Oracle® Communications

Cloud Native Core Console Installation Guide

Release 1.1.0
F32119-01
May 2020

Oracle Communications Cloud Native Core Console Installation Guide, Release 1.1.0

F32119-01

Contents

5 CNC Console Core Installation Instructions

6 Post Installation Steps for CNC Console IAM

List of Tables

1

What's New in This Guide

The helm parameters and the pre-deployment configuration of following functionalities:

- **Hypertext Transfer Protocol Secure** (https) protocol
- **Ingress Gateway**

The configuration of the following Network Functions:

- **UDR** 1.6 is integrated to the GUI
- **CNPCRF** 1.6 is integrated to the GUI
- **NRF** and **PCF** is upgraded to version 1.6.1
- **SCP** is upgraded to version 1.6

2
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 Policy Control Function User's Guide
- Cloud Native Policy and Charging Rules Function (CNPCRF) 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
- Cloud Native Policy Control Function Installation Guide
- Unified Data Repository (UDR) Cloud Native Installation and Upgrade Guide
- Cloud Native Policy and Charging Rules Function (CNPCRF) Installation Guide

Acronyms

Table 2-1 Acronyms

Terms	Definition
CNCC	Cloud Native Core Console
NRF	Network Repository Function
OSDC	Oracle Software Delivery Cloud
SCP	Service Communication Proxy
PCF	Policy Control Function
IAM	Identity Access Management
UDR	Unified Data Repository
CNPCRF	Cloud Native Policy and Charging Rules Function
LDAP	Lightweight Directory Access Protocol
HTTPS	Hypertext Transfer Protocol Secure

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.

3
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).

Table 3-1 Installation Preparation

Step Number	Procedure	Description

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-1.1.0.0.0.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: 1. CNCC Docker Images File: `cncc-images-<tag>.tar` 2. Helm Chart of CNCC IAM: the tar ball contains Helm Chart and templates `cncc-iam-<tag>.tgz` 3. Helm File of CNCC Core: tarball contains Helm charts and templates `cncc-core-<tag>.tgz` 4. Readme txt File `Readme.txt` Example : List of contents in cncc-pkg-1.1.0.tgz : cncc-pkg-1.1.0.tgz |_ _ _ _ _ _ cncc-core-1.1.0.tgz |_ _ _ _ _ _ cncc-iam-1.1.0.tgz |_ _ _ _ _ _ cncc-images-1.1.0.tar |_ _ _ _ _ _ Readme.txt
3.	Check the checksums	Check the checksums of tarballs mentioned in `Readme.txt`

Table 3-1 (Cont.) Installation Preparation

Step Number	Procedure	Description

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-1.1.0.tar`
		Sample Output:
		`Loaded image: cncc/apigw-configurationinit:1.1.0`
		`Loaded image: cncc/apigw-configurationupdate:1.1.0`
		`Loaded image: cncc/cncc-apigateway:1.1.0`
		`Loaded image: cncc/cncc-cmservice:1.1.0`
		`Loaded image: cncc/cncc-iam:1.1.0`

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	Download the CNCC custom templates package file:
		
		`cncc-custom-configtemplates-<marketing-release-number>.zip`
		Example:
		`cncc-custom-configtemplates-1.1.0.zip`

Table 3-1 (Cont.) Installation Preparation

Step Number	Procedure	Description
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-1.1.0.zip Archive: cncc-custom-configtemplates-1.1.0.zip creating: cncc-custom-configtemplates-1.1.0/ inflating: cncc-custom-configtemplates-1.1.0/custom-cncc-iam_values_1.1.0.yaml inflating: cncc-custom-configtemplates-1.1.0/cnccdb_1.1.0.sql inflating: cncc-custom-configtemplates-1.1.0/custom-cncc-core_values_1.1.0.yaml

Verify and Create Kubernetes Namespace

This section explains how user can verify whether a required namespace exists in system or not. CNCC can be installed at NF specific namespaces also.

If namespace does not exist, user must create it.

Procedure

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

   ```
   $ kubectl get namespaces
   ```

2. If the output of the above command does not display the required namespace then create the namespace by executing following command:

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

Example:

```
$ kubectl create namespace cncc
```

> **Note:**
>
> Step 2 is an optional step. In case required namespace already exists, proceed with next procedures.

4
CNC Console IAM Installation Instructions

Prerequisites

Create MySQL Database and User

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

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. If MySql has CNCC database created as per release 1.0.0, drop it before creating cnccdb by executing the following command:

      ```
      DROP DATABASE cnccdb
      ```

   c. Execute `$ CREATE DATABASE IF NOT EXISTS <CNCC Database> CHARACTER SET utf8;` for Database creation.
   d. Grant permission to 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;
# Check if required database is already in list. If MySql has cnccdb
already created as per 1.0.0 release creation, drop it.
$ DROP DATABASE cnccdb;
# Database creation for CNCC
$ CREATE DATABASE IF NOT EXISTS cnccdb CHARACTERSET 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 polulate 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;
```

Create a Kubernetes Secret for MySQL

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

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

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

2. 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
   ```

Create a Kubernetes Secret for Admin User

This section describes how to create a kubernetes secret for admin user.

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

   ```
   $ 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

Installation Sequence for CNC Console IAM

Table 4-1 Installation Sequence for CNC Console IAM

SL. No	Phase
1	Installation Preparation
2	Configure `custom-cncc-iam_values_<version>.yaml` file. This includes configuring the following based on the deployment: 1. Repository path 2. cncc-iam details Note: Other configurations might be changed based on the deployment.

Table 4-1 (Cont.) Installation Sequence for CNC Console IAM

SL. No	Phase
3	cncc-iam deployment: 1. With helm repository 2. With helm tar
4	Verify cncc-iam deployment

Deployment of CNC Console IAM

Table 4-2 Deployment of CNC Console IAM

Step.No	Procedure	Description

1	Search helm chart	Execute the following command to check the version of the helm chart installation.
		`helm search <deployment_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.

Table 4-2 (Cont.) Deployment of CNC Console IAM

Step.No	Procedure	Description

3	Deploy cncc-iam	**Installation using helm repository**
		Execute the following command:
		`helm install <helm-repo> -f <custom-cncc-iam_values_<version>.yaml --name <deployment_name> --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
		deployment_name and **namespace_name** : depends on customer configuration
		Example: `helm install ocscp-helm-repo/ocscp -f custom-cncc-iam_values_1.1.0.yaml --name cncc-iam --namespace cncc-iam --version 1.1.0`
		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`
		Note: DB must be created before its DB account name is mentioned.

		Installation using helm tar Execute the following command: `helm install -f custom-cncc-iam_values_<version>.yaml --name cncc-iam --namespace <namespace> <chartpath>./<chart>.tgz` Example:`helm install -f custom-cncc-iam_values_1.1.0.yaml --name cncc-iam --namespace cncc-iam ./cncc-iam.tgz`
4	Check repository status	Execute the following command to check the deployment status. `helm status <deployment_name>`

Table 4-2 Deployment of CNC Console IAM

<table>
<tr><th>Step.No</th><th>Procedure</th><th>Description</th></tr>
<tr><td>5</td><td>Check service status</td><td>Check if all the services are deployed and running:
<code>kubectl -n <namespace_name> get services</code>
Example:
<code>$ kubectl -n cncc get services</code>
<table>
<tr><th>NAME</th><th>TYPE</th><th>CLUSTER-IP</th><th>EXTERNAL-IP</th><th>PORT(S)</th><th>AGE</th></tr>
<tr><td>cncc-iam-headless</td><td>ClusterIP</td><td>None</td><td><none></td><td>8285/TCP, 8443/TCP</td><td>9m13s</td></tr>
<tr><td>cncc-iam-http</td><td>ClusterIP</td><td>10.233.25.75</td><td><none></td><td>8285/TCP, 8443/TCP</td><td>9m13s</td></tr>
<tr><td>cncc-iam-ingress-gateway</td><td>LoadBalancer</td><td>10.233.7.236</td><td>10.75.182.72</td><td>8080:30346/TCP, 5701:32182/TCP</td><td>9m13s</td></tr>
</table></td></tr>
<tr><td>6</td><td>Check pod status</td><td>Check if all the pods are up and running by executing the following command:
<code>kubectl -n <namespace_name> get pods</code>
Example:
<code>$ kubectl -n cncc get pods</code>
<table>
<tr><th>NAME</th><th>READY</th><th>STATUS</th><th>RESTARTS</th><th>AGE</th></tr>
<tr><td>cncc-iam-0</td><td>1/1</td><td>Running</td><td>0</td><td>44h</td></tr>
<tr><td>cncc-iam-ingress-gateway-6748d55f98-szdqm</td><td>1/1</td><td>Running</td><td>0</td><td>12h</td></tr>
</table></td></tr>
</table>

CNC Console IAM Microservices

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

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

Following is an example of services CNC Console IAM offers:

Table 4-3 CNC Console IAM Microservices

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
cncc-iam-headless	ClusterIP	None	<none>	8285/TCP,8443/TCP	9m13s
cncc-iam-http	NodePort	10.233.25.75	<none>	8285/TCP,8443/TCP	9m13s
cncc-iam-ingress-gateway	LoadBalancer	10.233.7.236	10.75.182.72	8080:30346/TCP, 5701:32182/TCP	9m13s

Latest CNC Console IAM artifacts

The followings are the cncc-iam artifacts necessary to deploy cncc-iam in Cloud Native Environment. For this release, the artifacts are not present in the CGBU Artifactory but eventually cncc-iam 1.1.0 artifacts will be available in CGBU Artifactory.

Currently these artifacts can be accessed from CNE Lab Helm and Image repositories.

Development cncc-iam helm chart

```
Helm Chart name: cncc-iam
Helm Repository name: ocspf-helm-repo
Helm Chart version: 1.1.0
```

CNC Console IAM helm configurable values

`custom-cncc-iam_values_1.1.0.yaml` with helm chart version 1.1.0

```
keycloak:
  image:
    repository: ocspf-registry.us.oracle.com:5000/ocscp/cncc/cncc-iam
    tag: 1.1.0
    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:
    httpPort: 8285

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

ingress-gateway:
  global:
    # Docker registry name
    dockerRegistry: ocspf-registry.us.oracle.com:5000/ocscp

    # 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: false
    staticHttpNodePort: 30085
    staticHttpsNodePort: 30053

  image:
    # image name
    name: cncc/cncc-apigateway-api-tag
    # 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-init-tag
    # 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-update-tag
    # 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

  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

  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

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

  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

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

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>-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>-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

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
keycloak.image.repository	<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 repository that contains cncc-iam container image. It comprises of the following: <registry-url>:<registry-port>/<repo> e.g.: ocspf-registry.us.oracle.com:5000/cncc/cncc-iam

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
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.
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.
keycloak.username	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	The name of cncc-iam user as given by the user. Example: admin
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 Example: cncc-iam-secret
keycloak.serviceAccount.create	<Boolean>	It can take either True or False value. By default, it is false.	O	Flag for creating service account.
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>

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
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 Example: iamAdminPasswordKey
keycloak.persistence.dbVendor	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	The database vendor name Example: mysql
keycloak.persistence.dbName	<String>	Valid String	M	The name of the database used for cncc-iam. User should create DB with the same name as provided here before deploying CNCC-IAM Example: cnccdb
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 Example: mysql-sds.default.svc.cluster.local
keycloak.persistence.dbPort	<Integer>	It can range from 0-65535	M	The db port for cncc-iam Example: 3306
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 Example: cncc-db-secret
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	The key in the existing secret that stores the password Example: dbPasswordKey
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	The key in the existing secret that stores the username Example: dbUserNameKey

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
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	The port number which makes cncc-iam service visible to other services running within the same K8s cluster.
ingress-gateway.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	
ingress-gateway.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	
ingress-gateway.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.		
ingress-gateway.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
ingress-gateway.global.type	<String>	It can take ClusterIP, NodePort, LoadBalancer and ExternalName.	M	
ingress-gateway.global.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
ingress-gateway.global.metalLbIpAllocationAnnotation	<String>	metallb.universe.tf/address-pool: oam	M	Address Pool Annotation for Metallb

Attribute Name	DataType	Range	Mandatory(M) / Optional(O)/ Conditional(C)	Description
ingress-gateway.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
ingress-gateway.global.staticIpAddress	<String>	Static Ip	O	Static Ip and applicable only when ingress-gateway.global.staticNodePortEnabled is true.
ingress-gateway.global.staticNodePortEnabled	<Boolean>	It can take either True or False value. By default, it is false.	O	Node Port Enabled
ingress-gateway.global.staticHttpNodePort	<String>	Http Node Port	O	Http Node Port and applicable only when ingress-gateway.global.staticNodePortEnabled is true.
ingress-gateway.global.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.global.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
ingress-gateway.global.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
ingress-gateway.global.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.global.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.global.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.global.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
ingress-gateway.global.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.global.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.global.service.ssl.tlsVersion		Default Value is TLSv1.2	M	TLS Version

Attribute Name	DataType	Range	Mandatory(M) / Optional(O)/ Conditional(C)	Description
ingress-gateway.global.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 Example: cncc-iam-ingress-secret
ingress-gateway.global.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 Example: cncc
ingress-gateway.global.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 Example: rsa_private_key_pkcs1.pem
ingress-gateway.global.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 Example: ssl_ecdsa_private_key.pem
ingress-gateway.global.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 Example: cncc-iam-ingress-secret
ingress-gateway.global.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 Example: cncc
ingress-gateway.global.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 Example: ssl_rsa_certificate.crt
ingress-gateway.global.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 Example: ssl_ecdsa_certificate.crt

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.global.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 Example: cncc-iam-ingress-secret
ingress-gateway.global.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 Example: cncc
ingress-gateway.global.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 Example: caroot.cer
ingress-gateway.global.service.ssl.initialAlgorithm	<String>	Default values is RSA256	M	
ingress-gateway.global.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 Example: cncc-iam-ingress-secret
ingress-gateway.global.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 Example: cncc
ingress-gateway.global.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 Example: ssl_keystore.txt
ingress-gateway.global.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 Example: cncc-iam-ingress-secret

Attribute Name	DataType	Range	Mandatory(M) / Optional(O)/ Conditional(C)	Description
ingress-gateway.global.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 Example: cncc
ingress-gateway.global.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 Example: ssl_truststore.txt
ingress-gateway.global.ports.containerPort	<String>	It can take value in the range: 0-65535.	M	ContainerPort represents a network port in a single container
ingress-gateway.global.ports.containersslPort	<String>	Default value is 8443	M	
ingress-gateway.global.ports.actuatorPort	<String>	Default value is 9090		
ingress-gateway.global.log.level.root	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc.	M	The level at which user wants to see the logs. E.g. WARN
ingress-gateway.global.log.level.ingress	<String>	It can take values like: WARN, DEBUG, INFO, TRACE etc. Default value is INFO	M	Log level for ingress logs
ingress-gateway.global.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.global.readinessProbe.timeoutSeconds	<String>	It can take value in the range: 0-65535. Default value:3	M	The number of seconds after which the probe times out
ingress-gateway.global.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

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.global.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.global.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.global.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.global.livenessProbe.timeoutSeconds	<String>	It can take value in the range: 0-65535. Default value:3	M	The number of seconds after which the probe times out
ingress-gateway.global.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.global.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
ingress-gateway.global.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.global.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.global.resources.limits.initServiceCpu	<String>	Default value is 1	M	Init Container CPU Limit

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.global.resources.limits.updateServiceCpu	<String>	Default value is 1	M	Update Container CPU Limit
ingress-gateway.global.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.global.resources.limits.updateServiceMemory	<String>	Default value is 1Gi	M	Update Container Memory Limit
ingress-gateway.global.resources.limits.initServiceMemory	<String>	1Gi	M	Init Container Memory Limit
ingress-gateway.global.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.global.resources.requests.initServiceCpu	<String>	Default value is 1	M	Init Container CPU Limit
ingress-gateway.global.resources.requests.updateServiceCpu	<String>	Default value is 1	M	Update Container CPU for requests
ingress-gateway.global.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'.

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.global.resources.requests.updateServiceMemory	<String>	1Gi	M	Update Container Memory for requests
ingress-gateway.global.resources.requests.initServiceMemory	<String>	1Gi	M	Init Container Memory for requests
ingress-gateway.global.resources.target.averageCpuUtil	<String>	A value in between 0-100	M	It gives the average CPU utilization percentage.
ingress-gateway.global.minAvailable	<String>	It can take value in the range: 0-65535. Default value:1	M	the number of pods that must always be available, even during a disruption.
ingress-gateway.global.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.global.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.global.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.global.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.global.enableIncomingHttps	<String>	It can take either True or False value. By default, it is false.	M	Server Configuration for http and https support

Attribute Name	DataType	Range	Mandatory(M) / Optional(O)/ Conditional(C)	Description
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_RS A_WITH_AES_256_GCM_ SHA384 TLS_ECDHE_ ECDSA_WITH_AES_128_ GCM_SHA25 6 TLS_ECDHE_ RSA_WITH_ AES_128_GC M_SHA256	M, if ingressgateway.enableIncomingHttpsis true	Allowed CipherSuites for TLS1.2
ingress-gateway.global.ingressGwCertReloadEnabled	<boolean>	It can take either True or False value. Default value is true	M	
ingress-gateway.global.ingressGwCertReloadPath	<String>		M	
ingress-gateway.global.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.global.routesConfig.[].uri	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	
ingress-gateway.global.routesConfig.[].path	<String>	Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods and dashes.	M	
ingress-gateway.global.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	

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.global.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	

CNC Console IAM service Access

CNC Console IAM services can be accessed as follows:

```
<scheme>://<cncc-iam-ingress-extrenal-ip>:<cncc-iam-ingress-service-port>
```

Example:

```
http://10.75.182.72:8080/*
```

CNC Console-IAM Upgrade

CNC Console-IAM can be upgraded as follows:

Table 4-4 CNC Console-IAM Upgrade

Step Number	Procedure	Description
1	Upgrade CNC Console-IAM	After upgrading `custom-cncconsole_values.yaml` file, execute the following command to upgrade CNC Console-IAM `$ helm upgrade -f custom-cncc-iam_values.yaml cncc-iam ocspf-helm-repo/cncc-iam`

CNC Console-IAM Uninstall

CNC Console-IAM can be uninstalled as follows:

Table 4-5 CNC Console-IAM Uninstall

Step Number	Procedure	Description

1	Un-deploy CNCConsole-IAM	Execute the following command to uninstall CNC Console-IAM:
		`$ helm delete <deployment name> --purge`
		Example:
		`$ helm delete cncc-iam --purge`

5
CNC Console Core Installation Instructions

Prerequisites for CNCC Core Installation

- 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

Installation Sequence for CNCC Core

Table 5-1 Installation Sequence for CNC Console

SL.No	Phase
1	Installation Preparation
2	Configure `custom-cncc-core_values.yaml` file. This includes configuring the following based on the deployment: 1. Repository path 2. Domain and clusterdomain 3. CNC Console details **Note:** Other configurations might be changed based on the deployment.
3	CNC Console deployment: 1. With helm repository 2. 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.

Table 5-2 CNCC Core Deployment

Step No.	Procedure	Description

1.	Search helm chart	Execute the following command to search helm chart.
		`helm search <deployment_name>`
		
		`Example: helm search cncc-core`
		
		`NAME CHART VERSION APP VERSION DESCRIPTION`
		
		`ocspf-helm-repo/cncc-core 1.1.0 1.0 A Helm chart for CNC Console`

Table 5-2 (Cont.) CNCC Core Deployment

Step No.	Procedure	Description
2.	Prepare `custom-cncc-core_values.yaml` file	Prepare a `custom-cncc-core_values.yaml` file with the required parameter information. **Note**: • The user needs to update the "domain" in the `custom-cncc-core_values.yaml` file per the name of cluster (default value of domain is "svc.cluster.local"). If the cluster name is XYZ then domain must be svc.XYZ. • The user needs to update the "clusterDomain" in the `custom-cncc-core_values.yaml` file per the name of cluster (default value of domain is "cluster.local"). If the cluster name is XYZ then domain must be XYZ.

3.	Deploy CNCC Core	**Installation using helm repository**
		Execute the following command:
		`helm install <helm-repo> -f custom-cncc-core_values.yaml --name <deployment_name> --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
		deployment_name and **namespace_name**: depends on customer configuration
		Example:
		`helm install ocscp-helm-repo/ocscp -f custom-cncc-core_values.yaml --name cncc-core --namespace cncc --version 1.1.0`

		Installation using helm tar Execute the following command: `helm install -f custom-cncc-core_values.yaml --name cncc-core --namespace <namespace> <chartpath>./<chart>.tgz`
4.	Check repository status	Execute following command to check the deployment status. `helm status <deployment_name>`

Table 5-2 CNCC Core Deployment

<table>
<tr><th>Step No.</th><th>Procedure</th><th>Description</th></tr>
<tr><td>5.</td><td>Check service status</td><td>Check if all the services are deployed and running:

kubectl -n <namespace_name> get services

Example: $ kubectl -n cncc get services

<table>
<tr><td>cncc-core-cmservice</td><td>ClusterIP</td><td>10.233.13.43</td><td><none></td><td>8442/TCP</td><td>6m13s</td></tr>
<tr><td>cncc-core-ingress-gateway</td><td>LoadBalancer</td><td>10.233.11.14</td><td>10.75.182.79</td><td>8080:31417/TCP, 5701:30487/TCP</td><td>6m13s</td></tr>
</table>
</td></tr>
<tr><td>6.</td><td>Check pod status</td><td>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

<table>
<tr><th>NAME</th><th>READY</th><th>STATUS</th><th>RESTARTS</th><th>AGE</th></tr>
<tr><td>cncc-core-cmservice-7f8b57c5bf-p4gvw</td><td>1/1</td><td>0</td><td>0</td><td>6m18s</td></tr>
<tr><td>cncc-core-ingress-gateway-5bfb8789cd-wls5p</td><td>1/1</td><td>0</td><td>0</td><td>6m18s</td></tr>
</table>
</td></tr>
</table>

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 5-3 CNCC Core Microservices

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
cncc-core_cmservice	ClusterIP	10.233.13.43	<none>	8442/TCP	6m13s

Table 5-3 (Cont.) CNCC Core Microservices

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
cncc-core-ingress-gateway	LoadBalancer	10.233.13.43	10.75.182.79	8080:31417/TCP, 5701:30487/TCP	6m13s

Latest CNCC Core Artifacts

The following are the CNCC Core artifacts necessary to deploy CNCC CoRE in Cloud Native Environment. For this release, the artifacts are not present in the CGBU Artifactory but eventually CNCC Core 1.1.0 artifacts will be available there.

For this release, these artifacts could be accessed from CNE Lab Helm and Image repositories.

Development CNCC HELM Chart

```
Helm Chart name: CNC Console Core

Helm Repo name: ocspf-helm-repo

Helm Chart version: 1.1.0
```

CNCC Core Helm Configurable Values

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

```
global:
  cnccServiceAccountName: ""

cmservice:
  dockerRegistry: ocspf-registry.us.oracle.com:5000
  image:
    name: ocscp/cncc/cncc-cmservice-cm-tag
    tag: helm-tag
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
    pullPolicy: Always
  resources:
    limits:
      cpu: 2
      memory: 2Gi
    requests:
      cpu: 1
      memory: 1Gi
  deployment:
    envManageNF: SCP, NRF, UDR, PCF, PCRF, POLICY, CONVERGED
    # This is the name of product which appears as brand name and can be
used to mention site name as well.
    # Example: envSystemName: CNCC - Site Name
```

```
    envSystemName: CNCC
    # This is the name of the Project that appears on the Window
    # Example: cmWindowName: CNCC
    cmWindowName: CNCC
    # Applicable only for PCF and PCRF deployment, this enables Import
Export buttons. Make cmEnableImportExport : true in case of PCF or PCRF
deployment
    cmEnableImportExport: false
    nodeSelectorEnabled: false
    nodeSelectorKey: zone
    nodeSelectorValue: app
  service:
    http:
      port: 8442
    type: ClusterIP

ingress-gateway:
  global:
    # Docker registry name
    dockerRegistry: ocspf-registry.us.oracle.com:5000/ocscp

    # 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: ""

    #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: false
    staticHttpNodePort: 30075
    staticHttpsNodePort: 30043

  image:
    # image name
    name: cncc/cncc-apigateway-api-tag
    # 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-init-tag
  # 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-update-tag
  # 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
```

```
  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

  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

  # To Initialize SSL related infrastructure in init/update container
  initssl: false
  #Server Configuration for http and https support
  enablehttp1: true
  enableIncomingHttp: true # 'true' only in case of http
  enableIncomingHttps: false # 'true' only in case of https

  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

  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-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: udr_configuration_1
    uri: http://<FQDN>:<PORT>
    path: /nudr-dr-prov/**
  - id: udr_configuration_2
    uri: http://<FQDN>:<PORT>
    path: /nudr-dr-mgm/**
  - id: udr_configuration_3
    uri: http://<FQDN>:<PORT>
    path: /nudr-group-id-map-prov/**
  - id: pcf_configuration
    uri: http://<FQDN>:<PORT>
    path: /pcfapi/**
    filters:
      rewritePath: "/pcfapi(?<segment>/?.*), $\\{segment}"
  - id: cnpcrf_configuration
    uri: http://<FQDN>:<PORT>
    path: /pcrfapi/**
    filters:
      rewritePath: "/pcrfapi(?<segment>/?.*), $\\{segment}"
  - id: policy_configuration
    uri: http://<FQDN>:<PORT>
    path: /policyapi/**
    filters:
      rewritePath: "/policyapi(?<segment>/?.*), $\\{segment}"
  - id: converged_configuration
    uri: http://<FQDN>:<PORT>
    path: /convergedapi/**
    filters:
      rewritePath: "/convergedapi(?<segment>/?.*), $\\{segment}"
  - id: default_configuration # Default configuration should be the last
routesConfig entry
    uri: http://<helmrelease>-cmservice.<namespace>.<domain>:8442
    path: /**

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

  # CNCC configuration
  cncc:
    # Enable cncc feature including iam
    enabled: 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 and
service port (e.g. http://10.75.182.72:8080)
    iam:
      uri: http://<IP>:<PORT>
```

> **Note:**
>
> The field `ingress-gateway.cncc.iam.uri` must include the CNCC IAM Console URL. Check Accessing CNCC IAM Services for the URL.
>
> For PCF or CNPCRF deployment set `cmEnableImportExport:true`, this enables Import Export buttons. It is applicable only for PCF or CNPCRF deployment.

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.cnccServiceAccountName	<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>
cmservice.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 cmservice's container image. It comprises of the following: <registry-url>:<registry-port> Example:: ocspf-registry.us.oracle.com:5000
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.

Attribute Name	DataType	Range	Mandatory(M) / Optional(O)/ Conditional(C)	Description
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.
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 NFs Example: SCP, PCF	M	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. Example: envSystemName: CNCC
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. Example: 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.service.http.port	<Integer>	It can take value in the range: 0-65535	O	The port number which makes cmservice visible to other services running within the same K8s cluster
cmservice.service.type	<String>	It can take only 'ClusterIP' as the value.	O	Used to decide where user wants to expose the service from outside the Kubernetes cluster or not.
ingress-gateway.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 CNC Console Core's container image. It comprises of the following: <registry-url>:<registry-port>/<repo> Example:: ocspf-registry.us.oracle.com:5000/cncc
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	The image name of the ingress-gateway as provided by the user

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
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.publicHttpSignalingPort	<Integer>	It can take value in the range: 0-65535	O	The port on which ingress-gateway service is exposed # If httpsEnabled is false, this Port would be HTTP/2.0 Port (unsecured) publicHttpSignalingPort: 80
ingress-gateway.publicHttpsSignallingPort	<Integer>	It can take value in the range: 0-65535.	O	The port on which ingress-gateway service is exposed # If httpsEnabled is true, this Port would be HTTPS/2.0 Port (secured SSL).
ingress-gateway.metalLbIpAllocationEnabled	<Boolean>	True/False By default, it is true.	O	This field enables or disables IP Address allocation from Metallb Pool
ingress-gateway.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"		The address Pool Annotation for Metallb

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.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.
ingress-gateway.staticIpAddress	<Stirng>	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	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.
ingress-gateway.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.
ingress-gateway.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.
ingress-gateway.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.
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.type	<String>	It can take value LoadBalance/ NodePort depending upon one wants to expose the service from outside the Kubernetes cluster or not.	O	Used to decide where user wants to expose the service from outside the Kubernetes cluster or not.
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	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 Example: 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 Example: 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 Example: 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 Example: 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 Example: 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 Example: 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 Example: 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.	O	ecdsa certificate file name Example: ssl_ecdsa_certificate.crt

Attribute Name	DataType	Range	Mandatory(M) / Optional(O)/ Conditional(C)	Description
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 Example: 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 Example: 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 Example: caroot.cer
ingress-gateway.service.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 Example: cncc-core-ingress-secret
ingress-gateway.service.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 Example: cncc
ingress-gateway.service.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 Example: ssl_keystore.txt
ingress-gateway.service.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 Example: cncc-core-ingress-secret
ingress-gateway.service.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 Example: cncc

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.service.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 Example: ssl_truststore.txt
ingress-gateway.service.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.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	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	The number of seconds after which the probe times out

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
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.
ingress-gateway.minAvailable	<Integer>	It can take value in the range: 0-65535. Default value:1	O	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

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
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_RS A_WITH_AES_256_GCM_ SHA384 TLS_ECDHE_ ECDSA_WITH_AES_128_ GCM_SHA25 6 TLS_ECDHE_ RSA_WITH_ AES_128_GC M_SHA256	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.
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	The URI of the cncc-iam ingress.
ingress-gateway.service.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	It is used to provide the name of the api-gateway service.
ingress-gateway.service.staticNodePortEnabled	<boolean>	It can take either True or False value. By default, it is false.	O	The flag for enabling/ disabling the static Nodeport for api-gateway service.
ingress-gateway.ports.containerPort	<Integer>	It can take value in the range: 0-65535. Default value: 8081	O	The http port of the container for the ingress-gateway.

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ingress-gateway.ports.containerslPort	<Integer>	It can take value in the range: 0-65535. Default value: 8443	O	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	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	The level at which user wants to see the logs. Example: 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.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.
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	

Attribute Name	DataType	Range	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
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	

CNCC Core Service Access

CNCC Core service can be accessed by following URL:

```
<scheme>://<cncc-core-ingress-extrenal-ip>:<cncc-core-ingress-service-port>
```

Example: `http://10.75.182.79:8080`

> **Note:**
>
> envSystemName can be used to display site name. Example:
> `envSystemName: CNCC - Site Name`

> **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 1.1 Post Installation Steps for CNCC-IAM

CNCC Core Upgrade

CNCC Core can be upgraded as follows:

Table 5-4 CNCC Core Upgrade

Step Number	Procedure	Description
1.	Upgrade CNCC Core	After upgrading `custom-cncc-core_values.yaml` file, execute the following command to upgrade CNCC Core: `$ helm upgrade -f custom-cncc-core_values.yaml cncc-core ocspf-helm-repo/cncc-core`

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:

Table 5-5 CNCC Core Uninstall

Step Number	Procedure	Description

1.	Un-deploy CNCC Core	Execute the following command to uninstall CNCC Core:
		
		`$ helm delete <deployment name> --purge`
		`Example:`
		`$ helm delete cncc-core --purge`

CNCC Supported NFs and Version Compatibility

SR.No	NF	NF Version	Enabling NFs	Route Configuration	Remarks

1	SCP	1.6.0	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/**`	

SR.No	NF	NF Version	Enabling NFs	Route Configuration	Remarks

2	NRF	1.6.1	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/**`	

3	PCF	1.6.1	Update in cmservice section of values.yaml
			`envManageNF:PCF,POLICY,CONVERGED`
				Update in routeConfig under ingress section in values.yaml
				
				`- id: pcf_configuration`
				`uri: http://<FQDN>:<PORT>`
				`path: /pcfapi/**`
				`filters:`
				`rewritePath: "/pcfapi(?<segment>/?.*), $\\{segment}"`
				`- id: policy_configuration`
				`uri: http://<FQDN>:<PORT>`
				`path: /policyapi/**`
				`filters:`
				`rewritePath: "/policyapi(?<segment>/?.*), $\\{segment}"`
				`- id: converged_configuration`
				`uri: http://<FQDN>:<PORT>`
				`path: /convergedapi/**`
				`filters:`
				`rewritePath: "/convergedapi(?<segment>/?.*), $\\{segment}"`
					For enabling PCF, add POLICY, CONVERGED in "envManageNF" field.

SR.No	NF	NF Version	Enabling NFs	Route Configuration	Remarks

5	PCRF	1.6.0	Update in cmservice section of values.yaml
			`envManageNF:PCRF,POLICY,CONVERGED`
				Update in routeConfig under ingress section in values.yaml
				
				`- id: pcrf_configuration`
				`uri: http://<FQDN>:<PORT>`
				`path: /pcrfapi/**`
				`filters:`
				`rewritePath: "/pcrfapi(?<segment>/?.*), $\\{segment}"`
				`- id: policy_configuration`
				`uri: http://<FQDN>:<PORT>`
				`path: /policyapi/**`
				`filters:`
				`rewritePath: "/policyapi(?<segment>/?.*), $\\{segment}"`
				`- id: common_policy_configuration`
				`uri: http://<FQDN>:<PORT>`
				`path: /convergedapi/**`
				`filters:`
				`rewritePath: "/convergedapi(?<segment>/?.*), $\\{segment}"`
					For enabling PCRF, add POLICY, CONVERGED in "envManageNF" field.

SR.No	NF	NF Version	Enabling NFs	Route Configuration	Remarks

6	UDR	1.6.0	Update in cmservice section of values.yaml
			`envManageNF:UDR`
				Update in routeConfig under ingress section in values.yam
				
				`- id: udr_configuration_1`
				`uri: http://<FQDN>:<PORT>`
				`path: /nudr-dr-prov/**`
				`- id: udr_configuration_2`
				`uri: http://<FQDN>:<PORT>`
				`path: /nudr-dr-mgm/**`
				`- id: udr_configuration_3`
				`uri: http://<FQDN>:<PORT>`
				`path: /nudr-group-id-map-prov/**`
				
				I	

6

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

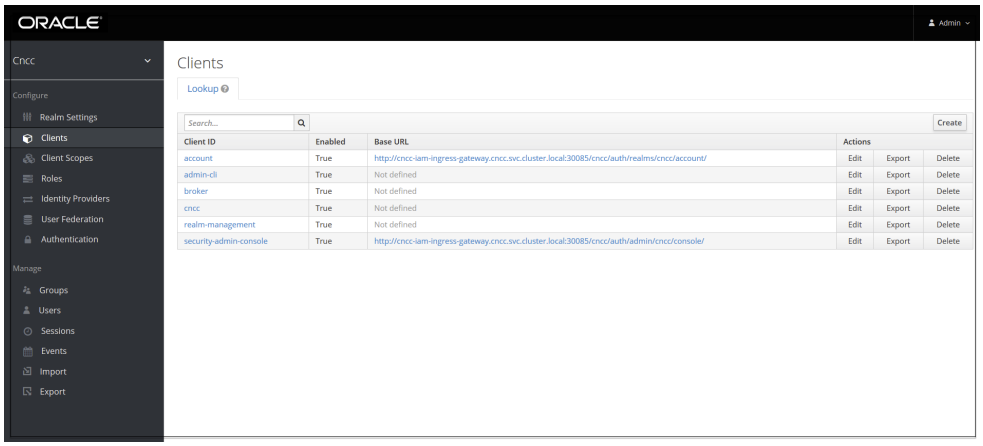
  Once CNCC IAM is deployed admin must do the following:
- Set the cncc redirection URL.
- Create the user and assign the roles (only applicable if not integrated with LDAP).

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.

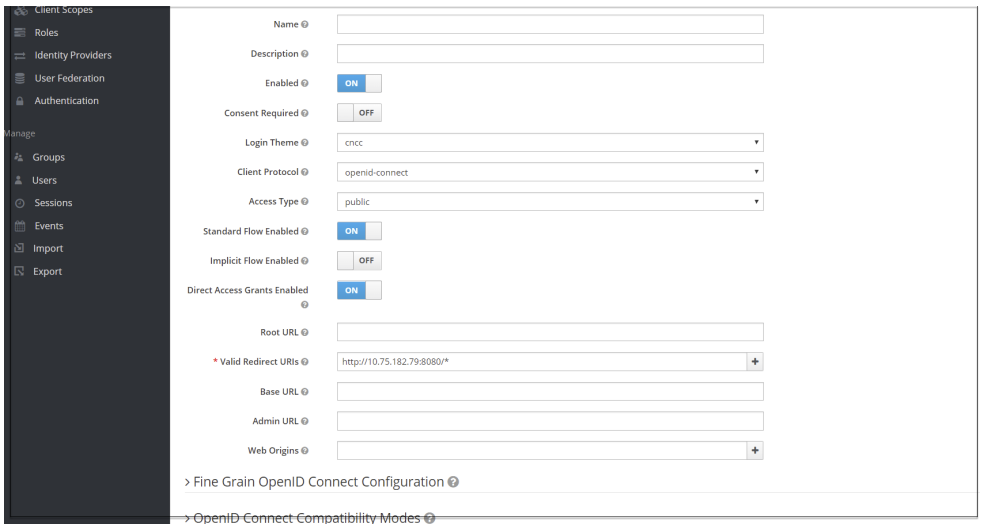
   ```
   <scheme>://<cncc-iam-ingress-extrenal-ip>:<cncc-iam-ingress-service-port>

   Example: http://10.75.182.72:8080/*
   ```

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

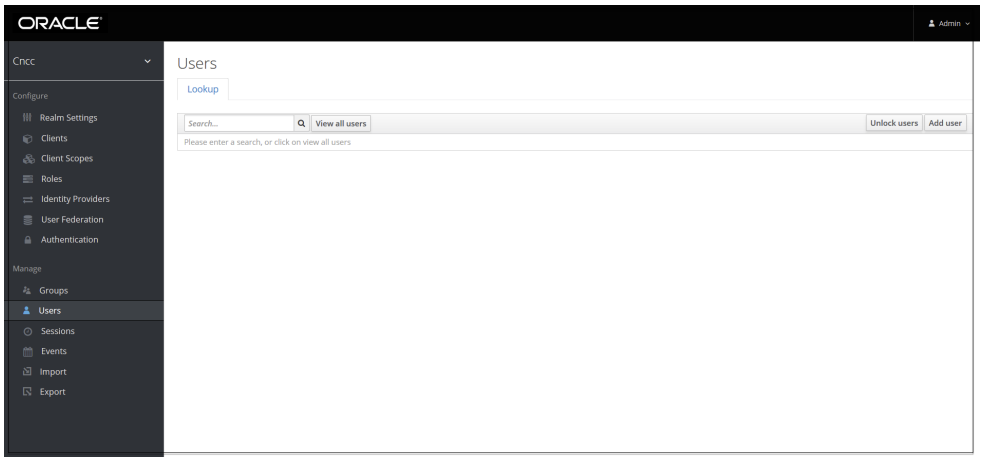


3. Enter CNCC Core Ingress URI in the **Valid Redirect URIs** field and **Save.**

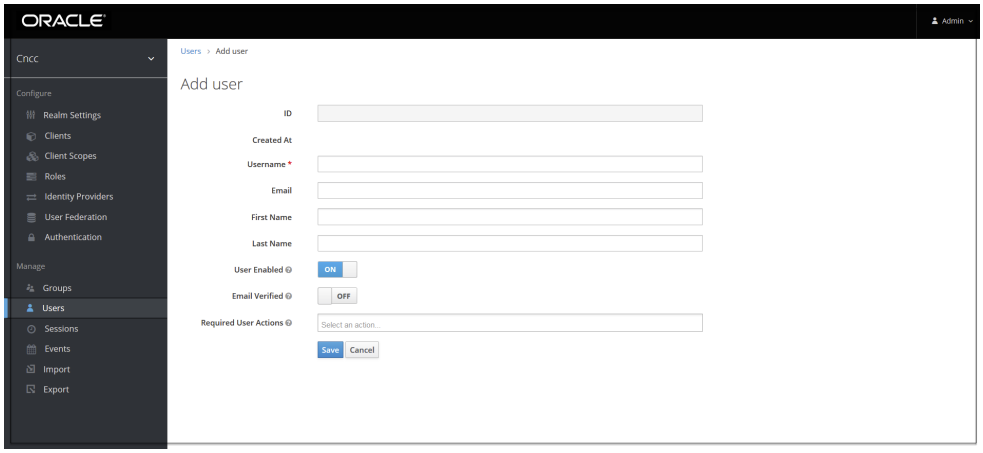
```
<scheme>://<cncc-core-ingress-extrenal-ip>:<cncc-core-ingress-service-port>/*
 Example:  http://10.75.182.79:8080/*
```



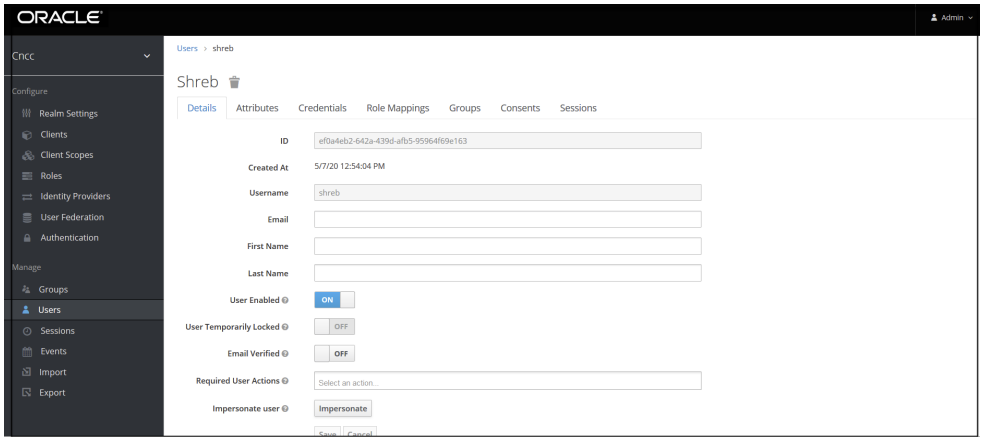
4. Select **Manage** and click **Users** and select **Add user** in the right pane.



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

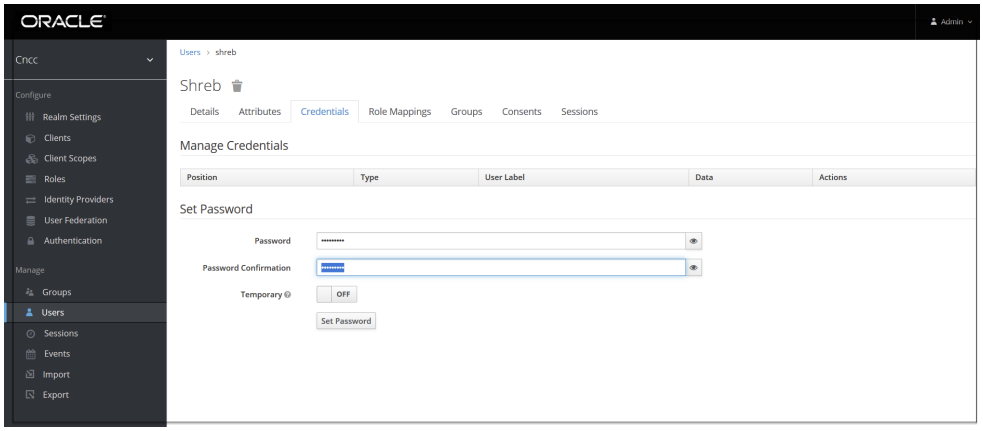


6. The user has been created and the user Details screen appears.

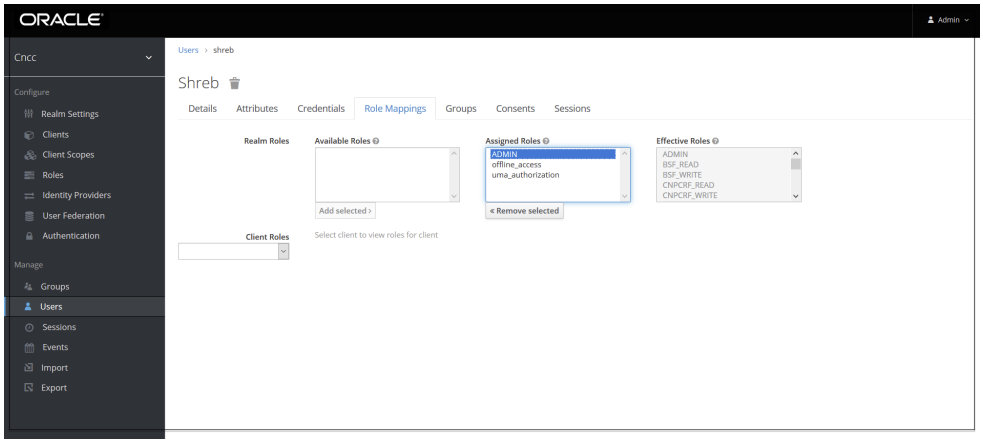


7. For setting the password for the user, select **Credentials** tab and set the password for that user.

> **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.