SCP) Cloud Native User's Guide
- Network Repository Function (NRF) Cloud Native User's Guide
- Cloud Native Core Policy User's Guide
- Unified Data Repository (UDR) Cloud Native User's Guide
- Network Repository Function (NRF) Cloud Native Installation and Upgrade Guide
- Service Communication Proxy (SCP) Cloud Native Installation Guide
- Unified Data Repository (UDR) Cloud Native Installation and Upgrade Guide
- Cloud Native Core Policy Installation Guide

Acronyms

Table 1-1 Acronyms

Terms	Definition
CNCC	Cloud Native Core Console
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity Access Management
LDAP	Lightweight Directory Access Protocol
NRF	Network Repository Function
OSDC	Oracle Software Delivery Cloud
SCP	Service Communication Proxy
SAML	Security Assertion Markup Language
UDR	Unified Data Repository
UE	User Equipment

2
CNC Console Package

Prerequisite

- The user must have their own repository for storing the CNCC images and repository which must be accessible from the Kubernetes cluster.

Installation Preparation

The following table describes the steps to download the CNCC Images and Helm files from OSDC (Oracle Software Delivery Cloud).

1. **Download the CNCC package file:**

   Customers are required to download the CNCC package file from OSDC. The package is named as follows:

   `cncc-pkg-<marketing-release-number>.tgz`
   Example:

   `cncc-pkg-1.3.0.0.0.tgz`

2. **Untar the CNCC package file:**

   Untar the cncc package to the specific repository:

   `tar -xvf cncc-pkg-<marketing-release-number>.tgz`
   The package file consists of the following:

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

   b. Helm Chart of CNCC IAM: the tar ball contains Helm Chart and templates `cncc-iam-<tag>.tgz`

   c. Helm File of CNCC Core: tarball contains Helm charts and templates `cncc-core-<tag>.tgz`

   d. Readme txt File `Readme.txt`

   Example :

   List of contents in cncc-pkg-1.3.0.tgz :

   cncc-pkg-1.3.0.tgz

   |_ _ _ _ _ _ cncc-core-1.3.0.tgz

   |_ _ _ _ _ _ cncc-iam-1.3.0.tgz

   |_ _ _ _ _ _ cncc-images-1.3.0.tar

   |_ _ _ _ _ _ Readme.txt

3. **Check the checksums:**
   Check the checksums of tarballs mentioned in `Readme.txt`

4. **Load the tarball to system:**

   Execute the following command to push the Docker images to docker repository:

   ```
   docker load --input <image_file_name.tar>
   Example
    docker load --input cncc-images-1.3.0.tar
   Sample Output:
   Loaded image: cncc/apigw-configurationinit:1.3.0
   Loaded image: cncc/apigw-configurationupdate:1.3.0
   Loaded image: cncc/cncc-apigateway:1.3.0
   Loaded image: cncc/cncc-cmservice:1.3.0
   Loaded image: cncc/cncc-iam:1.3.0
   Loaded image: cncc/nf-test:1.3.0
   ```

5. **Push docker files to Docker registry:**
   Execute the following command to push the docker files to docker registry:

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

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

6. **Check if all the images are loaded:**
   Execute the following command to search helm chart:

   ```
   docker images
   ```

7. **Push helm charts to helm repository:**
   Execute the following command to push the helm charts to helm repository:

   ```
   helm push --force <helm_repo> <image_name>.tgz
   ```

8. **Download the CNCC custom templates package file:**
   Execute the following command to download the CNCC custom templates package file:

   ```
   cncc-custom-configtemplates-<marketing-release-number>.zip
   Example:
   cncc-custom-configtemplates-1.3.0.zip
   ```

9. **Unzip the CNCC custom templates package file:**
   Unzip the cncc custom templates package:

   ```
   The package file consists of the following:
   CNCC IAM custom values file : custom-cncc-iam_values_<version>.yaml
   CNCC Core custom values file : custom-cncc-
   core_values_<version>.yaml
   CNCC IAM DB sql file : cnccdb_<version>.sql
   ```

```
Example:

unzip cncc-custom-configtemplates-1.3.0.zip
Archive:  cncc-custom-configtemplates-1.3.0.zip
   creating: cncc-custom-configtemplates-1.3.0/
  inflating: cncc-custom-configtemplates-1.3.0/custom-cncc-
iam_values_1.3.0.yaml
  inflating: cncc-custom-configtemplates-1.3.0/cnccdb_1.3.0.sql
  inflating: cncc-custom-configtemplates-1.3.0/custom-cncc-
core_values_1.3.0.yaml
```

Verify and Create Kubernetes Namespace

This section explains how user can verify whether a required namespace exists in system or not. If namespace does not exist, user must create it.

> **Note:**
>
> CNCC can be installed at NF specific namespaces also.For that replace **cncc** namespace with custom namespace.

Procedure

1. Verify whether the required namespace already exists in system by executing the following command:

   ```
   $ kubectl get namespaces
   ```

2. If the namespace does not exist,then create the namespace by executing following command:

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

   Example:

   ```
   $ kubectl create namespace cncc
   ```

3
CNC Console IAM Installation Instructions

Prerequisites for CNC Console IAM

Following are the prerequisites for the installation of CNC Console IAM:

1. Create MySQL Database and User
2. Populate CNCC Database with CNCC IAM Tables
3. Create a Kubernetes Secret for MySQL
4. Create a Kubernetes Secret for Admin User
5. CNCC IAM Secret Configuration to Enable HTTPS

Create MySQL Database and User

This section explains how to create CNCC user and CNCC database.

> **Note:**
>
> **CNCC DB Naming Convention**
>
> As the CNCC instances cannot share the same database, operator has to provide a unique name to the CNCC DB in the DB Tier either limited to a site or spanning across sites.
>
> Format for database name :
>
> ```
> <database-name>_<site-name>_<cluster>
> ```
>
> Example database name:
>
> ```
> cnccdb_site1_cluster1
> ```
>
> Naming Convention: Start with alphabet, can contain alpha numeric, underscore and cannot be longer than 64 characters.
>
> In this installation guide, the commands use "cnccdb" as sample db name. The sample db name "cnccdb" must be replaced with the dbname that the user chooses as per naming conventions defined by this note.

1. Login to the server or machine which has permission to access the SQL nodes of NDB cluster.
2. Connect to the SQL nodes of NDB cluster one by one.

3. Execute the following command to login to the MySQL prompt using root permission or user, which has permission to create users with permissions:
   `mysql -h -uroot -p`

   > **Note:**
   >
   > After writing the command mentioned above, user must enter MySql password.

4. Check whether CNCC user already exists. If user does not exist, create a CNCC user by executing following commands:
   a. Execute `$ SELECT User FROM mysql.user;` to list the users.
   b. If user does not exist, create the new user by executing `$ CREATE USER '<CNCC User Name>'@'%' IDENTIFIED BY '<CNCC Password>';`
5. Check if CNCC database already exists. If the database does not exist, create a CNCC database and provide permissions to CNCC user created in the previous step:
   a. Execute `$ show databases;` to check if database exists.
   b. Drop the existing cnccdb database by executing the following command:

      ```
      DROP DATABASE cnccdb;
      ```

   c. Execute `$ CREATE DATABASE IF NOT EXISTS <CNCC Database>;` for Database creation.
   d. Grant permission to cncc-user by executing the following command:

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

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

```
# Login to MySql prompt:
$ mysql -u root -p
Check user already exists or not:
$ SELECT User FROM mysql.user;
# In case, user already exists, move to next step. Command to create
new user is as mentioned below:
$ CREATE USER 'cnccusr'@'%' IDENTIFIED BY 'cnccpasswd'
# Command to check if database exists:
$ show databases;
# Drop the existing cnccdb database:
$ DROP DATABASE cnccdb;
# Database creation for CNCC:
$ CREATE DATABASE IF NOT EXISTS cnccdb CHARACTER SET utf8;
#Granting permission to user:
$ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE
TEMPORARY TABLES, DELETE, UPDATE, EXECUTE ON cnccdb .* TO'cnccusr'@'%';
```

Populate CNCC Database with CNCC IAM Tables

The user must load the CNCC database created with `cnccdb_<version>.sql` file provided in the `cncc-custom-configtemplate`package file. This section describes how to populate CNCC database with CNCC IAM tables.

1. Load the database with tables from `cnccdb_<version>.sql`. Ensure `cnccdb_<version>.sql` is in `/home/admusr/`directory of the MySql Query Node.

   ```
   mysql -u <username> -p <databasename> cnccdb_<version>.sql
   ```

   > **Note:**
   >
   > The user must enter the mysql password.

2. Verify the tables are loaded into the database using command:

   ```
   $ use <databasename>;
   ```

   ```
   $ show tables;
   ```

   > **Note:**
   >
   > It shows a list of 93 tables related to CNCC-IAM.

3. Exit from MySql Query Node using following command:
   `$ exit;`

Example to demonstrate loading of cnccdb with tables from `cnccdb_<version>.sql`:

```
#mysql -h 127.0.0.1 -uroot -pNextGenCne cnccdb < /home/admusr/
cnccdb.sql
#mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql>use cnccdb;
mysql> show tables;
```

CNCC IAM MySQL Secret Configuration

This section describes how to create a k8s secret for MySQL.

1. Execute the following command to create the k8s secret for MySQL:

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

2. Execute the following command to verify the secret creation:
   ```
   $ kubectl describe secret <database secret name> -n <Namespace of
   MYSQL secret>
   ```

Example:

```
$ kubectl create secret generic cncc-db-secret --from-
literal=dbUserNameKey=root --from-
literal=dbPasswordKey=mypass -n cncc
$ kubectl describe secret cncc-db-secret -n cncc
```

CNCC IAM Secret Configuration for Default (Admin) User

This section describes how to create k8s secret for default (admin) user's password.

1. Execute the following command to create the k8s secret for admin user:

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

2. Execute the following command to verify the secret creation:
   `$ kubectl describe secret <secret name> -n <namespace>`

   Example:

   ```
   $ kubectl create secret generic cncc-iam-secret
           --from-literal=iamAdminPasswordKey=cncciampasswordvalue --
   namespace cncc
   $ kubectl describe secret cncc-iam-secret -n cncc
   ```

CNCC IAM Secret Configuration to Enable HTTPS

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

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

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

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

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

1. Create a secret by executing the following command:

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

   Example:

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

2. On successfully executing the above command, the following message will be displayed:
   secret/cncc-iam-ingress-secret created
3. Execute the following command to verify the secret creation: :

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

This section explains how to update the secrets for enabling HTTPS, if they already exist:

1. Create a secret by executing the following command:

```
$ kubectl create secret generic <secret-name> --
fromfile=<ssl_ecdsa_private_key.pem>
      --from-file=<rsa_private_key_pkcs1.pem> --
fromfile=<ssl_truststore.txt>
      --from-file=<ssl_keystore.txt> --from-file=<caroot.cer> --
fromfile=<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
      --fromfile=ssl_ecdsa_private_key.pem  --from-
file=rsa_private_key_pkcs1.pem
```

```
      --fromfile=ssl_truststore.txt --from-file=ssl_keystore.txt --
from-file=caroot.cer
      --fromfile=ssl_rsa_certificate.crt --from-
file=ssl_ecdsa_certificate.crt --dry-run -o yaml -n
      cncc | kubectl replace -f - -n cncc
```

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

CNCC IAM Configuration for Service Account

This section describes about the CNCC IAM Configuration for Service Account. CNCC IAM provides option to configure custom service account.

Sample CNCC IAM service account yaml file

cncc-iam-sa

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

```
  kind: Role
  name: cncc-iam-role
subjects:
- kind: ServiceAccount
  name: cncc-iam-sa
  namespace: cncc
```

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

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

```
global:

  # ********  Sub-Section Start: Ingress Gateway Global Parameters
********
  #
**********************************************************************
***

  serviceAccountName: cncc-iam-sa


.
```

2. For keycloak provide custom service account under *kc.keycloak.serviceAccount.name.serviceAccountName.*

```
 kc:
  keycloak:

    serviceAccount:
      # Specifies whether a service account should be created
      create: false
      # The name of the service account to use.
      # If not set and create is true, a name is generated using
the fullname template
      name: cncc-iam-sa
```

CNCC IAM Configuration for Aspen Service Mesh (ASM)

This section describes about CNCC IAM Configuration for Aspen Service Mesh (ASM).

1. **Annotation:**
   a. Add **traffic.sidecar.istio.io/excludeInboundPorts: "\"8081\""** annotation under *global.customExtention.lbDeployments.annotations* section in *cncc-iam_values.yaml* to disable mTLS on cncc-iam ingress container port.

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

```
**

  customExtension:
    lbDeployments:
      labels: {}
      annotations:
        traffic.sidecar.istio.io/excludeInboundPorts: "\"8081\""


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

b. Add **sidecar.istio.io/rewriteAppHTTPProbers: "\"true\""**under *global.customExtention.allResources.annotations* section in *cncc-iam_values.yaml* for readiness and liveness probe to work.

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

  customExtension:
    allResources:
      labels: {}
      annotations:
        sidecar.istio.io/rewriteAppHTTPProbers: "\"true\""


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

> **Note:**
>
> This is only required when deployed ASM is configured with *rewriteAppHTTPProbe* set to **false**.
>
> ```
> sidecarInjectorWebhook:
>   rewriteAppHTTPProbe: false           # To enable istio
> to rewrite probes when mTLS is enabled
> ```

2. **External MySQL DB:**

> **Note:**
>
> Skip this step, if
>
> CNCC IAM is deployed in **same namespace** as other 5G NFs and those NFs are already configure with MySQL service then user can use same service for CNCC IAM also.
>
> Refer *CNCC IAM configuration for MySQL* section to configure and populate db with required configuration.

a. Create Service & Endpoint for External MySQL instance.
Example: **service and endpoint**

```
apiVersion: v1
kind: Endpoints
metadata:
  name: mysql-connectivity-service-headless
  namespace: cncc
subsets:
- addresses:
  - ip: 10.75.203.49 # IP of cluster where MySQL is running
  ports:
  - port: 3306
    protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-connectivity-service-headless
  namespace: cncc
spec:
  clusterIP: None
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-connectivity-service
  namespace: cncc
spec:
  externalName: mysql-connectivity-service-
headless.cncc.svc.cluster.local
  sessionAffinity: None
  type: ExternalName
---
```

b. Create service-entry and destination rule for MySQL service.

Example: **service-entry and destination-rule**

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: mysql-external-se
  namespace: cncc
spec:
  hosts:
  - mysql-connectivity-service-headless.cncc.svc.cluster.local
  ports:
  - number: 3306
    name: mysql
    protocol: MySQL
  location: MESH_EXTERNAL
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: mysql-external-dr
  namespace: cncc
spec:
  host: mysql-connectivity-service-
headless.cncc.svc.cluster.local
  trafficPolicy:
    tls:
      mode: DISABLE
```

c. In *cncc-iam_values.yaml* under keycloak section provide MySQL service FQDN as follows:

```
dbName: cnccdb
dbHost: mysql-connectivity-service
dbPort: 3306
```

CNCC IAM Configuration for Operations Services Overlay (OSO)

This section describes about CNCC IAM Configuration for Operations Services Overlay (OSO).

Add Annotation **oracle.com/cnc: "\"true\""** under *global.customExtention.lbDeployments.annotations* section in *cncc-iam_values.yaml* 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
******************
  #
**********************************************************************
```

Installation Sequence for CNC Console IAM

The installation sequence for CNC Console IAM is:

1. **Installation Preparation**.
2. **Configure `custom-cncc-iam_values_<version>.yaml` file**.

   This includes configuring the following based on the deployment:

   a. Repository path

   b. cncc-iam details

   Note: Other configurations might be changed based on the deployment.
3. **cncc-iam deployment:**

   a. With helm repository

   b. With helm tar
4. **Verify cncc-iam deployment.**

Deployment of CNC Console IAM

1. **Search helm chart:**

   Execute the following command to check the version of the helm chart installation.
   `helm search <release_name>`

   ```
   Example: helm search cncc-iam

   NAME                  CHART VERSION  APP VERSION  DESCRIPTION
   ocspf-helm-repo/cncc-iam  3.0.0        8.0.1     Open Source
   Identity and Access
                                                    Management For
   Modern App
   ```

2. **Prepare `custom-cncc-iam_values_<version>.yaml` file**:
   Prepare a `custom-cncc-iam_values_<version>.yaml`file with the required parameter information.
3. **Deploy cncc-iam:**

   Installation using helm repository

   Execute the following command:

   ```
   For helm 2 based:
   helm install --name <release_name> <helm-repo> -f custom-cncc-
   ```

```
iam_values_<version>.yaml
--namenamespace<deployment<namespace_name> --version <helm_version>
For helm 3 based:
helm install <release_name> <helm-repo> -f custom-cncc-
iam_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 customer configuration

Example:

```
For helm 2 based:
helm install --name cncc-iam ocscp-helm-repo/ocscp -f custom-cncc-
iam_values_1.3.0.yaml --namenamespace
cncc-iam --version 1.3.0
For helm 3 based:
helm install cncc-iam ocscp-helm-repo/ocscp -f custom-cncc-
iam_values_1.3.0.yaml --namespace
cncc-iam --version 1.3.0
```

Note: Update dbVendor, dbHost , dbName fields in `custom-cncc-iam_values_<version>.yaml`

Example:

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

Installation using helm tar

Execute the following command:

```
For helm 2 based:
helm install --name cncc-iam -f custom-cncc-
iam_values_<version>.yaml  --name namespace <namespace>
<chartpath>./<chart>.tgz
For helm 3 based:
helm install cncc-iam -f custom-cncc-iam_values_<version>.yaml  --
namespace <namespace> <chartpath>./<chart>.tgz
```

Example:

```
For helm 2 based:
helm install --name cncc-iam -f custom-cncc-iam_values_1.3.0.yaml
--namenamespace cncc-iam ./cncc-iam.tgz
For helm 3 based:
```

```
helm install cncc-iam -f custom-cncc-iam_values_1.3.0.yaml --
namespace cncc-iam ./cncc-iam.tgz
```

4. **Check repository status:**

   Execute the following command to check the deployment status.
   `helm status <release_name>`

5. **Check service status:**

   Check if all the services are deployed and running:
   `kubectl -n <namespace_name> get services`

   Example:

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

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

6. **Check pod status :**

   Check if all the pods are up and running by executing the following command:
   `kubectl -n <namespace_name> get pods`

   Example:

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

NAME	READY	STATUS	RESTARTS	AGE
cncc-iam-kc-0	1/1	Running	0	44h
cncc-iam-ingress-gateway-6748d55f98-szdqm	1/1	Running	0	12h

CNC Console IAM Microservices

CNC Console IAM has three microservices, which are responsible for Identity Access Management:

- cncc-iam-kc-headless
- cncc-iam-kc-http
- cncc-iam-ingress-gateway

Following is an example of services CNC Console IAM offers:

Table 3-1 CNC Console IAM Microservices

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

CNC Console IAM Sample Custom Values

`custom-cncc-iam_values_1.3.0.yaml` with helm chart version 1.3.0

```
############################################################
#            Section Start: global attributes              #
############################################################

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

  # user needs to set imageRepository to the repository where the
images are kept
  dockerRegistry: ocspf-registry.us.oracle.com:5000/ocscp

  customExtension:
    allResources:
      labels: {}
      annotations: {}
      #  sidecar.istio.io/rewriteAppHTTPProbers: "\"true\""

    lbServices:
      labels: {}
      annotations: {}

    lbDeployments:
      labels: {}
      annotations: {}
      #  traffic.sidecar.istio.io/excludeInboundPorts: "\"8081\""
      #  oracle.com/cnc: "\"true\""

    nonlbServices:
      labels: {}
      annotations: {}

    nonlbDeployments:
      labels: {}
```

```
    annotations: {}

  # Helm test related configurations
  test:
    nfName: cncc-iam
    image:
      name: cncc/nf_test
      tag: helm-tag
    config:
      logLevel: WARN
      timeout: 40

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

  # ********  Sub-Section Start: Ingress Gateway Global Parameters
********
  #
********************************************************************************

  # 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)
  publicHttpSignalingPort: 8080
  publicHttpsSignallingPort: 8443

  serviceAccountName: ""

  # Specify type of service - Possible values are :- ClusterIP,
NodePort, LoadBalancer and ExternalName
  type: LoadBalancer
  # Enable or disable IP Address allocation from Metallb Pool
  metalLbIpAllocationEnabled: true

  # Address Pool Annotation for Metallb
  metalLbIpAllocationAnnotation: "metallb.universe.tf/address-pool: oam"

  # 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
  staticIpAddressEnabled: false
  staticIpAddress: 10.75.212.60

  # If Static node port needs to be set, then set staticNodePortEnabled
flag to true and provide value for staticNodePort
  # Else random node port will be assigned by K8
  staticNodePortEnabled: true
  staticHttpNodePort: 30085
  staticHttpsNodePort: 30053

  nodeSelector:
    nodeKey: ""
```

```
    nodeValue: ""

  k8sResource:
    container:
      prefix: ""
      suffix: ""

# ********  Sub-Section End: Ingress Gateway Global Parameters ********
# ************************************************************************

#############################################################
#            Section End  : global attributes           #
#############################################################

#############################################################
#            Section Start  : IAM  attributes           #
#############################################################

kc:
  keycloak:
    image:
      name: cncc/cncc-iam
      tag: helm-tag
      pullPolicy: Always

    ## Username for the initial CNCConsole-IAM admin user
    username: admin

    # Specifies an existing secret to be used for the admin password
    existingSecret: cncc-iam-secret

    # The key in the existing secret that stores the password
    existingSecretKey: iamAdminPasswordKey

    serviceAccount:
      # Specifies whether a service account should be created
      create: false
      # The name of the service account to use.
      # If not set and create is true, a name is generated using the
fullname template
      name:

    ## Persistence configuration
    persistence:
      # The database vendor. Can be either "mysql", "mariadb", or "h2"
      dbVendor: mysql

      ## The database name, host and port
      ## If dbVendor is 'mysql", then database should be created in
mysql prior to installing cncn-iam
      dbName: cnccdb
      dbHost: ""
      dbPort: ""

      ## Database Credentials are loaded from a Secret residing in the
```

```
same Namespace as keycloak.
      ## The Chart can read credentials from an existing Secret OR it
can provision its own Secret.

      ## Specify existing Secret
      # If set, specifies the Name of an existing Secret to read db
credentials from.
      existingSecret: cncc-db-secret
      existingSecretPasswordKey: dbPasswordKey  # read keycloak db
password from existingSecret under this Key
      existingSecretUsernameKey: dbUserNameKey  # read keycloak db user
from existingSecret under this Key

    service:
      # Labels and Annotations that are specific to service IAM are
added here.
      customExtension:
        labels: {}
        annotations: {}
      httpPort: 8285

    resources:
      limits:
        cpu: 2
        memory: 2Gi
      requests:
        cpu: 1
        memory: 1Gi


############################################################
##            Section End  : IAM attributes              #
############################################################


############################################################
##            Section Start  : Ingress Gateway attributes #
############################################################

ingress-gateway:

  image:
    # image name
    name: cncc/cncc-apigateway
    # tag name of image
    tag: helm-tag
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
    pullPolicy: Always

  initContainersImage:
    # inint Containers image name
    name: cncc/apigw-configurationinit
    # tag name of init Container image
    tag: helm-tag
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
```

```
    pullPolicy: Always

  updateContainersImage:
    # update Containers image name
    name: cncc/apigw-configurationupdate
    # tag name of update Container image
    tag: helm-tag
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
    pullPolicy: Always

  service:
    ssl:
      tlsVersion: TLSv1.2

      privateKey:
        k8SecretName: cncc-iam-ingress-secret
        k8NameSpace: cncc
        rsa:
          fileName: rsa_private_key_pkcs1.pem
        ecdsa:
          fileName: ssl_ecdsa_private_key.pem

      certificate:
        k8SecretName: cncc-iam-ingress-secret
        k8NameSpace: cncc
        rsa:
          fileName: ssl_rsa_certificate.crt
        ecdsa:
          fileName: ssl_ecdsa_certificate.crt

      caBundle:
        k8SecretName: cncc-iam-ingress-secret
        k8NameSpace: cncc
        fileName: caroot.cer

      keyStorePassword:
        k8SecretName: cncc-iam-ingress-secret
        k8NameSpace: cncc
        fileName: ssl_keystore.txt

      trustStorePassword:
        k8SecretName: cncc-iam-ingress-secret
        k8NameSpace: cncc
        fileName: ssl_truststore.txt

      initialAlgorithm: RSA256

    # Labels and Annotations that are specific to service
ingressgateway are added here.
    customExtension:
      labels: {}
      annotations: {}

    # Labels and Annotations that are specific to deployment
ingressgateway are added here.
```

```
  deployment:
    customExtension:
      labels: {}
      annotations: {}


  ports:
    # ContainerPort represents a network port in a single container
    containerPort: 8081
    containersslPort: 8443
    actuatorPort: 9090

  #Set the root log level
  log:
    level:
      root: WARN
      ingress: INFO
      cncc:
        security: INFO

  readinessProbe:
    # tells the kubelet that it should wait second before performing
the first probe
    initialDelaySeconds: 30
    # Number of seconds after which the probe times out
    timeoutSeconds: 3
    # specifies that the kubelet should perform a liveness probe every
xx seconds
    periodSeconds: 10
    # Minimum consecutive successes for the probe to be considered
successful after having failed
    successThreshold: 1
    # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up
    failureThreshold: 3

  livenessProbe:
    # tells the kubelet that it should wait second before performing
the first probe
    initialDelaySeconds: 30
    # Number of seconds after which the probe times out
    timeoutSeconds: 3
    # specifies that the kubelet should perform a liveness probe every
xx seconds
    periodSeconds: 15
    # Minimum consecutive successes for the probe to be considered
successful after having failed
    successThreshold: 1
    # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up
    failureThreshold: 3

  # Resource details
  resources:
    limits:
```

```
    cpu: 2
    initServiceCpu: 1
    updateServiceCpu: 1
    memory: 2Gi
    updateServiceMemory: 1Gi
    initServiceMemory: 1Gi
  requests:
    cpu: 1
    initServiceCpu: 0.5
    updateServiceCpu: 0.5
    memory: 1Gi
    updateServiceMemory: 0.5Gi
    initServiceMemory: 0.5Gi
  target:
    averageCpuUtil: 80

# Number of Pods must always be available, even during a disruption.
minAvailable: 1
# Min replicas to scale to maintain an average CPU utilization
minReplicas: 1
# Max replicas to scale to maintain an average CPU utilization
maxReplicas: 5


allowedCipherSuites:
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
  - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

cipherSuites:
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
  - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

# To Initialize SSL related infrastructure in init/update container
initssl: false
# Server Configuration for http and https support
enableIncomingHttp: true
enableIncomingHttps: false

ingressGwCertReloadEnabled: false
ingressGwCertReloadPath: /ingress-gw/certificate/reload

routesConfig:
# Examples for routes cncc-iam.
# Note: Enable addRequestHeader when ever https is enabled
#- id: cncc-iam_route
#  uri: http://cncc-iam-kc-http.cncc.svc.cluster.local:8285
#  path: /
```

```
  #  filters:
  #    prefixPath: /cncc/auth/admin
  #    #addRequestHeader: # Enable this filter only incase of https
  #    #- name: X-Forwarded-Proto
  #    #  value: https
  #- id: cncc-iam_route
  #  uri: http://cncc-iam-kc-http.cncc.svc.cluster.local:8285
  #  path: /cncc/auth/**
  #  #filters:
  #  #  addRequestHeader:
  #  #  - name: X-Forwarded-Proto
  #  #    value: https
  - id: cncc-iam_login_route
    uri: http://<helmrelease>-kc-http.<namespace>.<domain>:8285
    path: /
    filters:
      prefixPath: /cncc/auth/admin
    #  addRequestHeader: # Enable this filter only incase of https
    #  - name: X-Forwarded-Proto
    #    value: https
  - id: cncc-iam_route
    uri: http://<helmrelease>-kc-http.<namespace>.<domain>:8285
    path: /cncc/auth/**
    #filters:
    #  addRequestHeader: # Enable this filter only incase of https
    #  - name: X-Forwarded-Proto
    #    value: https

  # CNCC configuration
  cncc:
    # Enable security logs
    securityLogEnabled: true

#############################################################
##            Section End  : Ingress Gateway attributes  #
#############################################################
```

Note:

When CNCC IAM is enabled with HTTPS, all the routes must be appended with `addRequestHeader` filter. Then the updated `routesConfig` under `ingress` section in values.yaml will be as follows:

```
routesConfig:
- id: cncc-iam_login_route
  uri: http://<helmrelease>-kc-http.<namespace>.<domain>:8285
  path: /
  filters:
    prefixPath: /cncc/auth/admin
    addRequestHeader: # Enable this filter only incase of https
    - name: X-Forwarded-Proto
      value: https
- id: cncc-iam_route
  uri: http://<helmrelease>-kc-http.<namespace>.<domain>:8285
  path: /cncc/auth/**
  filters:
    addRequestHeader: # Enable this filter only incase of https
    - name: X-Forwarded-Proto
      value: https
```

CNC Console IAM Configuration Options During Deployment

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
kc.keycloak.image.name	<String>	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	M	Image Name to be used for cncc-iam micro service.

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
kc.keycloak.image.tag	<String>	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	M	Image Tag to be used for cncc-iam micro service.
kc.keycloak.image.pullpolicy	<String>	It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy	O	Pull Policy decides from where to pull the image.
kc.keycloak.username	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	It is the name of cncc-iam user as given by the user. Ex: admin
kc.keycloak.existingSecret	<String>	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	M	It specifies an existing secret name to be used for the admin password Ex: cncc-iam-secret
kc.keycloak.serviceAccount.create	<Boolean>	It can take either True or False value. By default, it is false.	O	Flag for creating service account.

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
kc.keycloak.serviceAccount.name	<String>	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	O	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>`
kc.keycloak.existingSecretKey	<String>	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	M	Applicable only if keycloak.existingSecret is provided. It is the key in the existing secret that stores the password Ex: iamAdminPasswordKey
kc.keycloak.persistence.dbVendor	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	It is the database vendor name Ex: mysql
kc.keycloak.persistence.dbName	<String>	Valid String	M	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 Ex: cnccdb

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
kc.keycloak.persistence.dbHost	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	It the hostname for persistence db Ex: mysql-sds.default.svc.cluster.local
kc.keycloak.persistence.dbPort	<Integer>	It can range from 0-65535	M	It is the db port for cncc-iam Ex: 3306
kc.keycloak.persistence.existingSecret	<String>	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	M	It specifies an existing secret to be used for mysql username and password Ex: cncc-db-secret
kc.keycloak.persistence.existingSecretPasswordKey	<String>	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	M	It is the key in the existing secret that stores the password Ex: dbPasswordKey
kc.keycloak.persistence.existingSecretUsernameKey	<String>	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	M	It is the key in the existing secret that stores the username Ex: dbUserNameKey
kc.keycloak.service.httpPort	<String>	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	O	It is the port number which makes cncc-iam service visible to other services running within the same K8s cluster
kc.keycloak.service.customExtension.labels	<String>		O	This can be used to add custom label(s) that are specific to service and will be created by cncc-iam helm chart.

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
kc.keycloak.service.customExtension.annotations	<String>		O	This can be used to add custom annotations(s) that are specific to service and will be created by cncc-iam helm chart.
global.dockerRegistry	<String>	It may contain lowercase letters, digits, and separators. A separator is defined as a period, one or two underscores, or one or more dashes.	M	It is the docker registry where cncc-iam images are present.
global.test.nfName	<String>	NF Name	M	Name of deployment for which helm test is done
global.test.image.name	<String>	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	M	Image name for the helm test container image
global.test.image.tag	<String>	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	M	Image version tag for helm test
global.test.config.logLevel	<String>	It can take values like: WARN, DEBUG, INFO, etc.	M	Log level for helm test pod
global.test.config.timeout	<String>	Range: 1-300 Unit:seconds	M	Timeout value for the helm test operation. If exceeded helm test will be considered as failure

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
global.publicHttpSignalingPort	<String>	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	M	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)
global.publicHttpsSignallingPort	<String>	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		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)
global.serviceAccountName	<String>	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	O	Service Account name
global.type	<String>	It can take ClusterIP, NodePort, LoadBalancer and ExternalName.	M	It specifies type of service - Possible values are :- ClusterIP, NodePort, LoadBalancer and ExternalName
metalLbIpAllocationEnabled	<Boolean>	It can take either True or False value. By default, it is false.	M	Enable or disable IP Address allocation from Metallb Pool
global.metalLbIpAllocationAnnotation	<String>	metallb.universe.tf/ address-pool: oam	M	Address Pool Annotation for Metallb

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
global.staticIpAddressEnabled	<Boolean>	It can take either True or False value. By default, it is false.	O	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
global.staticIpAddress	<String>		O	It is Static Ip and applicable only when ingress-gateway.global.staticNodePortEnabled is true.
global.staticNodePortEnabled	<Boolean>	It can take either True or False value. By default, it is false.	O	Node Port Enabled
global.staticHttpNodePort	<String>		O	It is Http Node Port and applicable only when ingress-gateway.global.staticNodePortEnabled is true.
global.nodeSelector.nodeKey	<String>		O	global node selector key
global.nodeSelector.nodeValue	<String>		O	global node value key
global.customExtension.allResources.labels	<String>	Custom Labels that needs to be added to both the subcharts of cncc-iam	O	This can be used to add custom label(s) to all k8s resources that will be created by cncc-iam helm chart.
global.customExtension.allResources.annotations	<String>	Custom Annotations that needs to be added to both the sub-charts of cncc-iam	O	This can be used to add custom annotation(s) to all k8s resources that will be created by cncc-iam helm chart.

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
global.customExtension.lbServices.labels	<String>	Custom Labels that needs to be added for both the sub-charts of that are considered as Load Balancer type	O	This can be used to add custom label(s) to all Load Balancer Type Services that will be created by cncc-iam helm chart.
global.customExtension.lbServices.annotations	<String>	Custom Annotations that needs to be added for both the subcharts of cncc-iam that are considered as Load Balancer type	O	This can be used to add custom annotation(s) to all Load Balancer Type Services that will be created by cncc-iam helm chart.
global.customExtension.lbDeployments.labels	<String>	Custom Labels that needs to be added for both the subcharts of cncc-iam which is of Load Balancer type	O	This can be used to add custom label(s) to all Deployments that will be created by cncc-iam helm chart which are associated to a Service which if of Load Balancer Type.
global.customExtension.lbDeployments.annotations	<String>	Custom Annotations that needs to be added to both the subcharts of cncc-iam which is of Load Balancer type	O	This can be used to add custom annotation(s) to all Deployments that will be created by cncc-iam helm chart which are associated to a Service which if of Load Balancer Type.
global.customExtension.nonlbServices.labels	<String>	Custom Labels that needs to be added to cncc-iam that are considered as not Load Balancer type	O	This can be used to add custom label(s) to all non-Load Balancer Type Services that will be created by cncc-iam helm chart.
global.customExtension.nonlbServices.annotations	<String>	Custom Annotations that needs to be added for both the subcharts of cncc-iam that are considered as not Load Balancer type	O	This can be used to add custom annotation(s) to all non-Load Balancer Type Services that will be created by cncc-iam helm chart.

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
global.customExtension.nonlbDeployments.labels	<String>	Custom Labels that needs to be added for both the subcharts of cncc-iam that are associated to a Service which is not of Load Balancer type	O	This can be used to add custom label(s) to all Deployments that will be created by cncc-iam helm chart which are associated to a Service which if not of Load Balancer Type.
global.customExtension.nonlbDeployments.annotations	<String>	Custom Annotations that needs to be added for both the subcharts of cncc-iam that are associated to a Service which is not of Load Balancer type	O	This can be used to add custom annotation(s) to all Deployments that will be created bycncc-iam helm chart which are associated to a Service which if not of Load Balancer Type.
global.k8sResource.container.prefix	<String>	Value that will be prefixed to all the container names of Ingress-gateway.	O	This value will be used to prefix to all the container names of OCNRF.
global.k8sResource.container.suffix	<String>	Value that will be suffixed to all the container names of OCNRF.	O	This value will be used to suffix to all the container names of OCNRF.
ingress-gateway.image.name	<String>	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	M	Image Name to be used for "ingress-gateway" micro service
ingress-gateway.image.tag	<String>	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	M	Image Tag to be used for "ingress-gateway" micro service

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.image.pullPolicy	<String>	It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy	M	Pull Policy decides from where to pull the image.
ingress-gateway.initContainersImage.name	<String>	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	M	Image Name to be used for init container
ingress-gateway.initContainersImage.tag	<String>	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	M	Image tag to be used for init container
ingress-gateway.initContainersImage.pullPolicy	<String>	It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy	M	Pull Policy decides from where to pull the image.
ingress-gateway.updateContainersImage.name	<String>	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	M	Image Name to be used for update container

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.updateContainersImage.tag	<String>	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	M	Image tag to be used for update container
ingress-gateway.updateContainersImage.pullPolicy	<String>	It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy	M	Pull Policy decides from where to pull the image.
ingress-gateway.service.ssl.tlsVersion	<String>	Default Value is TLSv1.2	M	TLS Version
ingress-gateway.service.ssl.privateKey.k8SecretName	<String>	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	M	Name of the privatekey secret Ex: cncc-iam-ingress-secret
ingress-gateway.service.ssl.privateKey.k8NameSpace	<String>	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	M	Namespace of privatekey Ex: cncc
ingress-gateway.service.ssl.privateKey.rsa.fileName	<String>	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	M	rsa private key file name Ex: rsa_private_key_pkcs1.pem
ingress-gateway.service.ssl.privateKey.ecdsa.fileName	<String>	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	M	ecdsa private key file name Ex: ssl_ecdsa_private_key.pem

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.service.ssl.certificate.k8SecretName	<String>	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	M	Name of the certificate secret Ex: cncc-iam-ingress-secret
ingress-gateway.service.ssl.certificate.k8NameSpace	<String>	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	M	Namespace of certificate Ex: cncc
ingress-gateway.service.ssl.certificate.rsa.fileName	<String>	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	M	rsa certificate file name Ex: ssl_rsa_certificate.crt
ingress-gateway.service.ssl.certificate.ecdsa.fileName	<String>	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	M	ecdsa certificate file name Ex: ssl_ecdsa_certificate.crt
ingress-gateway.service.ssl.caBundle.k8SecretName	<String>	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	M	Name of the caBundle secret Ex: cncc-iam-ingress-secret
ingress-gateway.service.ssl.caBundle.k8NameSpace	<String>	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	M	Namespace of caBundle Ex: cncc

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.service.ssl.caBundle.fileName	<String>	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	M	rsa caBundle file name Ex: caroot.cer
ingress-gateway.service.ssl.initialAlgorithm	<String>	Default values is RSA256	M	
ingress-gateway.service.customExtension.labels	<String>	Custom Labels that needs to be added to ingress-gateway specific Service.	O	This can be used to add custom label(s) to ingress-gateway Service.
ingress-gateway.service.customExtension.annotations	<String>	Custom Annotations that needs to be added to ingress-gateway specific Services.	O	This can be used to add custom annotation(s) to ingress-gateway Service.
ingress-gateway.deployment.customExtension.labels	<String>	Custom Labels that needs to be added to ingress-gateway specific Deployment.	O	This can be used to add custom label(s) to ingress-gateway Deployment.
ingress-gateway.deployment.customExtension.annotations	<String>	Custom Annotations that needs to be added to ingress-gateway specific Deployment.	O	This can be used to add custom annotation(s) to ingress-gateway Deployment.
ingress-gateway.service.ssl.keyStorePassword.k8SecretName	<String>	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	M	Name of the keyStorePassword secret Ex: cncc-iam-ingress-secret
ingress-gateway.service.ssl.keyStorePassword.k8NameSpace	<String>	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	M	Namespace of keyStorePassword Ex: cncc

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.service.ssl.keyStorePassword.fileName	<String>	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	M	File name that has password for keyStore Ex: ssl_keystore.txt
ingress-gateway.service.ssl.trustStorePassword.k8SecretName	<String>	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	M	Name of the trustStorePassword secret Ex: cncc-iam-ingress-secret
ingress-gateway.service.ssl.trustStorePassword.k8NameSpace	<String>	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	M	Namespace of trustStorePassword Ex: cncc
ingress-gateway.service.ssl.trustStorePassword.fileName	<String>	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	M	File name that has password for trustStore Ex: ssl_truststore.txt
ingress-gateway.ports.containerPort	<String>	It can take value in the range: 0-65535.	M	ContainerPort represents a network port in a single container
ingress-gateway.ports.containersslPort	<String>	Default value is 8443	M	
ingress-gateway.ports.actuatorPort	<String>	Default value is 9090		
ingress-gateway.log.level.root	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc.	M	It is the level at which user wants to see the logs. E.g. WARN
ingress-gateway.log.level.ingress	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc. Default value is INFO	M	Log level for ingress logs

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.log.level.cncc.security	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc. Default value is INFO	M	Log level for cncc security logs
ingress-gateway.readinessProbe.initialDelaySeconds	<String>	It can take value in the range: 0-65535. Default value:30	M	It tells the kubelet that it should wait second before performing the first probe
ingress-gateway.readinessProbe.timeoutSeconds	<String>	It can take value in the range: 0-65535. Default value:3	M	It is the number of seconds after which the probe times out
ingress-gateway.readinessProbe.periodSeconds	<String>	It can take value in the range: 0-65535. Default value:10	M	It specifies that the kubelet should perform a liveness probe every xx seconds
ingress-gateway.readinessProbe.successThreshold	<String>	It can take value in the range: 0-65535. Default value:1	M	Minimum consecutive successes for the probe to be considered successful after having failed
ingress-gateway.readinessProbe.failureThreshold	<String>	It can take value in the range: 0-65535. Default value:3	M	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up
ingress-gateway.livenessProbe.initialDelaySeconds	<String>	It can take value in the range: 0-65535. Default value:30	M	It tells the kubelet that it should wait second before performing the first probe
ingress-gateway.livenessProbe.timeoutSeconds	<String>	It can take value in the range: 0-65535. Default value:3	M	It is the number of seconds after which the probe times out
ingress-gateway.livenessProbe.periodSeconds	<String>	It can take value in the range: 0-65535. Default value:15	M	It specifies that the kubelet should perform a liveness probe every xx seconds
ingress-gateway.livenessProbe.successThreshold	<String>	It can take value in the range: 0-65535. Default value:1	M	Minimum consecutive successes for the probe to be considered successful after having failed

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.livenessProbe.failureThreshold	<String>	It can take value in the range: 0-65535. Default value:3	M	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up
ingress-gateway.resources.limits.cpu	<String>	Valid floating point value between 0 and 1	M	It limits the number of CPUs to be used by the microservice.
ingress-gateway.resources.limits.initServiceCpu	<String>	Default value is 1	M	Init Container CPU Limit
ingress-gateway.resources.limits.updateServiceCpu	<String>	Default value is 1	M	Update Container CPU Limit
ingress-gateway.resources.limits.memory	<String>	Valid Integer value followed by Mi/Gi etc.	M	It limits the memory utilization by the "cncc-cmservice" microservice. By default, it is set to '2'.
ingress-gateway.resources.limits.updateServiceMemory	<String>	Default value is 1Gi	M	Update Container Memory Limit
ingress-gateway.resources.limits.initServiceMemory	<String>	1Gi	M	Init Container Memory Limit
ingress-gateway.resources.requests.cpu	<String>	Valid floating point value between 0 and 1	M	It limits the number of CPUs to be used by the "cncc-cmservice" microservice. By default, it is set to '2'.
ingress-gateway.resources.requests.initServiceCpu	<String>	Default value is 1	M	Init Container CPU Limit
ingress-gateway.resources.requests.updateServiceCpu	<String>	Default value is 1	M	Update Container CPU for requests

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.resources.requests.memory	<String>	Valid Integer value followed by Mi/Gi etc.	M	It limits the memory utilization by the "cncc-cmservice" microservice. By default, it is set to '2'.
ingress-gateway.resources.requests.updateServiceMemory	<String>	1Gi	M	Update Container Memory for requests
ingress-gateway.resources.requests.initServiceMemory	<String>	1Gi	M	Init Container Memory for requests
ingress-gateway.resources.target.averageCpuUtil	<String>	A value in between 0-100	M	It gives the average CPU utilization percentage.
ingress-gateway.minAvailable	<String>	It can take value in the range: 0-65535. Default value:1	M	It is the number of pods that must always be available, even during a disruption.
ingress-gateway.minReplicas	<String>	It can take value in the range: 0-65535. Default value:1	M	Min replicas to scale to maintain an average CPU utilization
ingress-gateway.maxReplicas	<String>	It can take value in the range: 0-65535. Default value:5	M	Max replicas to scale to maintain an average CPU utilization
ingress-gateway.initssl	<String>	It can take either True or False value. By default, it is false.	M	To Initialize SSL related infrastructure in init/update container
ingress-gateway.enableIncomingHttp	<String>	It can take either True or False value. By default, it is false.	M	Server Configuration for http and https support
ingress-gateway.enableIncomingHttps	<String>	It can take either True or False value. By default, it is false.	M	Server Configuration for http and https support

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.cipherSuites	<List[String]>	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA38 4 TLS_ECDHE_RSA_WITH_AES_256_GC M_SHA384 TLS_ECDHE_RSA_WITH_ CHACHA20_POLY1305_S HA256 TLS_DHE_RSA_WITH_AES_256_GCM_ SHA384 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA25 6 TLS_ECDHE_RSA_WITH_AES_128_GC M_SHA256	M, if ingressgateway.enableIncomingHttpsis true	Allowed CipherSuites for TLS1.2
ingress-gateway.ingressGwCertReloadEnabled	<boolean>	It can take either True or False value. Default value is true	M	
ingress-gateway.ingressGwCertReloadPath	<String>		M	
ingress-gateway.routesConfig.[].id	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	Routes to be added for cncc-iam ingress-gateway
ingress-gateway.routesConfig.[].uri	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	
ingress-gateway.routesConfig.[].path	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	
ingress-gateway.routesConfig[].order	<Integer>	Valid Integer value	O	

Name	Data type	Range Datatype	Mandatory(M)/ Optional(O)/ Conditional(C)DataType	DescriptionDataType
ingress-gateway.routesConfig.[].filters.addRequestHeader.[].name	<String>	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	M	
ingress-gateway.routesConfig.[].filters.addRequestHeader.[].value	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes. It component may not start or end with a separator	M	
ingress-gateway.cncc.securitylogEnabled	<boolean>	It can take either True or False value. By default, it is true	O	This flag is to enable/ disable security logs for cncc.
ingress-gateway.nodeselector.nodeKey	<String>		O	node selector value specific to chart (note this will be looked first and then if not present global node value will be picked)
ingress-gateway.nodeselector.nodeValue	<String>		O	node selector key specific to chart (note this will be looked first and then if not present global node key will be picked)

CNC Console IAM Access

CNC Console IAM can be accessed in the following ways:

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

# Ways to access CNCC IAM GUI
# 1] Node-IP and NodePort
http://10.75.xx.xx:30085/*
# 2] DNS Resolvable FQDN and NodePort
http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:30085/*
# 3] External LB-IP and ServicePort
http://10.75.xx.xx:8080/*
# 4] DNS Resolvable FQDN and ServicePort
http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:8080/*
```

CNC Console IAM Uninstall

CNC Console IAM can be uninstalled as follows:

To undeploy CNCConsole-IAM :

```
For Helm 2:
$ helm delete <deployment name> --purge
 Example:
$ helm delete cncc-iam --purge
For Helm 3:
$ helm uninstall <deployment name> --namespace <deployment namespace>
 Example:
$ helm uninstall cncc-iam --namespace cncc
```

4
CNC Console Core Installation Instructions

Prerequisites for CNC Console Core Installation

Following are the prerequisites for the installation of CNC Console Core:

- The NFs for which GUI is required must be deployed in the Kubernetes cluster.
- CNC Console IAM must be deployed.

CNCC Core Secret Configuration to Enable HTTPS

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

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

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

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

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

1. Create a secret by executing the following command:

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

Example:

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

2. On successfully executing the above command, the following message will be displayed:
   secret/cncc-core-ingress-secret created
3. Execute the following command to verify the secret creation:
   `$ kubectl describe secret cncc-core-ingress-secret -n cncc`

This section explains how to update the secrets for enabling HTTPS, if they already exist:

1. Create a secret by executing the following command:

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

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

CNCC Core Configuration for Service Account

This section describes about CNCC Core Configuration for Service Account. CNCC Core provides option to configure custom service account.

Sample CNCC Core service account yaml file

cncc-core-sa

```
## Service account yaml file for cncc-core
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cncc-core-sa
  namespace: cncc
  annotations: {}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: cncc-core-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-core-rolebinding
  namespace: cncc
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: cncc-core-role
subjects:
- kind: ServiceAccount
  name: cncc-core-sa
  namespace: cncc
```

Configure service account for ingress-gateway and keycloak in *cncc-core_values.yaml*

Provide custom service account for ingress-gateway and cmservice under *global.serviceAccountName* in *cncc-core_values.yaml* as follows:

cncc-core_values.yaml

```
global:
            serviceAccountName: cncc-core-sa
```

CNCC Core Configuration for Aspen Service Mesh (ASM)

This section describes about CNCC Core Configuration for Aspen Service Mesh (ASM).

1. **Annotations:**

   Add Annotation **traffic.sidecar.istio.io/excludeInboundPorts: "\"8081\""** under *global.customExtention.lbDeployments.annotations* section in *cncc-core_values.yaml* to disable mTLS on cncc-core ingress container port.

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

  customExtension:
    lbDeployments:
      labels: {}
      annotations:
        traffic.sidecar.istio.io/excludeInboundPorts: "\"8081\""

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

2. **Service Entry and Destination Rule**

   a. **For k8s cluster domain:**
   Create Destination rule to disable mTLS at cncc-iam service FQDN.

   Example: **Destination-Rule**

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: cncc-iam-exclude-mtls
  namesapce: cncc
spec:
  host: cncc-iam-ingress-gateway.cncc.svc.cluster.local
  trafficPolicy:
```

```
    tls:
      mode: DISABLE
---
```

b. **For custom domain:**
Create service-entry and destination rule to disable mTLS at cncc-iam domain.

Example: **Service-entry & Destination-rule**

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: cncc-iam-service-entry
  namesapce: cncc
spec:
  hosts:
  - ocnrf-cncc-iam # Custom CNCC IAM domain
  exportTo:
  - "."
  addresses:
  - 10.75.225.205 # IP of the k8s node where CNCC-IAM is deployed
  location: MESH_INTERNAL
  ports:
  - number: 30085
    name: http
    protocol: HTTP
  resolution: NONE
------------------------------
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: cncc-iam-exclude-mtls
  namesapce: cncc
spec:
  host: ocnrf-cncc-iam # Custom CNCC IAM domain
  trafficPolicy:
    tls:
      mode: DISABLE
------------------------------
```

CNCC Core Configuration for Operations Services Overlay (OSO)

This section describes about CNCC Core Configuration for Operations Services Overlay (OSO).

Add Annotation **oracle.com/cnc: "\"true\""** under *global.customExtention.lbDeployments.annotations* section in *cncc-core_values.yaml* 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
******************
  #
**************************************************************************
```

Installation Sequence for CNCC Core

Installation Sequence for CNCC Core:

1. **Installation Preparation.**
2. **Configure `custom-cncc-core_values.yaml` file.**

   This includes configuring the following based on the deployment:

   a. Repository path

   b. Domain and clusterdomain

   c. CNC Console details

   Note: Other configurations might be changed based on the deployment.
3. **CNC Console deployment:**

   a. With helm repository

   b. With helm tar
4. **Verify CNC Core deployment**

Deployment of CNCC Core

This procedure describes the steps to deploy CNCC Core. The below steps need to be executed from a server which has access to Kubectl and helm commands.

1. **Search helm chart:**

   Execute the following command to search helm chart.

   ```
   helm search <release_name>


   Example: helm search cncc-core

   NAME                  CHART VERSION  APP VERSION  DESCRIPTION

   ocspf-helm-repo/cncc-core 1.3.0      1.0        A Helm chart for
   CNC Console
   ```

2. **Prepare `custom-cncc-core_values.yaml` file:**

   Prepare a `custom-cncc-core_values.yaml` file with the required parameter information.

3. **Deploy CNCC Core:**

   Installation using helm repository

   Execute the following command:

   ```
   For helm 2 based:
   helm install --name <release_name> <helm-repo> -f custom-
   cncc-core_values.yaml --namenamespace<deployment<namespace_name> --
   version <helm_version>
   For helm 3 based:
   helm install <release_name> <helm-repo> -f custom-cncc-
   core_values.yaml --namespace <namespace_name> --version
   <helm_version>
   ```

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

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

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

   Example:

   ```
   For helm 2 based:
   helm install --name cncc-core ocscp-helm-repo/ocscp -f custom-cncc-
   core_values.yaml  --namenamespace cncc --version 1.3.0
   For helm 3 based:
   helm install cncc-core ocscp-helm-repo/ocscp -f custom-cncc-
   core_values.yaml --namespace cncc --version 1.3.0
   ```

   Installation using helm tar

   Execute the following command:

   ```
   For helm 2 based:
   helm install --name cncc-core -f custom-cncc-core_values.yaml --
   name namespace <namespace> <chartpath>./<chart>.tgz
   For helm 3 based:
   helm install cncc-core -f custom-cncc-core_values.yaml --namespace
   <namespace> <chartpath>./<chart>.tgz
   ```

4. **Check repository status**:

   Execute following command to check the deployment status.
   `helm status <release_name>`

5. **Check service status:**

   Check if all the services are deployed and running:

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

Example: `$ kubectl -n cncc get services`

cncc-core-cmservice	ClusterIP	10.233.13.43	<none>	8442/TCP	6m13s
cncc-core-ingress-gateway	LoadBalancer	10.233.11.14	10.75.182.79	8080:31417/TCP	6m13s

6. **Check pod status:**

   Check if all the pods are up and running by executing following command:

   ```
   kubectl -n <namespace_name> get pods
   ```
   Example:`$ kubectl -n cncc get pods`

NAME	READY	STATUS	RESTARTS	AGE
cncc-core-cmservice-7f8b57c5bf-p4gvw	1/1	Running	0	6m18s
cncc-core-ingress-gateway-5bfb8789cd-wls5p	1/1	Running	0	6m18s

CNCC Core Microservices

CNCC Core has two microservices:

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

Following is an example of services CNCC Core offers:

Table 4-1 CNCC Core Microservices

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
cncc-core-cmservice	ClusterIP	10.233.13.43	<none>	8442/TCP	6m13s
cncc-core-ingress-gateway	LoadBalancer	10.233.13.43	10.75.182.79	8080:31417/TCP	6m13s

CNCC Core Sample Custom Values

The **custom-cncc-core_values.yaml** file can also be downloaded from OHC.

```
############################################################
#            Section Start: global attributes              #
############################################################
global:

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

```
  # user needs to set imageRepository to the repository where the
images are kept
  dockerRegistry: ocspf-registry.us.oracle.com:5000/ocscp

  serviceAccountName: ""

  customExtension:
    allResources:
      labels: {}
      annotations: {}

    lbServices:
      labels: {}
      annotations: {}

    lbDeployments:
      labels: {}
      annotations: {}
      #  traffic.sidecar.istio.io/excludeInboundPorts: "\"8081\""
      #  oracle.com/cnc: "\"true\""

    nonlbServices:
      labels: {}
      annotations: {}

    nonlbDeployments:
      labels: {}
      annotations: {}
# Helm test related configurations
  test:
    nfName: cncc-core
    image:
      name: cncc/nf_test
      tag: helm-tag
    config:
      logLevel: WARN
      timeout: 40

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

  # ********  Sub-Section Start: Ingress Gateway Global Parameters
********

#************************************************************************
*

  # 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)
  publicHttpSignalingPort: 8080
  publicHttpsSignallingPort: 8443

  #Specify type of service - Possible values are :- ClusterIP,
NodePort, LoadBalancer and ExternalName
```

```
  type: LoadBalancer

  #Enable or disable IP Address allocation from Metallb Pool
  metalLbIpAllocationEnabled: true

  #Address Pool Annotation for Metallb
  metalLbIpAllocationAnnotation: "metallb.universe.tf/address-pool: oam"

  #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
  staticIpAddressEnabled: false
  staticIpAddress: ""

  #If Static node port needs to be set, then set staticNodePortEnabled
flag to true and provide value for staticNodePort
  #Else random node port will be assigned by K8
  staticNodePortEnabled: true
  staticHttpNodePort: 30075
  staticHttpsNodePort: 30043

  nodeSelector:
    nodeKey: ""
    nodeValue: ""

  k8sResource:
    container:
      prefix: ""
      suffix: ""

# ********  Sub-Section End: Ingress Gateway Global Parameters ********
#*************************************************************************
############################################################
#            Section End  : global attributes           #
############################################################

#####################################################################
#            Section Start  : cmservice attributes            #
#####################################################################
cmservice:
  envLoggingLevelApp: WARN

  image:
    # image name
    name: cncc/cncc-cmservice
    # tag name of image
    tag: helm-tag
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
    pullPolicy: Always

  # Resource details
  resources:
    limits:
      cpu: 2
```

```
        memory: 2Gi
      requests:
        cpu: 1
        memory: 1Gi

  # Deployment details
  deployment:
    customExtension:
      labels: {}
      annotations: {}

    envManageNF: SCP, NRF, UDR, POLICY
    # This is the name of product which appears as brand name and can
be used to mention site name as well.
    # envSystemName: CNCC - Site Name
    envSystemName: CNCC
    # This is the version of product which appears with brand name.
    envNFVersion: 1.3.0
    # This is the name of the Project that appears on the Window
    cmWindowName: CNCC
    # Applicable for POLICY deployment, this enables Import Export
buttons.
    # Make cmEnableImportExport : true in case of POLICY deployment
    cmEnableImportExport: false

    nodeSelectorEnabled: false
    nodeSelectorKey: zone
    nodeSelectorValue: app

    livenessProbe:
      initialDelaySeconds: 60
      periodSeconds: 3
      timeoutSeconds: 15
      successThreshold: 1
      failureThreshold: 3
    readinessProbe:
      initialDelaySeconds: 20
      timeoutSeconds: 3
      periodSeconds: 10
      successThreshold: 1
      failureThreshold: 3

    dependenciesLogging:
      - name: logging.level.org.springframework
        value: WARN
      - name: logging.level.io.undertow
        value: WARN

  service:
    customExtension:
      labels: {}
      annotations: {}
    type: ClusterIP

  servicePorts:
```

```
    cmServiceHttp: 8442
  containerPorts:
    monitoringHttp: 9000
    cmServiceHttp: 8442

#############################################################
#            Section End  : cmservice attributes        #
#############################################################

#####################################################################
#         Section Start  : ingress gateway attributes         #
#####################################################################
ingress-gateway:
  image:
    # image name
    name: cncc/cncc-apigateway
    # tag name of image
    tag: helm-tag
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
    pullPolicy: Always

  initContainersImage:
    # inint Containers image name
    name: cncc/apigw-configurationinit
    # tag name of init Container image
    tag: helm-tag
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
    pullPolicy: Always

  updateContainersImage:
    # update Containers image name
    name: cncc/apigw-configurationupdate
    # tag name of update Container image
    tag: helm-tag
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
    pullPolicy: Always

  service:
    ssl:
      tlsVersion: TLSv1.2

      privateKey:
        k8SecretName: cncc-core-ingress-secret
        k8NameSpace: cncc
        rsa:
          fileName: rsa_private_key_pkcs1.pem
        ecdsa:
          fileName: ssl_ecdsa_private_key.pem

      certificate:
        k8SecretName: cncc-core-ingress-secret
        k8NameSpace: cncc
        rsa:
          fileName: ssl_rsa_certificate.crt
        ecdsa:
```

```
            fileName: ssl_ecdsa_certificate.crt

        caBundle:
          k8SecretName: cncc-core-ingress-secret
          k8NameSpace: cncc
          fileName: caroot.cer

        keyStorePassword:
          k8SecretName: cncc-core-ingress-secret
          k8NameSpace: cncc
          fileName: ssl_keystore.txt

        trustStorePassword:
          k8SecretName: cncc-core-ingress-secret
          k8NameSpace: cncc
          fileName: ssl_truststore.txt

        initialAlgorithm: RSA256

    # Labels and Annotations that are specific to service
ingressgateway are added here.
    customExtension:
      labels: {}
      annotations: {}

    # Labels and Annotations that are specific to deployment
ingressgateway are added here.
  deployment:
    customExtension:
      labels: {}
      annotations: {}

  ports:
    # ContainerPort represents a network port in a single container
    containerPort: 8081
    containersslPort: 8443
    actuatorPort: 9090

  # Set the root log level
  log:
    level:
      root: WARN
      ingress: INFO
      cncc:
        security: INFO

  readinessProbe:
    # tells the kubelet that it should wait second before performing
the first probe
    initialDelaySeconds: 30
    # Number of seconds after which the probe times out
    timeoutSeconds: 3
    # specifies that the kubelet should perform a liveness probe every
xx seconds
    periodSeconds: 10
```

```
    # Minimum consecutive successes for the probe to be considered
successful after having failed
    successThreshold: 1
    # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up
    failureThreshold: 3

  livenessProbe:
    # tells the kubelet that it should wait second before performing
the first probe
    initialDelaySeconds: 30
    # Number of seconds after which the probe times out
    timeoutSeconds: 3
    # specifies that the kubelet should perform a liveness probe every
xx seconds
    periodSeconds: 15
    # Minimum consecutive successes for the probe to be considered
successful after having failed
    successThreshold: 1
    # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up
    failureThreshold: 3

  # Resource details
  resources:
    limits:
      cpu: 2
      initServiceCpu: 1
      updateServiceCpu: 1
      memory: 2Gi
      updateServiceMemory: 1Gi
      initServiceMemory: 1Gi
    requests:
      cpu: 1
      initServiceCpu: 0.5
      updateServiceCpu: 0.5
      memory: 1Gi
      updateServiceMemory: 0.5Gi
      initServiceMemory: 0.5Gi
    target:
      averageCpuUtil: 80

  # Nuber of Pods must always be available, even during a disruption.
  minAvailable: 1
  # Min replicas to scale to maintain an average CPU utilization
  minReplicas: 1
  # Max replicas to scale to maintain an average CPU utilization
  maxReplicas: 5

  allowedCipherSuites:
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
```

```
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

  cipherSuites:
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

  # To Initialize SSL related infrastructure in init/update container
  initssl: false
  #Server Configuration for http and https support
  enableIncomingHttp: true
  enableIncomingHttps: false

  ingressGwCertReloadEnabled: false
  ingressGwCertReloadPath: /ingress-gw/certificate/reload

  # Routes Configurations
  routesConfig:
  # Note: Update FQDN and PORT with actual values. If not remove those
routes else CNCC will fail to deploy.
  # CNCC requires complete routes and not placeholders.
  # Default mapping should be the last route entry.
  # Examples for routes
  #- id: scp_configuration
  #  uri: http://10.75.153.121:31131
  #  path: /soothsayer/v1/**
  #- id: default_configuration
  #  uri: http://cncc-core-cmservice.cncc.svc.cluster.local:8442
  #  path: /**
  - id: scpc_configuration
    uri: http://<FQDN>:<PORT>
    path: /soothsayer/v1/**
  - id: nrf_configuration
    uri: http://<FQDN>:<PORT>
    path: /nrf-configuration/v1/**
  - id: udr1
    uri: http://<FQDN>:<PORT>
    path: /nudr-dr-prov/**,/nudr-dr-mgm/**,/nudr-group-id-map-prov/**,/
slf-group-prov/**
  - id: udr2
    uri: http://<FQDN>:<PORT>
    path: /nudr-config/**
  - id: policy_configuration
    uri: http://<FQDN>:<PORT>
    path: /policyapi/**
    filters:
      rewritePath: "/policyapi(?<segment>/?.*), $\\{segment}"
  - id: policy_export_configuration # Route for Export configuration
under Policy screen
    uri: http://<FQDN>:<PORT>
    path: /oc-cnpolicy-configuration/**
  - id: default_configuration # Default configuration should be the
```

```
last routesConfig entry
    uri: http://<helmrelease>-cmservice.<namespace>.<domain>:8442
    path: /**

  nodeSelector:
    nodeKey: ""
    nodeValue: ""

  # CNCC configuration
  cncc:
    # Enable cncc feature including iam
    enabled: true
    # Enable security logs
    securityLogEnabled: true
    # Core Configuration
    core:
      # Session Timeout Value in Seconds. Default: 1800, Minimum: 300,
Maximum: 7200
      sessionTimeoutSeconds: 1800
    # IAM Configuration
    # uri should include the CNCC IAM ingress-gateway externalIp or
FQDN and service port.
    iam:
      uri: http://<IP or FQDN>:<PORT>
############################################################
#            Section End  : ingress gateway attributes  #
############################################################
```

> **Note:**
>
> - The field *ingress-gateway.cncc.iam.uri*should include the CNCC IAM Console URL. Check CNCC IAM Services for the URL.
> - For POLICY deployment set cmEnableImportExport : true , this enables **Import** and **Export** buttons. It is applicable only for POLICY deployment.
> - *envSystemName* can be used to display site name. Example: envSystemName: CNCC - Site Name.

CNCC Core Configuration Parameters

Following tables provide list of configuration parameters in the Helm file:

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
global.serviceAccountName	<String>	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	O	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>`
global.dockerRegistry	<String>	It may contain lowercase letters, digits, and separators. A separator is defined as a period, one or two underscores, or one or more dashes.	M	Here user provides the registry that contains cncc core images. It comprises of the following: <registry-url>:<registry-port> e.g : ocspf-registry.us.oracle.com:5000
global.test.nfName	<String>	NF Name	M	Name of deployment for which helm test is done
global.test.image.name	<String>	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	M	Image name for the helm test container image

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
global.test.image.tag	<String>	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	M	Image name for the helm test container image
global.test.config.logLevel	<String>	It can take values like: WARN, DEBUG, INFO, etc.	M	Log level for helm test pod
global.test.config.timeout	<String>	Range: 1-300 Unit:seconds	M	Timeout value for the helm test operation. If exceeded helm test will be considered as failure
global.publicHttpSignalingPort	<Integer>	It can take value in the range: 0-65535	O	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
global.publicHttpsSignallingPort	<Integer>	It can take value in the range: 0-65535.	O	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)
global.type	<String>	It can take value LoadBalance/NodePort depending upon one wants to expose the service from outside the Kubernetes cluster or not.	O	It is used to decide where user wants to expose the service from outside the Kubernetes cluster or not.
global.metalLbIpAllocationEnabled	<Boolean>	True/False By default, it is true.	O	This field enables or disables IP Address allocation from Metallb Pool

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
global.metalLbIpAllocationAnnotation	<Stirng>	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 set to: metallb.universe.tf/ address-pool: signaling"		It is the address Pool Annotation for Metallb
global.staticIpAddressEnabled	<Boolean>	True/False By default, it is false.	O	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
global.staticIpAddress	<Stirng>	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	O	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
global.staticNodePortEnabled	<Boolean>	True/False By default, it is true.	O	If Static node port needs to be set, then set staticNodePortEnabled flag to true and provide value for staticNodePort else random node port will be assigned by K8s

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
global.staticHttpNodePort	<Integer>	It can take value in the range: 0-65535. Default value:30075	O	If Static node port needs to be set, then set staticNodePortEnabled flag to true and provide value for staticNodePort else random node port will be assigned by K8s
global.staticHttpsNodePort	<Integer>	It can take value in the range: 0-65535. Default value:30075	O	If Static node port needs to be set, then set staticNodePortEnabled flag to true and provide value for staticNodePort else random node port will be assigned by K8s
global.nodeSelector.nodeKey	<String>		O	global node selector key
global.nodeSelector.nodeValue	<String>		O	global node value key
global.customExtension.allResources.labels	<String>	Custom Labels that needs to be added to all the Ingress-Gateway k8s resources	O	This can be used to add custom label(s) to all k8s resources that will be created by Ingress-Gateway helm chart.
global.customExtension.allResources.annotations	<String>	Custom Annotations that needs to be added to all the Ingress-Gateway k8s resources	O	This can be used to add custom annotation(s) to all k8s resources that will be created by Ingress-Gateway helm chart.
global.customExtension.lbServices.labels	<String>	Custom Labels that needs to be added to Ingress-Gateway Services that are considered as Load Balancer type	O	This can be used to add custom label(s) to all Load Balancer Type Services that will be created by Ingress-Gateway helm chart.
global.customExtension.lbServices.annotations	<String>	Custom Annotations that needs to be added to Ingress-Gateway Services that are considered as Load Balancer type	O	This can be used to add custom annotation(s) to all Load Balancer Type Services that will be created by Ingress-Gateway helm chart.

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
global.customExtension.lbDeployments.labels	<String>	Custom Labels that needs to be added to Ingress-Gateway Deployments that are associated to a Service which is of Load Balancer type	O	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.
global.customExtension.lbDeployments.annotations	<String>	Custom Annotations that needs to be added to Ingress-Gateway Deployments that are associated to a Service which is of Load Balancer type	O	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.
global.customExtension.nonlbServices.labels	<String>	Custom Labels that needs to be added to Ingress-GatewayServices that are considered as not Load Balancer type	O	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.
global.customExtension.nonlbServices.annotations	<String>	Custom Annotations that needs to be added to Ingress-Gateway Services that are considered as not Load Balancer type	O	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.
global.customExtension.nonlbDeployments.labels	<String>	Custom Labels that needs to be added to Ingress-Gateway Deployments that are associated to a Service which is not of Load Balancer type	O	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.

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
global.customExtension.nonlbDeployments.annotations	<String>	Custom Annotations that needs to be added to Ingress-Gateway Deployments that are associated to a Service which is not of Load Balancer type	O	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.
global.k8sResource.container.prefix	<String>	Value that will be prefixed to all the container names of Ingress-Gateway.		This value will be used to prefix to all the container names of Ingress-Gateway
global.k8sResource.container.suffix	<String>	Value that will be suffixed to all the container names of Ingress-Gateway.		This value will be used to suffix to all the container names of Ingress-Gateway.
cmservice.envLoggingLevelApp	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc.	O	It is the level at which user wants to see the logs. E.g. WARN
cmservice.image.name	<String>	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	M	Image Name to be used for "cncc-cmservice" micro service
cmservice.image.tag	<String>	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	M	Image Tag to be used for "cncc-cmservice" micro service
cmservice.image.pullPolicy	<String>	It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy	M	Pull Policy decides from where to pull the image.

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
cmservice.resources.limits.cpu	<Float>	Valid floating point value between 0 and 1	O	It limits the number of CPUs to be used by the "cncc-cmservice" microservice. By default, it is set to '2'.
cmservice.resources.limits.memory	<String>	Valid Integer value followed by Mi/Gi etc.	O	It limits the memory utilization by the "cncc-cmservice" microservice. By default, it is set to '2'.
cmservice.resources.requests.cpu	<Float>	Valid floating point value between 0 and 1	O	It provides a given number of CPUs for the "cncc-cmservice" microservice. By default, it is set to '2'.
cmservice.resources.requests.memory	<String>	Valid Integer value followed by Mi/Gi etc.	O	It provides a given amount of memory for the "cncc-cmservice" microservice. By default, it is set to '1.
cmservice.deployment.envManageNF	<String>	It is the List of NFsE.g. SCP, POLICY	M	It is the list of the enabled NFs and the same NFs will be displayed in the GUI
cmservice.deployment.envSystemName	<String>	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	M	This is the name of product which appears as brand name and can be used to mention site name as well. E.g. envSystemName: CNCC
cmservice.deployment.envNFVersion	<String>	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.	M	This is the version of product which appears with brand name. E.g. envNFVersion: 1.3.0
cmservice.deployment.cmWindowName	<String>	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	M	This is the name of the window that appears on the browser tab. E.g. cmWindowName: CNCC

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
cmservice.deployment.nodeSelectorEnabled	<boolean>	It can take either True or False value. By default, it is false.	O	NodeSelector is the simplest recommended form of node selection constraint. NodeSelector is a field of PodSpec. It specifies a map of key-value pairs. For the pod to be eligible to run on a node, the node must have each of the indicated key-value pairs as labels
cmservice.deployment.nodeSelectorKey	<String>	By default, its value is zone.	O	Node Selector Key
cmservice.deployment.nodeSelectorValue	<String>	By default, its value is app.	O	Node Selector value
cmservice.deployment.livenessProbe.initialDelaySeconds	<Integer>	It can take value in the range: 0-65535. Default value:60	O	It tells the kubelet that it should wait second before performing the first probe
cmservice.deployment.livenessProbe.periodSeconds	<Integer>	It can take value in the range: 0-65535. Default value:3	O	It specifies that the kubelet should perform a liveness probe every xx seconds
cmservice.deployment.livenessProbe.timeoutSeconds	<Integer>	It can take value in the range: 0-65535. Default value:15	O	It is the number of seconds after which the probe times out
cmservice.deployment.livenessProbe.successThreshold	<Integer>	It can take value in the range: 0-65535. Default value:1	O	Minimum consecutive successes for the probe to be considered successful after having failed
cmservice.deployment.livenessProbe.failureThreshold	<Integer>	It can take value in the range: 0-65535. Default value:3	O	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up
cmservice.deployment.readinessProbe.initialDelaySeconds	<Integer>	It can take value in the range: 0-65535. Default value:20	O	It tells the kubelet that it should wait second before performing the first probe

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
cmservice.deployment.readinessProbe.timeoutSeconds	<Integer>	It can take value in the range: 0-65535. Default value:3	O	It is the number of seconds after which the probe times out
cmservice.deployment.readinessProbe.periodSeconds	<Integer>	It can take value in the range: 0-65535. Default value:10	O	It specifies that the kubelet should perform a liveness probe every xx seconds
cmservice.deployment.readinessProbe.successThreshold	<Integer>	It can take value in the range: 0-65535. Default value:1	O	Minimum consecutive successes for the probe to be considered successful after having failed
cmservice.deployment.readinessProbe.failureThreshold	<Integer>	It can take value in the range: 0-65535. Default value:3	O	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up
cmservice.deployment.dependenciesLogging[].name	<String>	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	O	Name of the package that for which log level is to be set. Eg: logging.level.org.springframework
cmservice.deployment.dependenciesLogging[].value	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc.	O	It is the level at which user wants to see the logs. E.g. WARN
cmservice.service.customExtension.labels	<String>	Custom Labels that needs to be added to all the cmservice k8s resources	O	This can be used to add custom label(s) to all k8s resources that will be created by cmservice helm chart.
cmservice.service.customExtension.annotations	<String>	Custom Annotations that needs to be added to all the cmservice k8s resources	O	This can be used to add custom annotation(s) to all k8s resources that will be created by cmservice helm chart.
cmservice.service.type	<String>	It can take only 'ClusterIP' as the value.	O	It is used to decide where user wants to expose the service from outside the Kubernetes cluster or not.

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
cmservice.servicePorts.cmServiceHttp	<Integer>	It can take value in the range: 0-65535	O	It is the port number which makes cmservice visible to other services running within the same K8s cluster
cmservice.containerPorts.monitoringHttp	<Integer>	It can take value in the range: 0-65535	O	It is the monitoring container port for cm service
cmservice.containerPorts.cmServiceHttp	<Integer>	It can take value in the range: 0-65535	O	It is the container port for cm service
ingress-gateway.image.name	<String>	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	M	It is the image name of the ingress-gateway as provided by the user
ingress-gateway.image.tag	<String>	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	M	Image Tag to be used for ingress-gateway.
ingress-gateway.image.pullPolicy	<String>	It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy	O	Pull Policy decides from where to pull the image.
ingress-gateway.initContainersImage.name	<String>	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	M	Image Name to be used for "cncc-cmservice" micro service

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.initContainersImage.tag	<String>	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	M	Image Tag to be used for "cncc-cmservice" micro service
ingress-gateway.initContainersImage.pullPolicy	<String>	It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy	O	Pull Policy decides from where to pull the image.
ingress-gateway.updateContainersImage.name	<String>	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	M	Image Name to be used for "cncc-cmservice" micro service
ingress-gateway.updateContainersImage.tag	<String>	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	M	Image Tag to be used for "cncc-cmservice" micro service
ingress-gateway.updateContainersImage.pullPolicy	<String>	It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy	O	Pull Policy decides from where to pull the image.
ingress-gateway.service.ssl.tlsVersion	<String>	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. It is set to TLSv1.2	O	It is the TLS version

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.service.ssl.privateKey.k8SecretName	<String>	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	O	Name of the privatekey secret Ex: cncc-core-ingress-secret
ingress-gateway.service.ssl.privateKey.k8NameSpace	<String>	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	O	Namespace of privatekey Ex: cncc
ingress-gateway.service.ssl.privateKey.rsa.fileName	<String>	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	O	rsa private key file name Ex: rsa_private_key_pkcs1.pem
ingress-gateway.service.ssl.privateKey.ecdsa.fileName	<String>	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	O	ecdsa private key file name Ex: ssl_ecdsa_private_key.pem
ingress-gateway.service.ssl.certificate.k8SecretName	<String>	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	O	Name of the certificate secret Ex: cncc-core-ingress-secret
ingress-gateway.service.ssl.certificate.k8NameSpace	<String>	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	O	Namespace of certificate Ex: cncc
ingress-gateway.service.ssl.certificate.rsa.fileName	<String>	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	O	rsa certificate file name Ex: ssl_rsa_certificate.crt

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.service.ssl.certificate.ecdsa.fileName	<String>	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	O	ecdsa certificate file name Ex: ssl_ecdsa_certificate.crt
ingress-gateway.service.ssl.caBundle.k8SecretName	<String>	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	O	Name of the caBundle secret Ex: cncc-core-ingress-secret
ingress-gateway.service.ssl.caBundle.k8NameSpace	<String>	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	O	Namespace of caBundle Ex: cncc
ingress-gateway.service.ssl.caBundle.fileName	<String>	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	O	rsa caBundle file name Ex: caroot.cer
ingress-gateway.service.ssl.keyStorePassword.k8SecretName	<String>	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	O	Name of the keyStorePassword secret Ex: cncc-core-ingress-secret
ingress-gateway.service.ssl.keyStorePassword.k8NameSpace	<String>	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	O	Namespace of keyStorePassword Ex: cncc
ingress-gateway.service.ssl.keyStorePassword.fileName	<String>	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	O	File name that has password for keyStore Ex: ssl_keystore.txt

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.service.ssl.trustStorePassword.k8SecretName	<String>	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	O	Name of the trustStorePassword secret Ex: cncc-core-ingress-secret
ingress-gateway.service.ssl.trustStorePassword.k8NameSpace	<String>	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	O	Namespace of trustStorePassword Ex: cncc
ingress-gateway.service.ssl.trustStorePassword.fileName	<String>	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	O	File name that has password for trustStore Ex: ssl_truststore.txt
ingress-gateway.service.ssl.initialAlgorithm	<String>	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	O	Default values is RSA256
ingress-gateway.service.customExtension.labels	<String>	Custom Labels that needs to be added to ingress-gateway specific Service.	O	This can be used to add custom label(s) to ingress-gateway Service.
ingress-gateway.service.customExtension.annotations	<String>	Custom Annotations that needs to be added to ingress-gateway specific Services.	O	This can be used to add custom annotation(s) to ingress-gateway Service.
ingress-gateway.deployment.customExtension.labels	<String>	Custom Labels that needs to be added to ingress-gateway specific Deployment.	O	This can be used to add custom label(s) to ingress-gateway Deployment.
ingress-gateway.deployment.customExtension.annotations	<String>	Custom Annotations that needs to be added to ingress-gateway specific Deployment.	O	This can be used to add custom annotation(s) to ingress-gateway Deployment.

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.readinessProbe.initialDelaySeconds	<Integer>	It can take value in the range: 0-65535. Default value:30	O	It tells the kubelet that it should wait second before performing the first probe
ingress-gateway.readinessProbe.timeoutSeconds	<Integer>	It can take value in the range: 0-65535. Default value:3	O	It is the number of seconds after which the probe times out
ingress-gateway.readinessProbe.periodSeconds	<Integer>	It can take value in the range: 0-65535. Default value:10	O	It specifies that the kubelet should perform a liveness probe every xx seconds
ingress-gateway.readinessProbe.successThreshold	<Integer>	It can take value in the range: 0-65535. Default value:1	O	Minimum consecutive successes for the probe to be considered successful after having failed
ingress-gateway.readinessProbe.failureThreshold	<Integer>	It can take value in the range: 0-65535. Default value:3	O	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up
ingress-gateway.livenessProbe.initialDelaySeconds	<Integer>	It can take value in the range: 0-65535. Default value:30	O	It tells the kubelet that it should wait second before performing the first probe
ingress-gateway.livenessProbe.timeoutSeconds	<Integer>	It can take value in the range: 0-65535. Default value:3	O	It is the number of seconds after which the probe times out
ingress-gateway.livenessProbe.periodSeconds	<Integer>	It can take value in the range: 0-65535. Default value:15	O	It specifies that the kubelet should perform a liveness probe every xx seconds
ingress-gateway.livenessProbe.successThreshold	<Integer>	It can take value in the range: 0-65535. Default value:1	O	Minimum consecutive successes for the probe to be considered successful after having failed
ingress-gateway.livenessProbe.failureThreshold	<Integer>	It can take value in the range: 0-65535. Default value:3	O	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.minAvailable	<Integer>	It can take value in the range: 0-65535. Default value:1	O	It is the number of pods that must always be available, even during a disruption.
ingress-gateway.minReplicas	<Integer>	It can take value in the range: 0-65535. Default value:1	O	Min replicas to scale to maintain an average CPU utilization
ingress-gateway.maxReplicas	<Integer>	It can take value in the range: 0-65535. Default value:5	O	Max replicas to scale to maintain an average CPU utilization
ingress-gateway.initssl	<Boolean>	It can take either True or False value. By default, it is false.	O	To Initialize SSL related infrastructure in init/update container
ingress-gateway.enableIncomingHttp	<Boolean>	It can take either True or False value. By default, it is false.	O	Server Configuration for http and https support
ingress-gateway.enableIncomingHttps	<Boolean>	It can take either True or False value. By default, it is false.	O	Server Configuration for http and https support
ingress-gateway.cipherSuites	<List[String]>	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	M, if ingress gateway.enableIncomingHttps is true	Allowed CipherSuites for TLS1.2
ingress-gateway.cncc.enabled	<Boolean>	It can take either True or False value. By default, it is true.	M	It enables CNCC features i.e authentication and authorization on ingress

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.cncc.securitylogEnabled	<boolean>	It can take either True or False value. By default, it is true	O	This flag is to enable/ disable security logs for cncc.
ingress-gateway.cncc.core.sessionTimeoutSeconds	<Integer>	It can take value in the range: 0-65535.Default Value: 1800	M	It takes the timeout value for CNCC Session in seconds. Default: 1800 Minimum: 300 Maximum: 7200
ingress-gateway.cncc.iam.uri	<String>	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	M	It is the URI of the cncc-iam ingress.
ingress-gateway.ports.containerPort	<Integer>	It can take value in the range: 0-65535. Default value: 8081	O	It is the http port of the container for the ingress-gateway.
ingress-gateway.ports.containersslPort	<Integer>	It can take value in the range: 0-65535. Default value: 8443	O	It is the https port of the container for the ingress-gateway.
ingress-gateway.ports.actuatorPort	<Integer>	It can take value in the range: 0-65535. Default value: 9090	O	It is the actuator port of the container for the ingress-gateway.
ingress-gateway.log.level.root	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc.	O	It is the level at which user wants to see the logs. E.g. WARN
ingress-gateway.log.level.ingress	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc.	O	Log level for ingress logs
ingress-gateway.log.level.cncc.security	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc.	O	Log level for cncc security logs
ingress-gateway.resources.limits.cpu	<Float>	Valid floating point value between 0 and 1	O	It limits the number of CPUs to be used by the microservice.
ingress-gateway.resources.limits.memory	<String>	Valid Integer value followed by Mi/Gi etc.	O	It limits the memory utilization by the microservice.
ingress-gateway.resources.requests.cpu	<Float>	Valid floating point value between 0 and 1	O	It provides a given number of CPUs for the microservice.

Attribute Name	DataType	Range	Mandatory(M) / Optional(O)/ Conditional(C)	Description
ingress-gateway.resources.requests.memory	<String>	Valid Integer value followed by Mi/Gi etc.	O	It provides a given amount of memory for the microservice.
ingress-gateway.resources.target.averageCpuUtil	<Integer>	A value in between 0-100	O	It gives the average CPU utilization percentage.
ingress-gateway.routesConfig[].id	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	If SCP route needs to be added to CNC Console Core ingress-gateway
ingress-gateway.routesConfig[].uri	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	
ingress-gateway.routesConfig[].path	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	
ingress-gateway.routesConfig[].order	<Integer>	Valid Integer value	O	
ingress-gateway.routesConfig[].filters.rewritePath	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	O	
ingress-gateway.ingressGwCertReloadEnabled	<boolean>	It can take either True or False value. By default, it is false.	M	
ingress-gateway.ingressGwCertReloadPath	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	
ingress-gateway.nodeSelector.nodeKey	<String>		O	node selector key specific to chart (note this will be looked first and then if not present global node key will be picked)

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.nodeSelector.nodeValue	<String>		O	node selector value specific to chart (note this will be looked first and then if not present global node value will be picked)

CNCC Core Access

CNCC Core can be accessed in the following ways:

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

# Ways to access CNCC Core GUI
# 1] Node-IP and NodePort
http://10.75.xx.xx:30075/*
# 2] DNS Resolvable FQDN and NodePort
http://cncc-core-ingress-gateway.cncc.svc.cluster.local:30075/*
# 3] External LB-IP and ServicePort
http://10.75.xx.xx:8080/*
# 4] DNS Resolvable FQDN and ServicePort
http://cncc-core-ingress-gateway.cncc.svc.cluster.local:8080/*
```

> **Note:**
>
> Login to CNC IAM and add redirect url pointing CNCC Core. CNCC cannot be accessed before CNCC IAM is configured to redirect. Refer CNCConsole Post Installation Steps for CNCC-IAM

CNCC Core Uninstall

CNCC Core can be uninstalled as follows. The following step needs to be executed from a server that has access to Kubectl and helm commands:

Execute the following command to uninstall CNCC Core:

```
For Helm 2:
$ helm delete <deployment name> --purge
Example:
$ helm delete cncc-core --purge

For Helm 3:
```

```
$ helm uninstall <deployment name> --namespace <deployment namespace>
Example:
$ helm uninstall cncc-core --namespace cncc
```

5
CNC Console Supported NFs and Version Compatibility

The CNC Console supports the following NF versions:

SR. No	NF	NF Version	Enabling NFs	Route Configuration	Remarks

1	NRF	1.8.0	Update in cmservice section of values.yaml
			`envManageNF:NRF`
				Update in routeConfig under ingress section in values.yaml
				`- id: nrf_configuration`
				`uri: http://<FQDN>:<PORT>`
				`path: /nrf-configuration/v1/**`	

2	POLICY	1.8.0	Update in cmservice section of values.yaml
			`envManageNF: POLICY`
			`cmEnableImportExport: true`
				Update in routeConfig under ingress section in values.yaml
				`- id: policy_configuration`
				`uri: http://<FQDN>:<PORT>`
				`path: /policyapi/**`
				`filters:`
				`rewritePath: "/policyapi(?<segment>/?.*), $\\{segment}"`
				`- id: policy_export_configuration # Route for Export configuration under Policy screen`
				`uri: http://<FQDN>:<PORT>`
				`path: /oc-cnpolicy-configuration/**`
					1. For enabling POLICY, add POLICY in "envManageNF" field.
					2. To enable Import, Export buttons set cmEnableImportExport as true. It is applicable only for POLICY GUI.
					`cmEnableImportExport: true`

SR. No	NF	NF Version	Enabling NFs	Route Configuration	Remarks

3	SCP	1.8.0	Update in cmservice section of values.yaml
			`envManageNF:SCP`
				Update in routeConfig under ingress section in values.yaml
				
				`- id: scpc_configuration`
				`uri: http://<FQDN>:<PORT>`
				`path: /soothsayer/v1/**`	

4	UDR	1.8.0	Update in cmservice section of values.yaml
			`envManageNF:UDR`
				Update in routeConfig under ingress section in values.yaml
				
				`- id: udr1`
				`uri: http://<FQDN>:<PORT>`
				`path: /nudr-dr-prov/**,/nudr-dr-mgm/**,/nudr-group-id-map-prov/**,/slf-group-prov/**`
				`- id: udr2`
				`uri: http://<FQDN>:<PORT>`
				`path: /nudr-config/**`
				
				` `	

6
CNC Console Helm Test

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

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

The following instructions needs to be followed to execute the Helm test functionality. The configurations mentioned in the step below must be done before executing the helm install command.

- Configure the helm test configurations which are under global sections in values. Refer the configurations below:
  CNCC Core Helm Test

```
global:
  # Helm test related configurations
  test:
    nfName: cncc-core
    image:
      name: cncc/nf_test
      tag: 1.3.0
    config:
      logLevel: WARN
      timeout: 40
```

  CNCC IAM Helm Test

```
# Helm test related configurations
  test:
    nfName: cncc-iam
    image:
      name: cncc/nf_test
      tag: 1.3.0
    config:
      logLevel: WARN
      timeout: 40
```

- Execute the below command to run Helm test on the installation done:

```
helm3 test <helm_release_name> -n <k8s namespace>
```

```
Example:
helm3 test cncc-core -n cncc
NAME: cncc-core
LAST DEPLOYED: Tue Aug  4 07:52:08 2020
NAMESPACE: cncc
STATUS: deployed
REVISION: 1
TEST SUITE:     cncc-core-test
Last Started:   Tue Aug 11 07:16:09 2020
Last Completed: Tue Aug 11 07:16:20 2020
Phase:          Succeeded
NOTES:
# Copyright 2020 (C), Oracle and/or its affiliates. All rights
reserved.

Thank you for installing cncc-core.

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

  $ helm status cncc-core
  $ helm get cncc-core
```

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

7

Post Installation Steps for CNC Console IAM

Prerequisites

The CNC Console IAM and CNCC Core must be deployed.

Setting up the cncc redirection URL, Create user and Assign the roles

Once CNCC IAM is deployed admin must do the following:

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

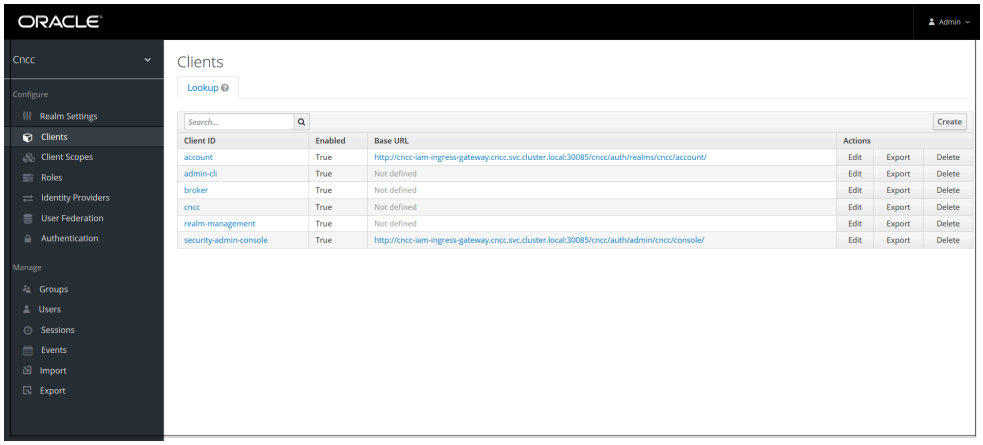
Steps for the setting up the cncc redirection URL, Create user and Assign the roles:

1. Login to CNCC IAM Console using admin credentials provided during installation of CNCC IAM.

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

# Ways to access CNCC IAM GUI
# 1] Node-IP and NodePort
http://10.75.xx.xx:30085/*
# 2] DNS Resolvable FQDN and NodePort
http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:30085/*
# 3] External LB-IP and ServicePort
http://10.75.xx.xx:8080/*
# 4] DNS Resolvable FQDN and ServicePort
http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:8080/*
```

2. Go to **Clients** and select **Cncc**.

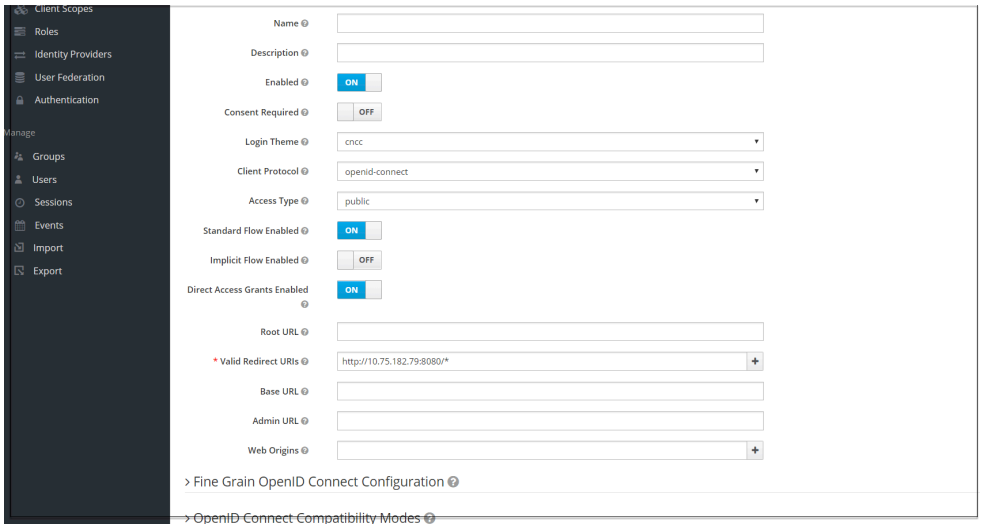


3. Enter CNCC Core Ingress URI in the **Valid Redirect URIs** field and **Save.**

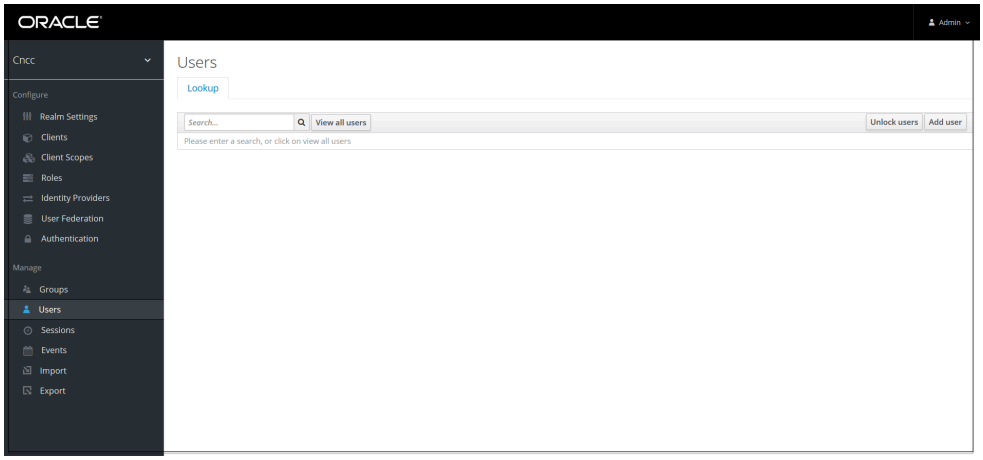
```
<scheme>://<cncc-core-ingress IP/FQDN>:<cncc-core-ingress Port>

# Possible ways to provide CNCC Core GUI URI
# 1] Node-IP and NodePort
http://10.75.xx.xx:30075/*
# 2] DNS Resolvable FQDN and NodePort
http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:30075/*
# 3] External LB-IP and ServicePort
http://10.75.xx.xx:8080/*
# 4] DNS Resolvable FQDN and ServicePort
http://cncc-iam-ingress-gateway.cncc.svc.cluster.local:8080/*
```

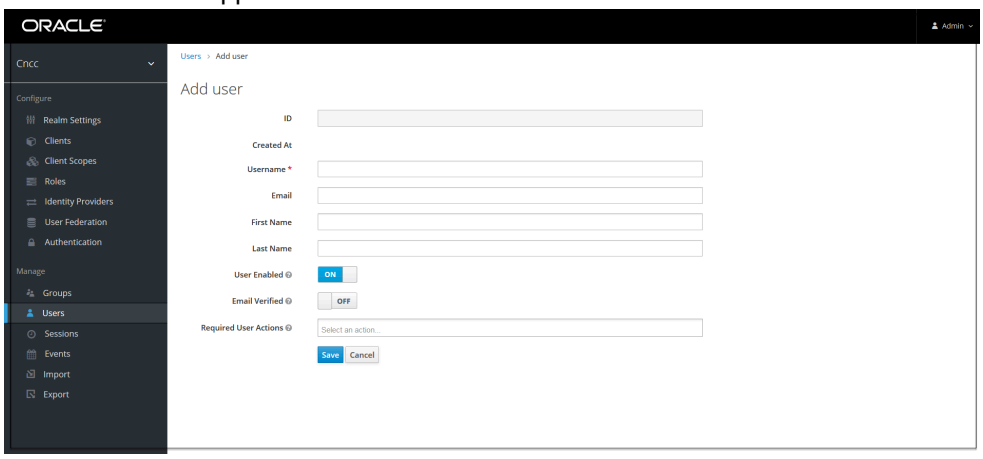
> **Note:**
>
> RedirectUri and Uri to access CNCC Core GUI must be same.
>
> The **valid_redirect_uri** provided in CNCC IAM and uri by which you access the CNCC Core GUI should be same. If there is mismatch then it would result in **“invalid redirect_uri”** on CNCC Core GUI. Example: In CNCC IAM the **valid_redirect_uri** of CNCC Core is mentioned as Service FQDN, i.e *http://cncc-core-ingress-gateway.cncc.svc.cluster.local:30075/** and if you are accessing the CNCC Core GUI with IP and NodePort, i.e *http://10.75.xx.xx:30075/** or vice-versa you will get **“invalid redirect_uri”** on CNCC Core GUI.



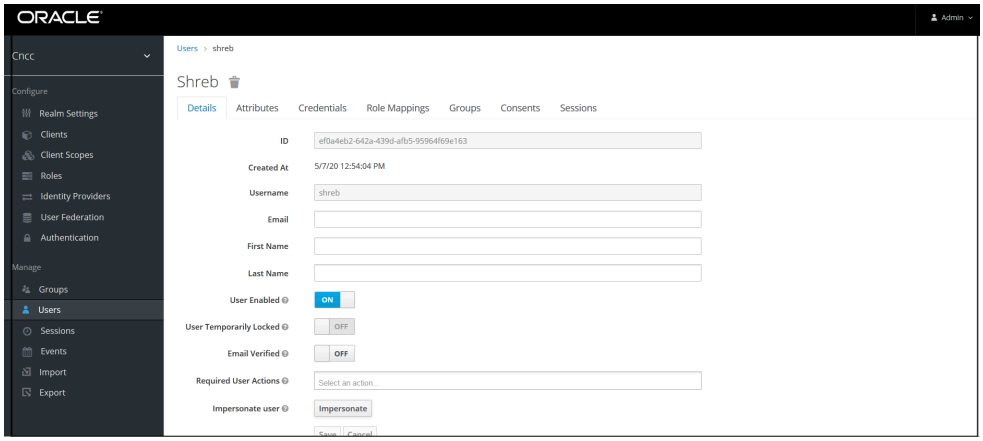
4. Select **Manage** and click **Users** and select **Add user** in the right pane.



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

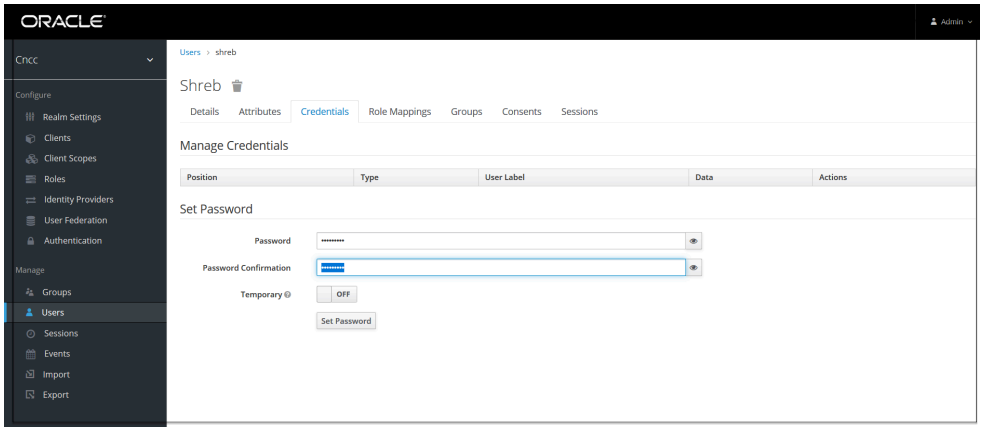


6. The user has been created and the user Details screen appears.

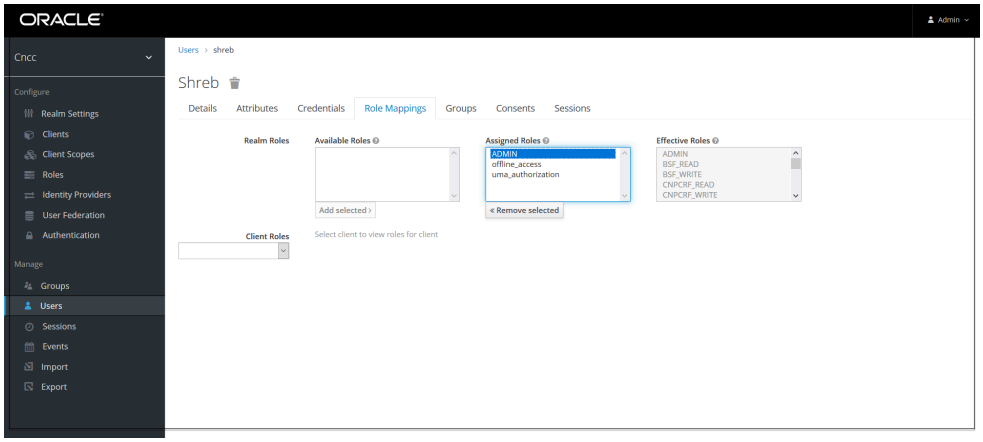


7. For setting the password for the user, select **Credentials** tab and set the password for that user.

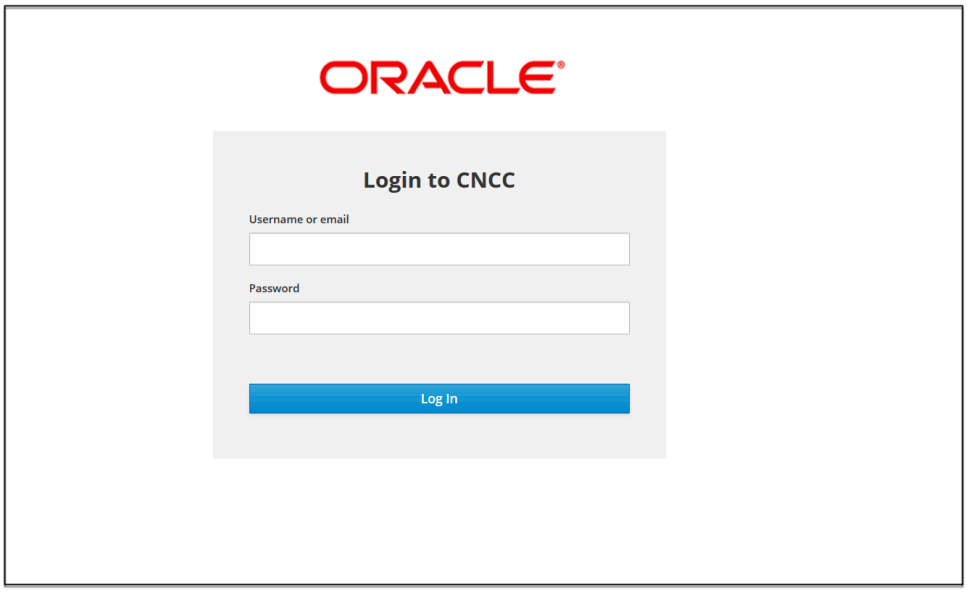
> **Note:**
>
> Setting **Temporary flag** as **ON** prompts the user to change the password while login for the first time to CNCC Core Interface.



8. Navigate to the **Role Mappings** tab and assign the user role.



9. Login to CNCC Core using the credentials of the user created earlier.



8
Troubleshooting CNC Console

This section provides information to troubleshoot the common error which can be encountered during the installation and upgrade of CNC Console.

Not able to display the release version of the NF at CNCC banner

Problem: CNCC banner displays the release version of CNCC. But not displaying the release version of the NF.

Solution:

- Both “About” section and Application name displayed next to Oracle logo use the envSystemName and envNFVersion helm fields.
- The value set of *envSystemName* and *envNFVersion* combines to display the Application name (Application name = envSystemName + envNFVersion).
- CNCC Core Custom values has *envSystemName* and *envNFVersion* mentioned in it, but these values can be overridden.

Unable to reach CNCC Core IP/port directly

Problem: Unable to reach CNCC Core IP/port directly. *redirect_uri* is inserted instead of directly accessing the CNCC Core.

Solution:

- As per design, CNCC redirects requests to CNCC IAM for authentication. On successful authentication, CNCC IAM redirects the user back to CNCC GUI.

'Admin' user created under Cncc realm is unable to access CNCC IAM

Problem: Once a user created under Cncc realm is assigned an ‘Admin’ privilege, whether the user have the access to CNCC IAM.

Solution:

- The users created under **Cncc** realm have access only to CNCC Core; not to CNCC IAM.
- If a new **admin** user needs to be created who can login to CNCC IAM, that user has to be created under **Master** realm.

CNCC returns 403 error during NF Configuration

Problem: CNCConsole returns a 403 Error Code and error *"Forbidden. Data could not be saved".*

Error Code/Error Message:
403/Forbidden

Solution:

- For doing the write operation on any NF via CNCConsole, user must have <NF>_WRITE role (permission).
- User must login to CNCC IAM to check and assign the roles.
- Check the roles of the user through which you have logged in, user must have both <NF>_READ and <NF>_WRITE roles assigned.

CNCConsole returns 500 - Internal Server Error

Problem: CNCConsole returns a 500 Error Code while accessing NF Resource.

Error Code/Error Message:
500/Internal Server Error

Solution:

- This issue occurs when the NF routes are not configured correctly.
- Ensure that correct routes for each NF are configured during deployment.
- You can to provide either of the following:
  - *http://<Service FQDN>:<Service Port>*
  - *http://<External LB IP/Node IP>:<LB Port/NodePort>*

CNCC IAM is accessible but CNC Core is not accessible

Problem: CNCC IAM is accessible but CNCC Core is not accessible.

Error Message:
The ID Token contains invalid claims. (This is a JWT validation error, usually this indicates that the system clock on your server Is off.)

Solution:

- In this case IAM (node1) was ahead of time and Ingress Gateway (node2) was 5 minutes behind.
- When Ingress Gateway receives the token it was of future time, so it invalidates it by throwing "The ID Token contains invalid claims: {iat=2020-05-26T08:32:12Z}".
- This issue occurs when Ingress Gateway is behind time and CNCC IAM is ahead of time.
- It is important to have the same time in CNCC IAM and Ingress Gateway.

- The key is to have the same time in both IAM and Ingress Gateway, in case they are running in different instances having a NTP server this issue will not happen, and in case both of them are deployed in the same instance neither.

The dependency issue of CNCC Core Routes id field name and service name

Problem:
The - *id: scpc-configuration* under route is mentioned as - *id: scpc_configuration* (with an underscore instead of a hyphen) and its not matching service name.

Solution:

- id field in routes is used to uniquely identify the route, it can be any string. There is no dependency with service name.
- - id: scpc-configuration or - id: scpc_configuration should not make any difference , it can be named as per convenience.

CNCC IAM admin password configured via kubectl secret is not reflected (Example: if configured cncc-iam-secret)

Problem:
CNCC IAM admin password change through cncc-iam-secret not working.

Solution:

- During first time installation with fresh database, CNCC IAM reads password from cncc-iam-secret and stores it in database. So any updates to the admin password must be done via CNCC IAM GUI.

"Invalid redirect uri" error while accessing CNCC Core GUI

Problem:

CNCC Core GUI is also not accessible and the browser is throwing an error "*Invalid redirect uri*".

Observation:

This error is thrown when, there is mismatch between **valid_redirect_uri** provided in CNCC IAM Admin Console and **uri** by which you access the CNCC Core GUI.

Example: In CNCC IAM the **redirect_uri** of CNCC Core is mentioned as *Service FQDN* i.e., http://cncc-core-ingress-gateway.cncc.svc.cluster.local:30075/*. If you are accessing the CNCC Core GUI with *IP and NodePort* i.e., http://10.75.xx.xx:30085/* or vice-versa you get **"invalid redirect_uri"** on CNCC Core GUI.

Solution:

The **valid_redirect_uri** provided in CNCC IAM and **uri** by which you access the CNCC Core GUI should be same.

CNCC Core Debugging through Logs

For information about logs, refer the *CNCC Logs* section in the *CNC Console User's Guide*.

FAQ

Does CNCC support Command Line Interface (CLI) ?

Problem: Can NF APIs integrated with CNCC be accessed through curl/postman ?

Solution

- NF configuration APIs can be accessed via CNCC GUI or directly using postman/curl.
- CNCC enforces authentication and authorization in both the flows.
- Steps are documented in *Cloud Native Core Console User's Guide*. about how to access APIs using postman/curl.

CNC Console Resource Requirement

This section includes information about CNC Console Resource Requirements.

CNC Console Resource Requirement

Table 1 CNC Console Resource Requirement

Micro-Service Name	CPU Per Pod (Maximum)	Memory Per Pod (GB) (Maximum)	Pod count(Maximum)	CPU All Pods - Maximum	Memory All Pods - Maximum (GB)
oc-cncc-iam-kc-http	2	2	1	2	2
oc-cncc-iam-ingress-gateway	4	4	1	4	4
oc-cncc-core-cmservice	2	2	1	2	2
oc-cncc-core-ingress-gateway	4	4	1	4	4
Total				12	12

CNC Console Microservices to Port Mapping

This section contains CNC Console microservices to port mapping details.

Serial Number	Service	Nature of Port	Nature of IP	Network Type	Port	Traffic Type	IPs Required	External IP
1	**oc-cncc-core-cmservice**	Internal	ClusterIP	Internal / K8s	8442/TCP	Configuration		
2	**oc-cncc-core-ingress-gateway**	External	LoadBalancer	RAN / F5	30075/TCP	Configuration	1	Yes
3	**oc-cncc-iam-kc-headless**	Internal	ClusterIP	Internal / K8s	8285/TCP	Configuration		
4	**oc-cncc-iam-kc-http**	Internal	ClusterIP	Internal / K8s	8285/TCP	Configuration		
5	**oc-cncc-iam-ingress-gateway**	External	LoadBalancer	RAN / F5	30085/TCP	Configuration	1	Yes