Oracle® Communications

Network Repository Function (NRF) Cloud Native Installation and Upgrade Guide

Release 1.6.1
F31207-02
June 2020

ORACLE

Oracle Communications Network Repository Function (NRF) Cloud Native Installation and Upgrade Guide, Release 1.6.1

F31207-02

Contents

List of Tables

My Oracle Support

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

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

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

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

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

What's New in This Guide

This section introduces the documentation updates performed in Oracle Communications Cloud Native Network Repository Function (NRF) Installation Guide.

New or Updated Features in Release 1.6.1

The following updates are performed in release 1.6.1:

- Updated the sample file in Creating Service Account, Role and Role bindings.
- The helm charts, parameters and file names are updated for Release 1.6.1.

New or Updated Features in Release 1.6.0

The following updates are performed in release 1.6.0:

- **nrfInstanceId**: Added the parameter in **nfaccesstoken**.
- Removed **apigateway** section.
- Updated pre-configuration steps with secret update procedures.
- The helm charts, parameters and file names are updated for Release 1.6.0.

1 OCNRF Overview

This section includes information about the role of Oracle Communications Network Repository Function (OCNRF) in 5G Service Based Architecture.

The OCNRF is one of the main components of the 5G Service Based Architecture. The OCNRF maintains an updated repository of all the Network Functions (NFs) available in the operator's network along with the services provided by each of the NFs in the 5G core that are expected to be instantiated, scaled and terminated with minimal or no manual intervention.

The OCNRF supports discovery mechanisms that allow NFs to discover each other and get updated status of the desired NFs.

The OCNRF supports the following functions:

- Maintains the profiles of the available NF instances and their supported services in the 5G core network.
- Allows consumer NF instances to discover other provider's NF instances in the 5G core network.
- Allows NF instances to track the status of other NF instances.
- Provides Oauth2 based Access Token service for consumer NF authorization.
- Provides specific NF Type selection based on subscriber identity.
- Supports forwarding of messages from one NRF to another NRF.

The OCNRF interacts with every other Network Function in the 5G core network and it supports the above functions through the following services:

- Management Services
- Discovery Services
- AccessToken Service

References

- Cloud Native Environment Installation Document 1.4
- Network Repository Function (NRF) User's Guide

Acronyms

The following table provides information about the acronyms and the terminology used in the document.

Table 1-1 Acronyms

Field	Description
5G System	3GPP system consisting of 5G Access Network (AN), 5G Core Network and UE
5G-AN	5G Access Network
5GC	5G Core Network
5G-NF	5G Network Function
AMF	Access and Mobility Management Function
API-Gateway	Application Program Interface Gateway
CNE	Cloud Native Environment
FQDN	Fully Qualified Domain Name
K8s	Kubernetes
MMI	Machine Machine Interface
MPS	Messages Per Second
NDB	Network Database
NF	Network Function
Network Function	A functional building block within a network infrastructure, which has well defined external interfaces and well defined functional behavior. In practical terms, a network function is often a network node or physical appliance.
Network Slice	A logical network that provides specific network capabilities and network characteristics.
Network Slice instance	A set of Network Function instances and the required resources (e.g. compute, storage and networking resources) which form a deployed Network Slice.
NF Consumer	A generic way to refer to an NF which consumes services provided by another NF. Ex: An AMF is referred to as a Consumer when it consumes AMPolicy services provided by the PCF.
NF Instance	A specific instance of a network function type.
NF Producer or NF Provider	A generic way to refer to an NF which provides services that can be consumed by another NF. Ex: A PCF is a provider NF and provides AMPolicy Services
NRF	Network Repository Function
OCNRF	Oracle Communications Network Repository Function
OHC	Oracle Help Center
OSDC	Oracle Software Download Center
PLMN	Public Land Mobile Network
Resiliency	The ability of the NFV framework to limit disruption and return to normal or at a minimum acceptable service delivery level in the fame of a fault, failure, or an event that disrupts normal operation.
Scaling	Ability to dynamically extend/reduce resources granted to the Virtual Network Function (VNF) as needed. This includes scaling out/in or scaling up/down.
Scaling Out/In/ Horizontally	The ability to scale by add/remove resource instances (e.g. VMs). Also called scaling Horizontally.
Scaling Up/Down/ Vertically	The ability to scale by changing allocated resources, e.g. increase/decrease memory, CPU capacity or storage size.

Table 1-1 (Cont.) Acronyms

Field	Description
PCF	Policy Control Function
SEPP	Security Edge Protection Proxy
SCP	Service Communication Proxy
SLF	Subscriber Location Function
URI	Universal Resource Identifier

2
Installing OCNRF

This section describes the prerequisites and installation procedure for OCNRF.

Prerequisites

This section includes information about the required prerequisites before initiating OCNRF Installation.

Following are the prerequisites to install and configure OCNRF:

OCNRF Software

The OCNRF software includes:

- OCNRF Helm charts
- OCNRF docker images

The following softwares must be installed before installing OCNRF:

Software	Version
Kubernetes	v1.15.3
HELM	v2.14.3

Additional software that needs to be deployed as per the requirement of the services:

Software	Chart Version	Notes
elasticsearch	5.5.4	Needed for Logging Area
elastic-curator	5.5.4	Needed for Logging Area
elastic-exporter	1.0.2	Needed for Logging Area
logs	2.0.7	Needed for Logging Area
kibana	6.7.0	Needed for Logging Area
grafana	6.1.6	Needed for Metrics Area
prometheus	9.1.2	Needed for Metrics Area
prometheus-node-exporter	0.17.0	Needed for Metrics Area
metallb	0.7.3	Needed for External IP
metrics-server	0.3.1	Needed for Metric Server
tracer	0.8.3	Needed for Tracing Area

> **Note:**
>
> Install the specified software items before proceeding, if any of the above services are needed and the respective software is not already installed in CNE.
> To check the installed software items, execute:
>
> ```
> helm ls
> ```
>
> Some of the systems may need to use helm command with `admin.conf` file, such as:
>
> ```
> helm --kubeconfig admin.conf
> ```

Network access

The Kubernetes cluster hosts must have network access to:

- Local docker image repository where the OCNRF images are available.
  To check if the Kubernetes cluster hosts has network access to the local docker image repository, try to pull any image with tag name to check connectivity by executing:

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

  > **Note:**
  >
  > Some of the systems may need to use helm command with `admin.conf` file, such as:
  >
  > ```
  > helm --kubeconfig admin.conf
  > ```

- Local helm repository where the OCNRF helm charts are available.
  To check if the Kubernetes cluster hosts has network access to the local helm repository, execute:

  ```
  helm repo update
  ```

  > **Note:**
  >
  > Some of the systems may need to use helm command with `admin.conf` file, such as:
  >
  > ```
  > helm --kubeconfig admin.conf
  > ```

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

Client machine requirement

Client machine needs to have the following minimum requirements:

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

Server or Space Requirements

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

Secret file requirement

For HTTPs and Access token, the following certs and pem files has to be created before creating secret files for Keys and MySql.

Note: The following files must be created before creating secret files.

1. ECDSA private Key and CA signed ECDSA Certificate (if initialAlgorithm: ES256)
2. RSA private key and CA signed RSA Certificate (if initialAlgorithm: RSA256)
3. TrustStore password file
4. KeyStore password file
5. CA signed ECDSA certificate

ServiceAccount requirement

Operator must create a service account, bind it with a Role for resource with permissions for atleast get, watch and list.

serviceAccountName is a mandatory parameter. Kubernetes Secret resource is used for providing the following:

- MYSQL DB Details to micro-services.
- NRF's Private Key, NRF's Certificate and CA Certificate Details to Ingress/Egress Gateway for TLS.
- NRF's Private and NRF's Public Keys to nfAccessToken micro-service for Digitally Signing AccessTokenClaims.

- Producer/Consumer NF's Service/Endpoint details for routing messages from/to Egress/Ingress Gateway.

The Secret(s) can be under same namespace where OCNRF is getting deployed (recommended) or # Operator can choose to use different namespaces for different secret(s). If all the Secret(s) are under same namespace as OCNRF, then Kubernetes Role can be binded with the given ServiceAccount. Otherwise ClusterRole needs to be binded with the given ServiceAccount. The Role/ClusterRole needs to be created with resources: (services, configmaps, pods, secrets, endpoints) and (verbs: get, watch, list). Refer to Creating Service Account, Role and Role bindings for more details.

Installation Sequence

This section explains the tasks to be performed for installing OCNRF.

OCNRF pre-deployment configuration

This chapter divided into different sections:

1. Creating OCNRF namespace

> **Note:**
>
> This is a mandatory procedure, execute this before proceeding any further. The namespace created/verified in this procedure is an input for next procedures.

2. Creating Service Account, Role and Role bindings

> **Note:**
>
> Following are sample steps, in case of already configured service account with role and role-bindings or the user has previously prepared procedure to create service account, skip this procedure.

3. Configuring MySql database and user
4. Configuring MySQL secret
5. Configuring secrets for enabling HTTPS
6. Configuring secret for enabling AccessToken service

Creating OCNRF namespace

This section explains how the user can verify if the required namespace is available in the system or not. If namespace does not exists, the user must create it.

Procedure

1. Verify required namespace already exists in system:

```
$ kubectl get namespaces
```

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

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

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

For example:-

```
$ kubectl create namespace ocnrf
```

Creating Service Account, Role and Role bindings

This section explains how user can create service account, required role and role bindings resources. Sample templates for the resources is as follows. Sample template can be filled inline and can be added into sample resource input yaml file. Input file name example:- *ocnrf-sample-resource-template.yaml*

Example command for creating the resources

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

Sample template to create the resources

> **Note:**
>
> Update **<helm-release>** and **<namespace>** with respective ocnrf namespace and planned ocnrf helm release name in below place holders.

```
## Sample template start#
apiVersion: v1
kind: ServiceAccount
metadata:
  name: <helm-release>-ocnrf-serviceaccount
  namespace: <namespace>
---

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: <helm-release>-ocnrf-role
  namespace: <namespace>
rules:
- apiGroups:
  - "" # "" indicates the core API group
```

```
  resources:
  - services
  - configmaps
  - pods
  - secrets
  - endpoints
  verbs:
  - get
  - watch
  - list
---

apiVersion: rbac.authorization.k8s.io/v1beta1
kind: RoleBinding
metadata:
  name: <helm-release>-ocnrf-rolebinding
  namespace: <namespace>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: <helm-release>-ocnrf-role
subjects:
- kind: ServiceAccount
  name:  <helm-release>-ocnrf-serviceaccount
  namespace: <namespace>
## Sample template end#
```

Configuring MySql database and user

This section explains how database administrator can create the OCNRF database and OCNRF application user.

> **Note:**
>
> 1. Procedure can be different for geo-redundant OCNRF sites and standalone OCNRF site.
> 2. For geo-redundant sites, before executing below procedure, assumption is geo-redundant DB-Tier sites are already up and replication channels enabled.

Procedure for Geo-Redundant OCNRF sites

1. Login to the machine which has permission to access the SQL nodes of NDB cluster.
2. Connect to the SQL node of NDB cluster successively. MySQL commands must be executed on all the SQL nodes.
3. Login to the MySQL prompt using root permission or user, which has permission to create users with conditions as mentioned below. For example: `mysql -h 127.0.0.1 -uroot -p`

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

4. Check OCNRF network function user already exists. If the user does not exists, create an OCNRF network function user by executing the following command:

```
# Check if user exists or not
$ SELECT User FROM mysql.user;

# In case, user already exists, move to next step.
# Command to create new user:-
$ CREATE USER '<OCNRF User Name>'@'%' IDENTIFIED BY '<OCNRF Password>';

# Example:-
$ CREATE USER 'nrfusr'@'%' IDENTIFIED BY 'nrfpasswd'
```

5. Check OCNRF network function database already exists. If the database does not exists, create an OCNRF network function database and provide permissions to OCNRF user name created in above step:
Execute the following command to check if database exists:-

```
$ show databases;
```

In case database already exists, then move to next step. Else, create database using the following command:

```
$ CREATE DATABASE IF NOT EXISTS <OCNRF Database> CHARACTER SET utf8;
```

Example:-

```
$ CREATE DATABASE IF NOT EXISTS nrfdb CHARACTER SET utf8;
```

Granting permission to user:-

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

Example:-

```
$ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE
TEMPORARY TABLES, DELETE, UPDATE, EXECUTE ON
nrfdb.* TO 'nrfusr'@'%';
```

6. Exit from MySQL prompt and SQL nodes.

> **Note:**
>
> Execute the commands on each SQL node on only one geo-redundant site. Other geo-redundant site(s) will get the data replicated automatically.

Procedure for standalone OCNRF site

1. Login to the machine which has permission to access the SQL nodes of NDB cluster.
2. Connect to the SQL node of NDB cluster successively. MySQL commands must be executed on all the SQL nodes.
3. Login to the MySQL prompt using root permission or user, which has permission to create users with conditions as mentioned below. For example: `mysql -h 127.0.0.1 -uroot -p`

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

4. Check OCNRF network function user already exists. If the NF does not exists, create an OCNRF network function user by executing the following command:

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

   In case the user already exists, move to next step. Else, Execute the following command to create new user:

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

   Example:-

   ```
   $ CREATE USER 'nrfusr'@'%' IDENTIFIED BY 'nrfpasswd';
   ```

5. Check OCNRF network function database already exists. If the database does not exists, create an OCNRF network function database and provide permissions to OCNRF user name created in above step:
   Execute the following command to check if database exists:-

   ```
   $ show databases;
   ```

   Check if required database is already in list. In case the database already exists, then move to next step. Else, create the database using the following command:

   ```
   $ CREATE DATABASE IF NOT EXISTS <OCNRF Database> CHARACTER SET utf8;
   ```

Example:-

```
$ CREATE DATABASE IF NOT EXISTS nrfdb CHARACTER SET utf8;
```

Granting permission to user:-

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

Example:-

```
$ GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE
TEMPORARY TABLES, DELETE, UPDATE, EXECUTE ON
nrfdb.* TO 'nrfusr'@'%';
```

6. Exit from MySQL prompt and SQL nodes.

> **Note:**
>
> Execute the commands on each SQL node of standalone site.

Configuring MySQL secret

MySQL secret creation

This section explains the steps for accessing MySql database and user details created by database administer in above section. This section must be execute before deploying OCNRF.

Following is the procedure to create and verify the secret:

1. Execute the following command to create kubernetes secret:

```
$ kubectl create secret generic <database secret name> --from-
literal=dbUsername=<OCNRF Mysql database username> --from-
literal=dbPassword=<OCNRF Mysql database passsword> --from-
literal=dbName=<OCNRF Mysql database name> -n <Namespace of MYSQL
secret>
```

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

Example

```
$ kubectl create secret generic database-secret --from-
literal=dbUsername=nrfusr --from-literal=dbPassword=nrfpasswd --from-
literal=dbName=nrfdb -n ocnrf
```

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

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

Example

```
 $ kubectl describe secret database-secret -n ocnrf
```

MySQL secret update

This section explains how to update the mysql secret with updated details.

1. Copy the exact command used in above section during creation of secret:

```
$ kubectl create secret generic <database secret name> --from-
literal=dbUsername=<OCNRF Mysql database username> --from-
literal=dbPassword=<OCNRF Mysql database passsword> --from-
literal=dbName=<OCNRF Mysql database name> -n <Namespace of MYSQL
secret>
```

2. Update the same command with string "--dry-run -o yaml" and "kubectl replace -f - -n <Namespace of MYSQL secret>".
3. Create secret command will look like:

```
$ kubectl create secret generic <database secret name> --from-
literal=dbUsername=<OCNRF Mysql database username> --from-
literal=dbPassword=<OCNRF Mysql database passsword> --from-
literal=dbName=<OCNRF Mysql database name> --dry-run -o yaml -n
<Namespace of MYSQL secret> | kubectl replace -f - -n <Namespace of
MYSQL secret>
```

4. Execute the updated command.
5. After successful secret update, the following message is displayed:

```
secret/<database secret name> replaced
```

Configuring secrets for enabling HTTPS

Creation of secrets for enabling HTTPS in OCNRF Ingress gateway

This section explains the steps to create secret for HTTPS related details. This section must be executed before enabling HTTPS in OCNRF Ingress gateway.

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

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

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

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

1. Execute the following command to create secret:

```
$ kubectl create secret generic <ocingress-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 OCNRF Ingress Gateway
secret>
```

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

Example: The names used below are same as provided in custom values.yaml in OCNRF deployment.

```
$ kubectl create secret generic ocingress-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 ocnrf
```

2. Verify the secret created using the following command:

   ```
   $ kubectl describe secret <ocingress-secret-name> -n <Namespace of
   OCNRF Ingress Gateway secret>
   ```

   Example:

   ```
   $ kubectl describe secret ocingress-secret -n ocnrf
   ```

Update the secrets for enabling HTTPS in OCNRF Ingress gateway

This section explains how to update the secret with updated details.

1. Copy the exact command used in above section during creation of secret.
2. Update the same command with string "--dry-run -o yaml" and "kubectl replace -f - -n <Namespace of OCNRF Ingress Gateway secret>".
3. Create secret command will look like:

   ```
   $ kubectl create secret generic <ocingress-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
   OCNRF Ingress Gateway secret> | kubectl replace -f - -n <Namespace of
   OCNRF Ingress Gateway secret>
   ```

   Example:-

   The names used below are same as provided in custom_values.yaml in OCNRF deployment:

   ```
   $ kubectl create secret generic ocingress-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 ocnrf | kubectl replace -f - -n ocnrf
   ```

4. Execute the updated command.
5. After successful secret update, the following message is displayed:

   ```
   secret/<ocingress-secret> replaced
   ```

Creation of secrets for enabling HTTPS in OCNRF Egress gateway

This section explains the steps to create secret for HTTPS related details. This section must be executed before enabling HTTPS in OCNRF Egress gateway.

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

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

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

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

1. Execute the following command to create secret.

```
$ kubectl create secret generic <ocegress-secret-name> --
fromfile=<ssl_ecdsa_private_key.pem>  --from-
file=<ssl_rsa_private_key.pem> --fromfile=<ssl_truststore.txt> --from-
file=<ssl_keystore.txt> --from-file=<ssl_cabundle.crt> --
fromfile=<ssl_rsa_certificate.crt> --from-
file=<ssl_ecdsa_certificate.crt> -n <Namespace of OCNRF Egress Gateway
secret>
```

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

Example: The names used below are same as provided in custom_values.yaml in OCNRF deployment.

```
$ kubectl create secret generic ocegress-secret --
fromfile=ssl_ecdsa_private_key.pem --from-file=ssl_rsa_private_key.pem
--fromfile=ssl_truststore.txt --from-file=ssl_keystore.txt --from-
file=ssl_cabundle.crt --fromfile=ssl_rsa_certificate.crt --from-
file=ssl_ecdsa_certificate.crt -n ocnrf
```

2. Command to verify secret created:

```
$ kubectl describe secret <ocegress-secret-name> -n <Namespace of OCNRF
Egress Gateway secret>
```

Example:

```
$ kubectl describe secret ocegress-secret -n ocnrf
```

Update the secrets for enabling HTTPS in OCNRF Egress gateway

This section explains how to update the secret with updated details.

1. Copy the exact command used in above section during creation of secret:
2. Update the same command with string "--dry-run -o yaml" and "kubectl replace -f - -n <Namespace of OCNRF Egress Gateway secret>".
3. Create secret command will look like:

```
kubectl create secret generic <ocegress-secret-name> --
fromfile=<ssl_ecdsa_private_key.pem> --from-
file=<ssl_rsa_private_key.pem> --fromfile=<ssl_truststore.txt> --from-
file=<ssl_keystore.txt> --from-file=<ssl_cabundle.crt> --
fromfile=<ssl_rsa_certificate.crt> --from-
file=<ssl_ecdsa_certificate.crt> --dry-run -o yaml -n <Namespace of
OCNRF Egress Gateway secret> | kubectl replace -f - -n <Namespace of
OCNRF Egress Gateway secret>
```

Example:-

The names used below are same as provided in custom_values.yaml in OCNRF deployment:

```
$ kubectl create secret generic egress-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 ocnrf | kubectl replace -f - -n ocnrf
```

4. Execute the updated command.
5. After successful secret update, the following message is displayed:

```
secret/<ocegress-secret> replaced
```

Configuring secret for enabling AccessToken service

Access Token secret creation

This section explains the steps to create secret for AccessToken service of OCNRF. This section must be executed before enabling Access Token in OCNRF.

> **Note:**
>
> The passwords for KeyStore is stored in respective password file below.

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

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

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

1. Execute the following command to create secret. The names used below are same as provided in custom values.yaml in OCNRF deployment:

   ```
   $ kubectl create secret generic <ocnrfaccesstoken-secret-name> --
   fromfile=<ecdsa_private_key.pem> --from-file=<rsa_private_key.pem> --
   fromfile=<ssl_truststore.txt> --from-file=<keystore_password.txt> --
   fromfile=rsa_certificate.crt --from-file=<ecdsa_certificate.crt> -n
   <Namespace of OCNRF AccessToken secret>
   ```

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

   Example:

   ```
   $ kubectl create secret generic ocnrfaccesstoken-secret --
   fromfile=ecdsa_private_key.pem --from-file=rsa_private_key.pem --
   fromfile=ssl_truststore.txt --from-file=keystore_password.txt --
   fromfile=rsa_certificate.crt --from-file=ecdsa_certificate.crt -n ocnrf
   ```

2. Execute the following command to verify secret created:

   ```
   $ kubectl describe secret <ocnrfaccesstoken-secret-name> -n <Namespace
   of OCNRF AccessToken secret>
   ```

   Example:

   ```
    $ kubectl describe secret ocnrfaccesstoken-secret -n ocnrf
   ```

Access Token secret update

This section explains how to update the access token secret with updated details.

1. Copy the exact command used in above section during creation of secret.
2. Update the same command with string "--dry-run -o yaml" and "kubectl replace -f - -n <Namespace of Access Token secret>".
3. Create secret command will look like:

   ```
   kubectl create secret generic <ocnrfaccesstoken-secret> --
   fromfile=<ecdsa_private_key.pem> --from-file=<rsa_private_key.pem> --
   fromfile=<ssl_truststore.txt> --from-file=<keystore_password.txt> --
   fromfile=<rsa_certificate.crt> --from-file=<ecdsa_certificate.crt> --
   dry-run -o yaml -n <Namespace of Access Token secret> | kubectl replace
   -f - -n <Namespace of Access Token secret>
   ```

   Example:-

   The names used below are same as provided in custom_values.yaml in OCNRF deployment:

   ```
   $ kubectl create secret generic ocnrfaccesstoken-secret --
   fromfile=ecdsa_private_key.pem --from-file=rsa_private_key.pem --
   fromfile=ssl_truststore.txt --from-file=keystore_password.txt --
   fromfile=rsa_certificate.crt --from-file=ecdsa_certificate.crt --dry-
   run -o yaml -n ocnrf | kubectl replace -f - -n ocnrf
   ```

4. Execute the updated command.
5. After successful secret update, the following message is displayed:

   ```
   secret/<ocnrfaccesstoken-secret> replaced
   ```

Installation Tasks

This section describes the tasks that the user needs to follow for installing OCNRF.

Downloading OCNRF package

Following is the porcedure to download the release package from MOS:

1. Login to MOS using the appropriate login credentials.
2. Select **Product & Updates** tab.
3. In **Patch Search** console select **Product or Family (Advanced)** tab.
4. Enter *Oracle Communications Cloud Native Core - 5G* in **Product** field and select the product from the Product drop-down.
5. Select *Oracle Communications Cloud Native Core Network Repository Function <release_number>* in **Release** field.
6. Click **Search**. The **Patch Advanced Search Results** list appears.
7. Select the required patch from the list. The Patch Details window appears.

8. Click on **Download**. File Download window appears.
9. Click on the **<p********_<release_number>_Tekelec>.zip** file.
10. Click on the zip file to download the network function patch to the system where network function must be installed.

Install OCNRF

1. Unzip the release package file to the system where you want to install the network function. You can find the OCNRF package as follows:
   `ReleaseName-pkg-Releasenumber.tgz`

   where:
   ReleaseName is a name which is used to track this installation instance.

   Releasenumber is the release number.

   For example, ocnrf-pkg-1.6.1.0.0.tgz
2. Untar the OCNRF package file to get OCNRF docker image tar file:

   ```
   tar -xvzf ReleaseName-pkg-Releasenumber.tgz
   ```
3. Load the *ocnrf-images-<release_number>.tar* file into the Docker system:

   ```
   docker load --input /IMAGE_PATH/ocnrf-images-<release_number>.tar
   ```
4. Verify that the image is loaded correctly by entering this command:

   ```
   docker images
   ```
5. Execute the following commands to push the docker images to docker registry:

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


   docker push <docker-repo>/<image-name>:<image-tag>
   ```
6. Untar the helm files:

   ```
   tar -xvzf ocnrf-<release_number>.tgz
   ```
7. Create the customize ocnrf-custom-values-1.6.1.yaml file with the required input parameters. To customize the file, refer to Customizing OCNRF chapter.
8. Go to the extracted OCNRF package as explained in:

   ```
   cd ocnrf-<release_number>
   ```
9. Install OCNRF by executing the following command:

   ```
   helm install ocnrf/ --name <helm-release> --namespace <k8s namespace> -f <ocnrf_customized_values.yaml>
   ```

Example: `helm install ocnrf/ --name ocnrf --namespace ocnrf -f ocnrf-custom-values-1.6.1.yaml`

10. Execute the following command to check the status:
    `helm status <helm-release>`

    For example: `helm status ocnrf`

11. Execute the following command to check status of the services:
    `kubectl -n <k8s namespace> get services`

    For example:
    `kubectl -n ocnrf get services`

    Note: If `metallb` is used, `EXTERNAL-IP` is assigned to `<helm release name>-endpoint`. `ocnrf` is the helm release name.

```
NAME                    TYPE          CLUSTER-IP     EXTERNAL-IP
PORT(S)            AGE
ocnrf-egressgateway     ClusterIP     10.233.1.61    <none>       8080/
TCP,5701/TCP  30h
ocnrf-ingressgateway    LoadBalancer  10.233.52.194  <pending>
80:31776/TCP       30h
ocnrf-nfaccesstoken     ClusterIP     10.233.53.115  <none>       8080/
TCP           30h
ocnrf-nfdiscovery       ClusterIP     10.233.21.28   <none>       8080/
TCP           30h
ocnrf-nfregistration    ClusterIP     10.233.4.140   <none>       8080/
TCP           30h
ocnrf-nfsubscription    ClusterIP     10.233.44.98   <none>       8080/
TCP           30h
ocnrf-nrfauditor        ClusterIP     10.233.1.71    <none>       8080/
TCP           30h
ocnrf-nrfconfiguration LoadBalancer  10.233.40.230  <pending>
8080:30076/TCP     30h
```

12. Execute the following command to check status of the pods:
    `kubectl get pods -n <k8s namespace>`

    Status column of all the pods should be 'Running'.

    Ready column of all the pods should be n/n, where n is number of containers in the pod.

    For example:

    `kubectl get pods -n ocnrf`

```
NAME                                    READY   STATUS    RESTARTS   AGE
ocnrf-egressgateway-d6567bbdb-9jrsx     2/2     Running   0          30h
ocnrf-egressgateway-d6567bbdb-ntn2v     2/2     Running   0          30h
ocnrf-ingressgateway-754d645984-h9vzq   2/2     Running   0          30h
ocnrf-ingressgateway-754d645984-njz4w   2/2     Running   0          30h
ocnrf-nfaccesstoken-59fb96494c-k8w9p    2/2     Running   0          30h
ocnrf-nfaccesstoken-49fb96494c-k8w9q    2/2     Running   0          30h
ocnrf-nfdiscovery-84965d4fb9-rjxg2      1/1     Running   0          30h
ocnrf-nfdiscovery-94965d4fb9-rjxg3      1/1     Running   0          30h
ocnrf-nfregistration-64f4d8f5d5-6q92j   1/1     Running   0          30h
ocnrf-nfregistration-44f4d8f5d5-6q92i   1/1     Running   0          30h
```

```
ocnrf-nfsubscription-5b6db965b9-gcvpf    1/1    Running  0          30h
ocnrf-nfsubscription-4b6db965b9-gcvpe    1/1    Running  0          30h
ocnrf-nrfauditor-67b676dd87-xktbm        1/1    Running  0          30h
ocnrf-nrfconfiguration-678fddc5f5-c5htj  1/1    Running  0          30h
```

3
Customizing OCNRF

This section includes information about OCNRF customization.

- OCNRF Configuration
- OCNRF Configurable Parameters

OCNRF Configuration

This section describes about the OCNRF customization.

The OCNRF deployment is customized by overriding the default values of various configurable parameters.

Follow the below steps to customize the `ocnrf-custom-values-1.6.1.yaml` file as per the required parameters:

1. Go to the Oracle Help Center (OHC) Web site.
2. Navigate to **Industries->Communications->Cloud Native Core->Release 2.2.0**.
3. Click the **NRF Custom Template** link to download the zip file.
4. Unzip the file to get ocnrf-custom-configTemplates-1.6.1.0.0 file that contains the ocnrf-custom-configTemplates-1.6.1.0.0. This file is used during installation.
   - `ocnrf-custom-values-1.6.1.yaml`: This file is used during installation.
   - `NrfDashboard-1.6.1.json`: This file is used by `grafana`.
   - `NrfAlertrules-1.6.1.yaml`: This file is used for `prometheus`.
   - `OCNRF-MIB-TC-1.6.1.mib`: This is considered as OCNRF top level mib file, where the Objects and their data types are defined .
   - `OCNRF-MIB-1.6.1.mib` : This file fetches the Objects from the top level mib file and based on the Alert notification, these objects can be selected for display.
5. Customize the ocnrf-custom-values-1.6.1.yaml file.
6. Save the updated ocnrf-custom-values-1.6.1.yaml file in the helm chart directory.

> **Note:**
>
> Refer section OCNRF Configuration Parameters to know more about the configurable parameters.

OCNRF Images

Following are the OCNRF images:

Table 3-1 OCNRF Images

Services	Image	Tag
<helm-release-name>-NFRegistration	ocnrf-nfregistration	1.6.1
<helm-release-name>-NFSubscription	ocnrf-nfsubscription	1.6.1
<helm-release-name>-NFDiscovery	ocnrf-nfdiscovery	1.6.1
<helm-release-name>-NRF Auditor	ocnrf-nrfauditor	1.6.1
<helm-release-name>-NRF Configuration	ocnrf-nrfconfiguration	1.6.1
<helm-release-name>-NFAccessToken	configurationinit	1.1.1
	configurationupdate	1.1.1
	ocnrf-nfaccesstoken	1.6.1
<helm-release-name>-EgressGateway	configurationinit	1.1.1
	configurationupdate	1.1.1
	ocegress_gateway	1.6.4
<helm-release-name>-IngressGateway	configurationinit	1.1.1
	configurationupdate	1.1.1
	ocingress_gateway	1.6.4

Note:

IngressGateway, EgressGateway and NFAccessToken uses same configurationinit and configurationupdates docker images.

OCNRF Configuration Parameters

This section includes information about the configuration parameters of OCNRF.

OCNRF allows customization of parameters for the following services and related settings.

Global Parameters

Table 3-2 Global Parameters

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`mysql.primary.host`	Primary DB Connection Service IP or Hostname	ocnrf-mysql	M	Primary DB Connection Service HostName or IP	OCNRF connects to Primary DB Connection Service if not available then it connects to Secondary DB Connection Service. For NDB Cluster, use Host/IP of the DB Connection Service.
`mysql.primary.port`	Primary DB Connection Service	3306	M	Primary DB Connection Service Port	Port that is used while connecting to Primary DB Connection Service.
`mysql.secondary.host`	Secondary DB Connection Service IP or Hostname	ocnrf-mysql	O	Secondary DB Connection Service HostName or IP	OCNRF connects to Secondary DB Connection Service only if the Primary DB Connection Service is unavailable. It again switch pack to Primary DB Connection Service one it is available. For NDB Cluster, use Host/IP of the Remote DB Connection Service (if available).
`mysql.secondary.port`	Secondary DB Connection Service Port	3306	O	Secondary DB Connection Service Port	Port that is used while connecting to Secondary DB Connection Service.

Table 3-2 (Cont.) Global Parameters

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`endpoint`	OCNRF END Point Name	ocnrf-ingress-gateway.ocnrf.svc.cluster.local	M	Service Name for OCNRF ingress gateway	OCNRF Ingress Gateway's Name and Port. This value is used in UriList of NfListRetrival Service Operation response. The endpoint needs to be OCNRF's External Routable FQDN (e.g. ocnrf.oracle.com) OR External Routable IpAddress (e.g. 10.75.212.60) OR for routing with in the same K8 cluster use full OCNRF ingress gateway Service FQDN as below format # <helm-release-name>-ingress-gateway.<namespace>.svc.<cluster-domain-name> e.g ocnrf-ingress-gateway.nrf-1.svc.cluster.local where "ocnrf": is the helm release name (deployment name that will be used during "helm install") "nrf-1": is the namespace in which OCNRF is deployed "cluster.local": is the K8's dnsDomain name (dnsDomain can be found using "kubectl -n kube-system get configmap kubeadm-config -o yaml | grep -i dnsDomain") **Note**: This value must be changed during deployment based on the configuration.
`endpointPort`	OCNRF END Point Port	80	M	Port for OCNRF ingress gateway	This parameter is used as OCNRF end point port.

Table 3-2 (Cont.) Global Parameters

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`nrfInstanceId`	OCNRF's NF Instance ID	6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c	M		OCNRF's NfInstance Id (UUID format)
`dockerRegistry`	Registry for docker	ocnrf-registry.us.oracle.com:5000	M		Docker Registry's FQDN/ Port where OCNRF's docker images are available.
`database.nameSpace`	Namespace for database connection	ocnrf	M		The Namespace where the Kubernetes Secret is created which contains MYSQL details. **Note**: See database.name configuration for more details.
`database.name`	Secret name for database connection	database-secret	M		The Kubernetes Secret which contains the Database name, Database User name and the Password. **Note**: Refer OCNRF Pre-requisites section for the file format.

Table 3-2 (Cont.) Global Parameters

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`serviceAccountName`	ServiceAccount which is having permission for get, watch and list operation for below kubernetes resources; services, configmaps, pods, secrets and endpoints		M		This SeviceAccount is used for: • fetching MYSQL DB Details from configured kubernetes secret • fetching OCNRF's Private Key, OCNRF's Certificate and CA Certificate from configured kubernetes secret • fetching OCNRF's Private and OCNRF's Public Keys for Digitally Signing AccessTokenClaims. • fetching Producer/ Consumer NF's Service/Endpoint details for routing messages from/to Egress/Ingress Gateways. Refer to prerequisites for command details.

Ingress Gateway Global Parameters

Table 3-3 Ingress Gateway Global Parameters

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`metalLbIpAllocationEnabled`	Enable or disable IP Address allocation from Metallb Pool	false	O	true/false	
`metalLbIpAllocationAnnotation`	Address Pool Annotation for Metallb	metallb.universe.tf/address-pool: signaling	M when `metalLbIpAllocationEnabled` is true		

Table 3-3 (Cont.) Ingress Gateway Global Parameters

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
staticIpAddressEnabled	Static load balancer IP enabled flag	false	O	true/false	
staticIpAddress	Static IP address assigned to the Load Balancer from the metalLB IP pool.	10.75.212.50	M, when staticIpAddressEnabledis true		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.
staticNodePortEnabled	Static Node Port enabled flag	false	O	true/false	If Static node port needs to be set, then set staticNodePortEnabled flag to true and provide value for staticHttpNodePort or staticHttpsNodePort. Else random node port will be assigned by K8.
staticHttpNodePort	HTTP node port	30080	M, when staticNodePortEnabled is true and ingress-gateway.enableIncomingHttp is true		

Table 3-3 (Cont.) Ingress Gateway Global Parameters

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
staticHttpsNodePort	HTTPs node port	30443	M, when staticNodePortEnabled is true and ingress-gateway.enableIncomingHttps is true		
publicHttpSignalingPort	Service Port on which OCNRF's Ingress Gateway is exposed	80	O		If enableIncomingHttp is true, publicHttpSignalingPort will be used as HTTP/2.0 Port (unsecured)
publicHttpsSignallingPort	Service Port on which OCNRF's Ingress Gateway is exposed	443	O		If enableIncomingHttps is true, publicHttpsSignallingPort Port will be used as HTTPS/2.0 Port (secured TLS)

Ingress Gateway

Table 3-4 Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
ingress-gateway.enableIncomingHttp	This flag is for enabling/ disabling HTTP/2.0 (insecure) in Ingress Gateway.	true	O	true/false	If the value is set to false, OCNRF will not accept any HTTP/2.0 (unsecured) Traffic. If the value is set to true, OCNRF will accept HTTP/2.0 (unsecured) Traffic

Table 3-4 (Cont.) Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`ingress-gateway.enableIncomingHttps`	This flag is for enabling/ disabling HTTPS/2.0 (secure) in Ingress Gateway.	false	O	true/false	If the value is set to false, OCNRF will not accept any HTTPS/2.0 (unsecured) Traffic. If the value is set to true, OCNRF will accept HTTPS/2.0 (unsecured) Traffic
`ingress-gateway.image.name`	Ingress Gateway image name.	ocingress_gateway	O		
`ingress-gateway.image.tag`	Tag name of Ingress Gateway image	1.6.4	O		
`ingress-gateway.image.pullPolicy`	This setting will tell if image need to be pulled or not	IfNotPresent	O	Always, IfNotPresent, Never	
`ingress-gateway.initContainersImage.name`	Image Name for Ingress Gateway init container	configurationinit	O		
`ingress-gateway.initContainersImage.tag`	Tag name of Ingress Gateway init container	1.1.1	O		
`ingress-gateway.initContainersImage.pullPolicy`	This setting will tell if image need to be pulled or not	IfNotPresent	O	Always, IfNotPresent, Never	
`ingress-gateway.updateContainersImage.name`	Image Name for Ingress Gateway update container	configurationupdate	O		
`ingress-gateway.updateContainersImage.tag`	Tag name of Ingress Gateway update container	1.1.1	O		

Table 3-4 (Cont.) Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`ingress-gateway.updateContainersImage.pullPolicy`	This setting will tell if image need to be pulled or not	IfNotPresent	O	Always, IfNotPresent, Never	
`ingress-gateway.jaegerTracingEnabled`	Flag to enable or disable the Jaeger Tracing at ingress-gateway	false	O	true / false	While making this flag as true, update the below attributes with correct values.
`ingress-gateway.opentracing.jaeger.udpsender.host`	Host name of Jaeger Agent Service	jaeger-agent.cne-infra	M, if `ingress-gateway.jaegerTracingEnabled` is true		
`ingress-gateway.opentracing.jaeger.udpsender.port`	Port of Jaeger Agent Service	6831	M, if `ingress-gateway.jaegerTracingEnabled` is true		
`ingress-gateway.opentracing.jaeger.probabilisticSampler`	Jaeger message sampler	0.5	O	0 to 1	# Jaeger message sampler. Value range: 0 to 1 # e.g. Value 0: No Trace will be sent to Jaeger collector # e.g. Value 0.3: 30% of message will be sampled and will be sent to Jaeger collector # e.g. Value 1: 100% of message (i.e. all the messages) will be sampled and will be sent to Jaeger collector

Table 3-4 (Cont.) Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`ingress-gateway.cipherSuites`	Allowed CipherSuites for TLS1.2		M, if `ingress-gateway.enableIncomingHttps` is true	- 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	
`ingress-gateway.service.ssl.privateKey.k8SecretName`	Secret name that contains OCNRF Ingress gateway Private Key	ocingress-secret	M, if `ingress-gateway.enableIncomingHttps` is true		

Table 3-4 (Cont.) Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`ingress-gateway.service.ssl.privateKey.k8NameSpace`	Namespace in which k8SecretName is present	ocnrf	M, if `ingress-gateway.enableIncomingHttps` is true		
`ingress-gateway.service.ssl.privateKey.rsa.filename`	OCNRF's Private Key (RSA type) file name	rsa_private_key_pkcs1.pem	M, if `ingress-gateway.enableIncomingHttps` is true and `ingress-gateway.service.ssl.initialAlgorithm` is RSA256		If initialAlgorithm is configured as RSA, then rsa file name must be configured. Otherwise OCNRF's ingress gateway will not comeup.

Table 3-4 (Cont.) Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`ingress-gateway.service.ssl.privateKey.ecdsa.filename`	OCNRF's Private Key (ECDSA type) file name	ssl_ecdsa_private_key.pem	M, if `ingress-gateway.enableIncomingHttps` is true and `ingress-gateway.service.ssl.initialAlgorithm` is ES256		If initialAlgorithm is configured as ECDSA, then rsa file name must be configured. Otherwise OCNRF's ingress gateway will not comeup.
`ingress-gateway.service.ssl.certificate.k8SecretName`	Secret name that contains OCNRF's Certificate for HTTPS	ocingress-secret	M, if `ingress-gateway.enableIncomingHttps` is true		This is a Secret object for OCNRFcertificate details for HTTPS.
`ingress-gateway.service.ssl.certificate.k8NameSpace`	Namespace in which OCNRF's Certificate is present	ocnrf	M, if `ingress-gateway.enableIncomingHttps` is true		

Table 3-4 (Cont.) Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`ingress-gateway.service.ssl.certificate.rsa.filename`	OCNRF's Certificate (RSA type) file name	ssl_rsa_certificate.crt	M, if `ingress-gateway.enableIncomingHttps` is true and `ingress-gateway.service.ssl.initialAlgorithm` is RSA256		If initialAlgorithm is configured as RSA, then rsa file name must be configured. Otherwise OCNRF's ingress gateway will not comeup.
`ingress-gateway.service.ssl.certificate.ecdsa.filename`	OCNRF's Certificate (ECDSA type) file name	ssl_ecdsa_certificate.crt	M, if `ingress-gateway.enableIncomingHttps` is true and `ingress-gateway.service.ssl.initialAlgorithm` is ES256		If initialAlgorithm is configured as ECDSA, then rsa file name must be configured. Otherwise OCNRF's ingress gateway will not comeup.

Table 3-4 (Cont.) Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`ingress-gateway.service.ssl.caBundle.k8SecretName`	Secret name that contains OCNRF's CA details for HTTPS	ocingress-secret	M, if `ingress-gateway.enableIncomingHttps` is true		
`ingress-gateway.service.ssl.caBundle.k8NameSpace`	Namespace in which OCNRF's CA details is present	ocnrf	M, if `ingress-gateway.enableIncomingHttps` is true		
`ingress-gateway.service.ssl.caBundle.filename`	OCNRF's CA bundle filename	caroot.cer	M, if `ingress-gateway.enableIncomingHttps` is true		
`ingress-gateway.service.ssl.keyStorePassword.k8SecretName`	Secret name that contains keyStorePassword	ocingress-secret	M, if `ingress-gateway.enableIncomingHttps` is true		
`ingress-gateway.service.ssl.keyStorePassword.k8NameSpace`	Namespace in which OCNRF's keystore password is present	ocnrf	M, if `ingress-gateway.enableIncomingHttps` is true		

Table 3-4 (Cont.) Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`ingress-gateway.service.ssl.keyStorePassword.fileName`	OCNRF's Key Store password Filename	ssl_keystore.txt	M, if `ingress-gateway.enableIncomingHttps` is true		
`ingress-gateway.service.ssl.trustStorePassword.k8SecretName`	Secret name that contains trustStorePassword	ocingress-secret	M, if `ingress-gateway.enableIncomingHttps` is true		
`ingress-gateway.service.ssl.trustStorePassword.k8NameSpace`	Namespace in which trustStorePassword is present	ocnrf	M, if `ingress-gateway.enableIncomingHttps` is true		
`ingress-gateway.service.ssl.trustStorePassword.fileName`	OCNRF's trustStorePassword Filename	ssl_truststore.txt	M, if `ingress-gateway.enableIncomingHttps` is true		
`ingress-gateway.service.ssl.initialAlgorithm`	Initial Algorithm for HTTPS	ES256	O	ES256, RSA256	Algorithm that will be used in TLS handshake

Table 3-4 (Cont.) Ingress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
ingress-gateway.service.log.level.root	setting logging level	WARN	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE	
ingress-gateway.service.log.level.ingress	setting logging level	INFO	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE	
ingress-gateway.service.log.level.oauth	setting logging level	INFO	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE	

Egress Gateway

Table 3-5 Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
egress-gateway.enableOutgoingHttps	This flag is for enabling/ disabling HTTPS/2.0 (secured TLS) in Egress Gateway.	false	O	true/false	If the value is set to false, OCNRF will not accept any HTTPS/2.0 (unsecured) Traffic. If the value is set to true, OCNRF will accept HTTPS/2.0 (unsecured) Traffic
egress-gateway.deploymentegressgateway.image	Egress Gateway image name	ocegress_gateway	O		

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
egress-gateway.deploymentegressgateway.imageTag	tag name of image	1.6.4	O		
egress-gateway.deploymentegressgateway.pullPolicy	This setting will tell if image need to be pulled or not	IfNotPresent	O	Always, IfNotPresent, Never	
egress-gateway.initContainersImage.name	Image Name for Egress Gateway init container	configurationinit	O		
egress-gateway.initContainersImage.tag	Tag name of Egress Gateway init container	1.1.1	O		
egress-gateway.initContainersImage.pullPolicy	This setting will tell if image need to be pulled or not	IfNotPresent	O	Always, IfNotPresent, Never	
egress-gateway.updateContainersImage.name	Image Name for Egress Gateway update container	configurationupdate	O		
egress-gateway.updateContainersImage.tag	Tag name of Egress Gateway update container	1.1.1	O		
egress-gateway.updateContainersImage.pullPolicy	This setting will tell if image need to be pulled or not	IfNotPresent	O	Always, IfNotPresent, Never	
egress-gateway.jaegerTracingEnabled	Flag to enable or disable the Jaeger Tracing at egress gateway	false	O	true / false	While making this flag as true, update the below attributes with correct values.

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`egress-gateway.opentracing.jaeger.udpsender.host`	Host name of Jaeger Agent Service	jaeger-agent.cne-infra	M, if `egress-gateway.jaegerTracingEnabled` is enabled		
`egress-gateway.opentracing.jaeger.udpsender.port`	Port of Jaeger Agent Service	6831	M, if `egress-gateway.jaegerTracingEnabled` is enabled		
`egress-gateway.opentracing.jaeger.probabilisticSampler`	Jaeger message sampler	0.5	O	0 to 1	# Jaeger message sampler. Value range: 0 to 1 # e.g. Value 0: No Trace will be sent to Jaeger collector # e.g. Value 0.3: 30% of message will be sampled and will be sent to Jaeger collector # e.g. Value 1: 100% of message (i.e. all the messages) will be sampled and will be sent to Jaeger collector
`egress-gateway.scpIntegrationEnabled`	Using SCP as an Proxy in Egress Gateway	false	O	true/false	If it is configured as false, SCP will not be used as an proxy. Messages will be directly sent to the Producers/HTTP Servers. If it is configured as true, SCP will be used as an Proxy for delivering messages to the Producers/HTTP Servers.

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`egress-gateway.scpHttpHost`	SCP Configuration For Egress Gateway	localhost	M, if `egress-gateway.scpIntegrationEnabled` is true		All the SCP related configuration will be used only if scpIntegrationEnabled is set to true. SCP's HTTP Host/IP and Port Combination. This will be while sending HTTP/2.0 (unsecured) traffic.
`egress-gateway.scpHttpPort`	SCP's HTTP Port	80	M, if `egress-gateway.scpIntegrationEnabled` is true		
`egress-gateway.scpHttpsHost`	SCP Configuration For Egress Gateway	localhost	M, if `egress-gateway.scpIntegrationEnabled` is true		All the SCP related configuration will be used only if scpIntegrationEnabled is set to true. SCP's HTTP Host/IP and Port Combination. This will be while sending HTTP/2.0 (unsecured) traffic.
`egress-gateway.scpHttpsPort`	SCP's HTTPS Port	443	M, if `egress-gateway.scpIntegrationEnabled` is true		This will be while sending HTTPS/2.0 (unsecured) traffic.
`egress-gateway.scpApiPrefix`	SCP's API Prefix. (Applicable only for SCP with TLS enabled)	/	O		This will be used for constructing the Egress messgage's APIROOT while proxying message to SCP. Change this value to SCP's apiprefix. "/" is not expected to be provided along.

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`egress-gateway.scpDefaultScheme`	SCP's default scheme when 3gpp-sbi-target-apiroot header is missing	https	O		
`egress-gateway.cipherSuites`	Allowed CipherSuites for TLS1.2		M, if `egress-gateway.enableOutgoingHttps` is true	- 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	
`egress-gateway.service.ssl.privateKey.k8SecretName`	Secret name that contains OCNRF Egress gateway Private Key	ocegress-secret	M, if `egress-gateway.enableOutgoingHttps` is true		

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`egress-gateway.service.ssl.privateKey.k8NameSpace`	Namespace in which k8SecretName is present	ocnrf	M, if `egress-gateway.enableOutgoingHttps` is true		
`egress-gateway.service.ssl.privateKey.rsa.filename`	OCNRF's Private Key (RSA type) file name	ssl_rsa_private_key.pem	M, if `egress-gateway.enableOutgoingHttps` is true and `egress-gateway.service.ssl.initialAlgorithm` is RSA256		If initialAlgorithm is configured as RSA, then rsa file name must be configured. Otherwise OCNRF's egress gateway will not comeup.

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`egress-gateway.service.ssl.privateKey.ecdsa.filename`	OCNRF's Private Key (ECDSA type) file name	ssl_ecdsa_private_key.pem	M, if `egress-gateway.enableOutgoingHttps` is true and `egress-gateway.service.ssl.initialAlgorithm` is ES256		If initialAlgorithm is configured as ECDSA, then rsa file name must be configured. Otherwise OCNRF's egress gateway will not comeup.
`egress-gateway.service.ssl.certificate.k8SecretName`	Secret name that contains OCNRF's Certificate for HTTPS	ocegress-secret	M, if `egress-gateway.enableOutgoingHttps` is true		This is a Secret object for OCNRFcertificate details for HTTPS.
`egress-gateway.service.ssl.certificate.k8NameSpace`	Namespace in which OCNRF's Certificate is present	ocnrf	M, if `egress-gateway.enableOutgoingHttps` is true		

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`egress-gateway.service.ssl.certificate.rsa.filename`	OCNRF's Certificate (RSA type) file name	ssl_rsa_certificate.crt	M, if `egress-gateway.enableOutgoingHttps` is true and `egress-gateway.service.ssl.initialAlgorithm` is RSA256		If initialAlgorithm is configured as RSA, then rsa file name must be configured. Otherwise OCNRF's egress gateway will not comeup.
`egress-gateway.service.ssl.certificate.ecdsa.filename`	OCNRF's Certificate (ECDSA type) file name	ssl_ecdsa_certificate.crt	M, if `egress-gateway.enableOutgoingHttps` is true and `egress-gateway.service.ssl.initialAlgorithm` is ES256		If initialAlgorithm is configured as ECDSA, then rsa file name must be configured. Otherwise OCNRF's egress gateway will not comeup.

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`egress-gateway.service.ssl.caBundle.k8SecretName`	Secret name that contains OCNRF's CA details for HTTPS	ocegress-secret	M, if `egress-gateway.enableOutgoingHttps` is true		
`egress-gateway.service.ssl.caBundle.k8NameSpace`	Namespace in which OCNRF's CA details is present	ocnrf	M, if `egress-gateway.enableOutgoingHttps` is true		
`egress-gateway.service.ssl.caBundle.filename`	OCNRF's CA bundle filename	ssl_cabundle.crt	M, if `egress-gateway.enableOutgoingHttps` is true		
`egress-gateway.service.ssl.keyStorePassword.k8SecretName`	Secret name that contains keyStorePassword	ocegress-secret	M, if `egress-gateway.enableOutgoingHttps` is true		
`egress-gateway.service.ssl.keyStorePassword.k8NameSpace`	Namespace in which OCNRF's keystore password is present	ocnrf	M, if `egress-gateway.enableOutgoingHttps` is true		

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
egress-gateway.service.ssl.keyStorePassword.fileName	OCNRF's Key Store password Filename	ssl_keystore.txt	M, if egress-gateway.enableOutgoingHttps is true		
egress-gateway.service.ssl.trustStorePassword.k8SecretName	Secret name that contains trustStorePassword	ocegress-secret	M, if egress-gateway.enableOutgoingHttps is true		
egress-gateway.service.ssl.trustStorePassword.k8NameSpace	Namespace in which trustStorePassword is present	ocnrf	M, if egress-gateway.enableOutgoingHttps is true		
egress-gateway.service.ssl.trustStorePassword.fileName	OCNRF's trustStorePassword Filename	ssl_truststore.txt	M, if egress-gateway.enableOutgoingHttps is true		
egress-gateway.service.ssl.initialAlgorithm	Initial Algorithm for HTTPS	RSA256	O	ES256, RSA256	Algorithm that will be used in TLS handshake
egress-gateway.service.log.level.root	setting logging level	WARN	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE	

Table 3-5 (Cont.) Egress Gateway

Parameter	Description	Default value	Mandatory (M)/ Optional (O)	Range or Possible Values (If applicable)	Notes
`egress-gateway.service.log.level.egress`	setting logging level	INFO	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE	
`egress-gateway.service.log.level.oauth`	setting logging level	INFO	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE	

NF Registration Micro service (`nfregistration`)

Table 3-6 NF Registration

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
`nfregistration.image.registry`	Docker registry name	ocnrf	O	Registry name	
`nfregistration.image.name`	Full Image Path	ocnrf-nfregistration	O	Full image path of image	
`nfregistration.image.tag`	Tag of Image	1.6.1	O	Tag of image in docker repository	
`nfregistration.image.pullPolicy`	This setting will tell if image need to be pulled or not	IfNotPresent	O	Possible Values - `Always, IfNotPresent, Never`	
`nfregistration.log.level`	Logging level	WARN	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE	Logging level

NF Subscription Micro service (`nfsubscription`)

Table 3-7 NF Subscription

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)
`nfsubscription.image.registry`	Docker registry name	ocnrf	O	
`nfsubscription. image.name`	Full Image Path	ocnrf-nfsubscription	O	Full image path of image
`nfsubscription.image.tag`	Tag of Image	1.6.1	O	Tag of image in docker repository
`nfsubscription.image.pullPolicy`	This setting will tell if image need to be pulled or not	IfNotPresent	O	Possible Values: `Always, IfNotPresent, Never`
`nfsubscription.log.level`	Logging level	WARN	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE

OCNRF Auditor Micro service (`nrfauditor`)

Table 3-8 OCNRF Auditor

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)
`nrfauditor.image.registry`	Docker registry name	ocnrf	O	
`nrfauditor.image.name`	Full Image Path	ocnrf-nrfauditor	O	Full image path of image
`nrfauditor.image.tag`	Tag of Image	1.6.1	O	Tag of image in docker repository
`nrfauditor.image.pullPolicy`	This setting indicates if the image needs to be pulled or not	IfNotPresent	O	Possible Values: `Always, IfNotPresent, Never`
`nrfauditor.log.level`	Logging level	WARN	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE

NF Discovery Micro service (`nfdiscovery`)

Table 3-9 NF Discovery

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)
`nfdiscovery.image.registry`	Docker registry name	ocnrf	O	Registry name
`nfdiscovery.image.name`	Full Image Path	ocnrf-nfdiscovery	O	Full image path of image
`nfdiscovery.image.tag`	Tag of Image	1.6.1	O	Tag of image in docker repository
`nfdiscovery.image.pullPolicy`	This setting determines if image needs to be pulled or not	IfNotPresent	O	Possible Values: `Always`, `IfNotPresent`, `Never`
`nfdiscovery.log.level`	Logging level	WARN	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE

OCNRF Configuration

Table 3-10 OCNRF Configuration

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
`image.registry`	Docker registry name	ocnrf	O	Registry name	
`image.name`	Full Image Path	nrfconfiguration	O	Full image path of image	
`image.tag`	Tag of Image	1.6.1	O	Tag of image in docker repository	
`image.pullPolicy`	This setting determines if image needs to be pulled or not	IfNotPresent	O	Possible Values: `Always`, `IfNotPresent`, `Never`	
`log.level`	Logging level	WARN	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE	

Table 3-10 (Cont.) OCNRF Configuration

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
`service.metalLbIpAllocationEnabled`	Enable or disable IP Address allocation from Metallb Pool	false	O	As defined by operator	If this flag is enabled, the IP Address is allocated from Metallb Pool.
`service.metalLbIpAllocationAnnotation`	Address Pool for Metallb	metallb.universe.tf/address-pool : oam	M, if `nrfconfiguration.service.metalLbIpAllocationEnabled` is true		Address Pool Annotation for Metallb
`service.staticIpAddressEnabled`	Static load balancer IP enabled flag	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
`service.staticIpAddress`	Static load balancer IP	10.75.212.50	M, if `nrfconfiguration.service.metalLbIpAllocationEnabled` is true		Static IP address assigned to the Load Balancer from the metalLB IP pool.
`service.staticNodePortEnabled`	Static Node Port enabled flag	false	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 K8

Table 3-10 (Cont.) OCNRF Configuration

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
service.staticNodePort	Static Node Port	30076	M, if nrfconfiguration.service.staticIpAddressEnabled is enabled.		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

NF Access Token(nfaccesstoken)

Table 3-11 NF Access Token

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
nfaccesstoken.enabled	Flag to disable Oauth functionality	true	O	true / false	If AccessToken service is not required, operator can choose to set it as false so that nfAccessToken micro-service will not be deployed.
nfaccesstoken.image.name	Full Image Path for access token service container	ocnrf-nfaccesstoken	O	Full image path of image	
nfaccesstoken.image.tag	Tag of Image	1.6.1	O	Tag of image in docker repository	
nfaccesstoken.image.pullPolicy	This setting will tell if image need to be pulled or not	IfNotPresent	O	Possible Values - Always IfNotPresent Never	
nfaccesstoken.initContainersImage.name	Full Image Path for init container	configurationinit	O	Image Name for Access token Key certificate infrastructure	This image is used by OCNRF gateway for Key/ Certificate infrastructure.

Table 3-11 (Cont.) NF Access Token

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
nfaccesstoken.initContainersImage.tag	Tag of Image	1.1.1	O	Tag of image in docker repository	
nfaccesstoken.initContainersImage.pullPolicy	This setting will tell if image need to be pulled or not	IfNotPresent	O	Possible Values - Always IfNotPresent Never	
nfaccesstoken.updateContainersImage.name	Full Image Path for update container	configurationupdate	O	Image Name for Access token Key certificate infrastructure	
nfaccesstoken.updateContainersImage.tag	Tag of Image	1.1.1	O	Tag of image in docker repository	
nfaccesstoken.updateContainersImage.pullPolicy	This setting will tell if image need to be pulled or not	IfNotPresent	O	Possible Values - Always IfNotPresent Never	

Table 3-11 (Cont.) NF Access Token

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
`nfaccesstoken.oauth.nrfInstanceId`	OCNRF's NF Instance ID that is used for signing AccessTokenClaim	6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c	M		NRF Instance ID that is used for signing AccessTokenClaim (iss IE of AccessTokenClaim). If NRF needs to issue AccessTokenClaim using it's own NF instance ID then the nrfInstanceId configured in the global section needs to configured here again,. If NRF needs to issue AccessTokenClaim using a common/virtual then a common/virtual NF instance ID needs to be configured here (along with the common/virtual PrivateKey and Certificate Pair). The same NF instance id and PrivateKey and Certificate Pair needs to be configured in all other NRFs as well so that tokens issues by all the NRF can be validated using a Single NfIstanceId and KeyPair.
`nfaccesstoken.oauth.privateKey.k8SecretName`	Secret name that contains OCNRF Private key	ocnrfaccesstoken-secret	M, if `nfaccesstoken.enabled` is true		This is a Secret object for OCNRFPrivate Key.
`nfaccesstoken.oauth.privateKey.k8NameSpace`	Namespace in which OCNRF Private key is present	ocnrf	M, if `nfaccesstoken.enabled` is true		

Table 3-11 (Cont.) NF Access Token

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
nfaccesstoken.oauth.privateKey.rsa.filename	OCNRF's Private Key (RSA type) file name	rsa_private_key.pem	M, if nfaccesstoken.enabled is true and nfaccesstoken.oauth.initialAlgorithm is RSA256		If initialAlgorithm is configured as RSA, then rsa file name must be configured. Otherwise OCNRF gateway will not comeup.
nfaccesstoken.oauth.privateKey.ecdsa.filename	ECDSA key file names	ecdsa_private_key.pem	M, if nfaccesstoken.enabled is true and nfaccesstoken.oauth.initialAlgorithm is ES256		If initialAlgorithm is configured as ECDSA, then rsa file name must be configured. Otherwise OCNRF's NFAccessToken microservice will not comeup.
nfaccesstoken.oauth.certificate.k8SecretName	Secret name that contains OCNRF's certificate	ocnrfaccesstoken-secret	M, if nfaccesstoken.enabled is true		This is a Secret object for OCNRFcertificate details for HTTPS.
nfaccesstoken.oauth.certificate.k8NameSpace	Namespace in which k8SecretName is present	ocnrf	M, if nfaccesstoken.enabled is true		

Table 3-11 (Cont.) NF Access Token

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
`nfaccesstoken.oauth.certificate.rsa.filename`	OCNRF's certificate (RSA type) file name	rsa_certificate.crt	M, if `nfaccesstoken.enabled` is true and `nfaccesstoken.oauth.initialAlgorithm` is RSA256		If initialAlgorithm is configured as RSA, then rsa file name must be configured. Otherwise OCNRF's NFAccessToken microservice will not comeup.
`nfaccesstoken.oauth.certificate.ecdsa.filename`	OCNRF's certificate (ECDSA type) file name	ecdsa_certificate.crt	M, if `nfaccesstoken.enabled` is true and `nfaccesstoken.oauth.initialAlgorithm` is ES256		If initialAlgorithm is configured as ECDSA, then rsa file name must be configured. Otherwise OCNRF's NFAccessToken microservice will not comeup.
`nfaccesstoken.oauth.keyStorePassword.k8SecretName`	Secret name that contains OCNRF's keystore password	ocnrfaccesstoken-secret	M, if `nfaccesstoken.enabled` is true		
`nfaccesstoken.oauth.keyStorePassword.k8NameSpace`	Namespace in which OCNRF's keystore password is present	ocnrf	M, if `nfaccesstoken.enabled` is true		Password that is used for creating in-memory Java Key Store (JKS)

Table 3-11 (Cont.) NF Access Token

Parameter	Description	Default value	Mandatory (M) / Optional (O)	Range or Possible Values (If applicable)	Notes
`nfaccesstoken.oauth.keyStorePassword.filename`	KeyStore password file	keystore_password.txt	M, if `nfaccesstoken.enabled` is true		
`nfaccesstoken.oauth.initialAlgorithm`	Initial Algorithm for Access Token key certificate infrastructure	ES256	O	ES256, RSA256	
`nfaccesstoken.log.level`	Logging level	WARN	O	OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE	

4
Upgrading OCNRF

This section includes information about upgrading an existing OCNRF deployment.

When you attempt to upgrade an existing OCNRF deployment, the running set of containers and pods are replaced with the new set of containers and pods. However, If there is no change in the pod configuration, the running set of containers and pods are not replaced.

If you need to change any configuration then change the `ocnrf-custom-values-1.6.1.yaml` file with new values.

> **Note:**
>
> It is advisable to create a backup of the file before changing any configuration.

To configure the parameters, see section OCNRF Configuration.

> **Caution:**
>
> Upgrading OCNRF within same release supports only configuration changes.

Execute the following command to upgrade an existing OCNRF deployment:

```
$ helm upgrade <release> <helm chart> [--version <OCNRF version>] -f
<ocnrf_customized_values.yaml>
```

For example:

```
$ helm upgrade <release> <helm chart> [--version <OCNRF version>] -f
ocnrf-custom-values-1.6.1.yaml
```

To check the status of the upgrade, execute:

```
helm status <helm-release>
```

For example: `helm status ocnrf`

Table 4-1 Parameters and Definitions during OCNRF Upgrade

Parameters	Definitions
`<helm chart>`	It is the name of the chart that is of the form `<repository/ocnrf>`. For example: `reg-1/ocnrf or cne-repo/ocnrf`

Table 4-1 (Cont.) Parameters and Definitions during OCNRF Upgrade

Parameters	Definitions
`<release>`	It can be found in the output of `helm list` command

In case of backout:

1. Check the history of helm deployment:
   ```
   helm history <helm_release>
   ```
2. Rollback to the required revision:
   ```
   helm rollback <release name> <revision number>
   ```

5
Uninstalling OCNRF

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

Deleting the OCNRF deployment

This procedure explains how to delete the OCNRF deployment:

1. Execute the following command to completely delete or remove the OCNRF deployment:

   ```
   $ helm del --purge <helm-release>
   ```

   Example:

   ```
   $ helm del --purge ocnrf
   ```

2. Execute the following command to delete kubernetes namespace:

   ```
   $ kubectl delete namespace <ocnrf kubernetes namespace>
   ```

   Example:

   ```
   $ kubectl delete namespace ocnrf
   ```

Deleting the OCNRF MySQL details

This procedure explains how to delete the OCNRF MySQL database after deletion of OCNRF deployment.

> **Note:**
>
> Procedure can be different for Geo-Redundant OCNRF sites and standalone OCNRF site.

Procedure for Geo-Redundant OCNRF sites

1. Login to the machine which has permission to access the SQL nodes of NDB cluster.
2. Connect to the SQL node of NDB cluster successively. MySQL commands must be run on all the SQL nodes.
3. Login to the MySQL prompt using root permission or user, which has permission to delete the table records. For example: `mysql -h 127.0.0.1 -uroot -p`

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

4. Execute the following command to delete data specific to purged site:

```
$ DELETE FROM NfScreening WHERE nrfInstanceId = '<OCNRF's NF Instance
ID of Site under deletion>';
$ DELETE FROM NrfSystemOptions WHERE nrfInstanceId = '<OCNRF's NF
Instance ID of Site under deletion>';
$ DELETE FROM NfInstances WHERE nrfInstanceId = '<OCNRF's NF Instance
ID of Site under deletion>';
$ DELETE FROM NfStatusMonitor WHERE nrfInstanceId = '<OCNRF's NF
Instance ID of Site under deletion>';
$ DELETE FROM NfSubscriptions WHERE nrfInstanceId = '<OCNRF's NF
Instance ID of Site under deletion>';
```

5. Exit from MySQL prompt and SQL nodes

> **Note:**
>
> Execute the commands on any one SQL node on one geo-redundant site. Other Geo-redundant sites will get the data records removed automatically.
> If the last site of a Geo-Redundant OCNRF is getting un-installed, then that site can be considered as a Standalone site as there is no DB replication will take place. Hence, procedure documented under section 5.2.2 and 5.2.3 can be followed for cleaning up OCNRF MySQL resource.

Procedure for standalone OCNRF site

1. Login to the machine which has permission to access the SQL nodes of NDB cluster
2. Connect to the SQL node of NDB cluster successively. MySQL commands must be run on all the SQL nodes.
3. Login to the MySQL prompt using root permission or user, which has permission to drop the tables. For example: `mysql -h 127.0.0.1 -uroot -p`

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

4. Execute the following commands to drop the tables:

```
$ DROP TABLE IF EXISTS 'NfInstances';
$ DROP TABLE IF EXISTS 'NfStatusMonitor';
$ DROP TABLE IF EXISTS 'NfSubscriptions';
$ DROP TABLE IF EXISTS 'NfScreening';
$ DROP TABLE IF EXISTS 'NrfSystemOptions';
```

5. Exit from MySQL prompt and SQL node

> **Note:**
>
> Execute the commands on any SQL node of standalone site.

Procedure for complete removal of MySql database and username

This procedure explains the steps to complete removal of MySql database and username in below cases:

1. OCNRF is not going to be install on that cluster.
2. Change the MySql database name or MySql user name.

Procedure

1. Login to the machine which has permission to access the SQL nodes of NDB cluster.
2. Connect to the SQL node of NDB cluster successively. MySQL commands must be run on all the SQL nodes.
3. Login to the MySQL prompt using root permission or user, which has permission to drop the tables. For example: `mysql -h 127.0.0.1 -uroot -p`

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

4. Execute the following command to remove database:

```
$ DROP DATABASE if exists <OCNRF database name>;
```

Example

```
$ DROP DATABASE if exists nrfdb;
```

5. Execute the following command to remove the MySql User

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

Example

```
$ DROP USER IF EXISTS nrfusr;
```

6. Exit from MySQL prompt and SQL node.

> **Note:**
>
> Execute the commands on any SQL node of standalone site.