Oracle® Communications

Networks Data Analytics Function Installation and Fault Recovery Guide

Release 23.1.0
F76784-02
April 2023

Oracle Communications Networks Data Analytics Function Installation and Fault Recovery Guide, Release 23.1.0

F76784-02

Contents

My Oracle Support

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

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

- 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**.
- For Hardware, Networking and Solaris Operating System Support, select **3**.

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

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

Acronyms

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

Table Acronyms

Acronym	Description
3GPP	3rd Generation Partnership Project
5GC	5G Core Network
5GS	5G System
AF	Application Function
API	Application Programming Interface
AMF	Access and Mobility Management Function
CNC	Cloud Native Core
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment (CNE)
FQDN	Fully Qualified Domain Name
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure
KPI	Key Performance Indicator
HA	High Availability
IMSI	International Mobile Subscriber Identity
K8s	Kubernetes
ME	Monitoring Events
Network Slice	A logical network that provides specific network capabilities and network characteristics.
NEF	Oracle Communications Cloud Native Core, Network Exposure Function (NEF)
NF	Network Function
NRF	Oracle Communications Cloud Native Core, Network Repository Function (NRF)
NSI	Network Slice Instance. A set of Network Function instances and the required resources (such as compute, storage and networking resources) which form a deployed Network Slice.
NSSF	Oracle Communications Cloud Native Core, Network Slice Selection Function (NSSF)
NWDAF	Network Data Analytics Function
OAM	Operations, Administration, and Maintenance
PLMN	Public Land Mobile Network
REST	Representational State Transfer
SBA	Service Based Architecture
SBI	Service Based Interface
SMF	Session Management Function
SNMP	Simple Network Management Protocol
SUPI	Subscription Permanent Identifier
UDM	Unified Data Management

Table (Cont.) Acronyms

Acronym	Description
UE	User Equipment
URI	Uniform Resource Identifier

What's New in This Guide

This section introduces the documentation updates for Release *23.1.0* in Oracle Communications Network Data Analytics Function (OCNWDAF) Installation and Fault Recovery Guide.

Release 23.1.0 - F76784-02, April 2023

- Removed redundant information about database creation from Installing OCNWDAF CAP4C section.
- Updated the export engine variable parameter in the Install NRF Client section.

Release 23.1.0 - F76784-01, March 2023

- **Support for Georedundancy**: Georedundancy is data replication of one site across multiple sites to efficiently handle failure scenarios and ensure High Availability (HA). OCNWDAF now supports both 2 and 3 site Georedundant deployments. For more information georedundancy parameters, see, Georedundancy Parameters section.
- **Configure OCNWDAF using CNC Console**: Oracle Communications Networks Data Analytics Function (OCNWDAF) now supports configuring different global and service parameters using the CNC Console application. For more information see, Configuring OCNWDAF GUI.
- **NRF Client Service**: This service integrates OCNWDAF with NRF for registration, discovery, and service status or load related information, along with application and performance information services. For more information see, Install NRF Client and NRF Client Parameters sections.
- The installation instructions in the following sections have been updated:
  - Software Requirements
  - Environment Setup Requirements
  - Installation Package Download
  - Installing OCNWDAF CAP4C
  - Verifying and Creating OCNWDAF Namespace
  - Creating Service Account, Role, and RoleBinding
  - Configuring Database, Creating Users, and Granting Permissions
  - Pushing the Images to Customer Docker Registry
  - Seeding Slice Load and Geographical Data for Simulation
- Added a new chapter, Fault Recovery, to describe the various fault recovery scenarios and their solutions. The *Oracle Communications Network Data Analytics Function (OCNWDAF) Disaster Recovery Guide* is deprecated from this release and all content from the document is moved to this chapter.

1 Introduction

This document provides information on how to install Oracle Communications Network Data Analytics Function (OCNWDAF) and its microservices. It also includes information on performing fault recovery for OCNWDAF.

1.1 Overview

Oracle Communications Networks Data Analytics Function (OCNWDAF) is a Network Function (NF) in the 5G core network.

OCNWDAF is a NF that assists in collecting and analyzing data in a 5G network. It enables the operator to collect and analyze the data in the network through an analytics function. The 5G technology requires prescriptive analytics to drive closed-loop automation and self-healing networks. In a 5G network, the consumers and producers of data are 5G Network Functions (NFs), Application Functions (AFs), and Operations, Administration, and Maintenance (OAM).

OCNWDAF broadly supports the following functions:

- OCNWDAF collects data from AMF, SMF, and NRF in the network. The data is collected directly from the NFs or through the Network Exposure Function (NEF).
- OCNWDAF is designed to provide analytics information to consumer NFs.

A 5G network contains a vast number of devices and sensors generating an enormous amount of data. OCNWDAF allows the Communications Service providers (CSPs) to efficiently monitor, manage, automate, and optimize their network operations after analyzing the data collected across the network. OCNWDAF also helps the CSPs in achieving operational efficiency and provides an enhanced service experience.

The analytics information provided by OCNWDAF is either statistical information on past events or predictive information which can be used to balance the resources on the network. Based on the collected analytics information, the CSPs can roll out new services or modify the existing services without waiting for a maintenance window in the network. This also ensures significantly fewer chances of network experiencing downtime.

A OCNWDAF consumer can avail analytics information for different analytic events. Alternatively, the consumers can subscribe or unsubscribe for specific analytics information as a one-time event or periodically get notified when a specifically defined event is detected.

Multiple instances of OCNWDAF may be deployed in the 5G network. NRF discovers OCNWDAF instances for the consumer network functions. The OCNWDAF information can also be locally configured on the consumer NFs. The OCNWDAF selection function in the consumer NF selects an OCNWDAF instance based on the available OCNWDAF instances by using the OCNWDAF discovery principles.

Different OCNWDAF instances present in the 5G network can be configured to provide a specific type of analytics information. This information about the OCNWDAF instance is described in the OCNWDAF profile stored in the NRF. The OCNWDAF instance provides the list of Analytics IDs that it supports when registering to the NRF, apart from other NRF registration elements required for registration on the NRF. The consumer NFs that need specific analytics types query the NRF and include the Analytics ID based on the required data.

1.2 References

For more information about OCNWDAF, refer to the following documents:

- *Oracle Communications Networks Data Analytics Function User Guide*
- *Oracle Communications Networks Data Analytics Function Solutions Guide*
- *Oracle Communications Networks Data Analytics Function Troubleshooting Guide*
- *Oracle Communications Cloud Native Environment Installation and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core cnDBTier Installation,Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core cnDBTier User Guide*

2
Prerequisites

Before you begin with the procedure for installing OCNWDAF, ensure that the following requirements are met:

- Software Requirements
- Environment Setup Requirements
- Resource Requirements

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

2.1 Software Requirements

This section describes the software requirements for installing OCNWDAF:

Oracle Communications Cloud Native Environment Specification

Oracle Communications Network Data Analytics Function (OCNWDAF) 23.1.0 can be installed on OCI, Oracle Communications, Cloud Native Environment (CNE) 1.9.x ,1.10.x and 22.1.x. releases.

Verify the CNE version with the following command:

```
echo $OCCNE_VERSION
```

> **ⓘ Note**
>
> From CNE 1.8.x and later, the container platform is podman instead of docker. For more information about podman installation, see *Oracle Communications Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Mandatory Software

The following software items must be installed before starting the OCNWDAF installation:

Table 2-1 Mandatory Software

Software	Version
Kubernetes	1.20.7, 1.21.7, 1.22.5
HELM	3.1.2, 3.5.0, 3.6.3, 3.8.0

Table 2-1 (Cont.) Mandatory Software

Software	Version
podman	2.2.1, 3.2.3, 3.3.1
cnDBTier	22.4.1
CNC Console	23.1.0

To verify the current helm and Kubernetes version installed on CNE, use the following commands:

- To check Kubernetes version, run the following command:

  ```
  kubectl version
  ```

- To check the Helm version, run the following command:

  ```
  helm3 version
  ```

Additional Software

Depending on your requirement, you may have to install additional software while deploying OCNWDAF. The list of additional software items, along with the supported versions and usage, is given in the following table:

Table 2-2 Additional Software

Software	App Version	Required For
elasticsearch	7.9.3	Logging
elastic-client	0.3.6	Metric Server
elastic-curator	5.5.4	Logging
elastic-exporter	1.1.0	Logging
elastic-master	7.9.3	Logging
logs	3.1.0	Logging
kibana	7.9.3	Logging
grafana	7.5.11	Metrics
prometheus	2.32.1	Metrics
prometheus-kube-state-metrics	1.9.7	Metrics
prometheus-node-exporter	1.0.1	Metrics
metalLb	0.12.1	External IP
metrics-server	0.3.6	Metrics
tracer	1.21.0	Tracing

To verify the installed software items, run the following command:

```
helm3 ls -A
```

If you need any services related to the above software items, and if the respective software is unavailable in CNE, then install that software before proceeding further.

2.2 Environment Setup Requirements

This section provides information on environment setup requirements for installing OCNWDAF.

Network Access

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

- **Local docker image repository** – It contains the OCNWDAF docker images. To check if the Kubernetes cluster hosts can access the local docker image repository, pull any image with an image-tag, using the following command:

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

  where:

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

  `image-name` is the docker image name.

  `image-tag` is the tag assigned to the docker image used for the OCNWDAF pod.

- **Local helm repository** – It contains the OCNWDAF helm charts. To check if the Kubernetes cluster hosts can access the local helm repository, run the following command:

  ```
  helm repo update
  ```

Client Machine Requirements

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

The client machine must meet the following requirements:

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

cnDBTier Requirements

OCNWDAF supports cnDBTier in a virtual CNE (vCNE) environment. cnDBTier must be up and active in case of containerized CNE. For more information on installation procedure, see the *Oracle Communications Cloud Native Core cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

2.3 Resource Requirements

This section provides information about OCNWDAF resource requirements.

Resource Requirements for Helm Test

This section provides the details on resource requirement to install and run OCNWDAF Helm Test.

Helm Test Job
This job runs on demand when helm test command is executed. It runs the helm test and stops after completion. These are short-lived jobs, which gets terminated after the work is completed. Hence, they are not part of active deployment resource, but considered only during helm test procedures.

Table 2-3 Helm Test Requirement

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

Below is an example of the configurations that should be included under the global section of the *oc-nwdaf-custom-values.yaml* file.

```
global:
  testJobResources:
    limits:
      cpu: 2
      memory: 2Gi
      ephemeral-storage: 2Gi
    requests:
      cpu: 1
      memory: 1Gi
      ephemeral-storage: 200Mi
```

3
Downloading Installation Package

This chapter describes how to download Oracle Communications Network Data Analytics Function (OCNWDAF) package.

- Installation Package Download
- Pushing the Images to Customer Docker Registry

3.1 Installation Package Download

This section provides information about how to download OCNWDAF package.

To download the OCNWDAF package from My Oracle Support:

1. Log in to My Oracle Support using the appropriate credentials.
2. Click **Patches & Updates** to locate the patch.
3. In the **Patch Search** console, select the **Product or Family (Advanced)** option.
4. Enter `Oracle Communications Cloud Native Core - 5G` in the **Product** field. Select the product from the **Product** drop-down
5. From the **Release** drop-down, select "**Oracle Communications Network Data Analytics Function <release_number>**" Where, `<release_number>` indicates the required release number of NWDAF.
6. Click **Search**.
   The Patch Advanced Search Results displays a list of releases.
7. Select the required patch from the list. The Patch Details window appears.
8. Click **Download**. The File Download window appears.
9. Click the **<p********_<release_number>_Tekelec>.zip** file.
10. Extract the release package zip file.
    Package is named as follows:

    nwdaf-pkg-<marketing-release-number>.zip

    For example: *nwdaf-pkg-23.1.0.0.zip*

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

Untar the Package ZIP File

Run the following command to untar the OCNWDAF package zip file to get the docker image tar file:

```
tar -xvf nwdaf-pkg-<marketing-release-number>.tgz
```

or

```
unzip  nwdaf-pkg-<marketing-release-number>.zip
```

The `nwdaf-pkg.tgz` directory consists of following files:

```
# Root
- images
- installer/
    - cap4c-installer
        - etc
        - scripts
        - sql
    - nrf-client-installer
        - etc
        - scripts
        - sql
    - nwdaf-cap4c-installer
        - etc
        - scripts
        - sql
    - nwdaf-installer
       - etc
       - scripts
       - sql
    - nwdaf-ats
      - ocn-ats-nwdaf-tool
        - ocn-ats-nwdaf-chart
      - ocnwdaf_tests
        - data
        - features
        - steps
```

3.2 Pushing the Images to Customer Docker Registry

The OCNWDAF deployment package includes ready-to-use docker images (inside the images tar file) and Helm charts to help orchestrate containers in Kubernetes. The communication between service pods of OCNWDAF are preconfigured in the Helm charts.

Table 3-1 Docker Images for OCNWDAF

Service Name	Docker Image Name	Image Tag
NWDAF Analytics Info Service	ocn-nwdaf-analytics	23.1.0.0.0
NWDAF Communication Service (Egress Gateway)	ocn-nwdaf-communication	23.1.0.0.0
NWDAF Configuration Service	ocn-nwdaf-configuration-service	23.1.0.0.0
NWDAF Data Collection Service	ocn-nwdaf-data-collection-service	23.1.0.0.0
NWDAF Gateway Service (Ingress Gateway)	ocn-nwdaf-gateway-service	23.1.0.0.0
NWDAF MTLF Service	ocn-nwdaf-mtlf-service	23.1.0.0.0
NWDAF Subscription Service	ocn-nwdaf-subscription-service	23.1.0.0.0
AMF NF Simulator Service	ocn-amf-simulator-service	23.1.0.0.0

Table 3-1 (Cont.) Docker Images for OCNWDAF

Service Name	Docker Image Name	Image Tag
SMF NF Simulator Service	ocn-smf-simulator-service	23.1.0.0.0
NRF NF Simulator Service	ocn-nrf-simulator-service	23.1.0.0.0
Mesa Simulator Service (Data Generator)	mesa-simulator	2.22.4.0.0
cap4c ML model controller	cap4c-model-controller	23.1.0.0.0
cap4c ML model executor	cap4c-model-executor	23.1.0.0.0
cap4c stream analytics	cap4c-stream-analytics	23.1.0.0.0
kafka to mysql serializer	cap4c-kafka-ingestor	23.1.0.0.0
Reporting service	nwdaf-cap4c-reporting-service	23.1.0.0.0
kafka	nwdaf-cap4c-kafka	3.3.1
nwdaf-cap4c-scheduler	nwdaf-cap4c-scheduler-service	23.1.0.0.0
nwdaf-cap4c-spring-cloud-config-server	nwdaf-cap4c-spring-cloud-config-server	2.22.4.0.0
nwdaf-portal	nwdaf-portal	23.1.0.0.0
nwdaf-portal-service	nwdaf-portal-service	23.1.0.0.0
redis	nwdaf-cap4c-redis	7.0.4
zookeeper	nwdaf-cap4c-zookeper	3.8.0
nwdaf-cap4c-initial-setup-script	nwdaf-cap4c-initial-setup-script	2.22.4.0.0
ocn-ats-nwdaf-service	ocats-nwdaf-subs	23.1.0.0.0
ocn-ats-nwdaf-notify-service	ocats-nwdaf-notify-api	23.1.0.0.0
Helm Test	nf-test	22.2.0
geo redundancy agent	ocn-nwdaf-geo-redundacy-agent	23.1.0.0.0

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

1. Verify the package content, checksums of tarballs in the `Readme.txt` file.
2. Load the `nwdaf-images.tar` file into the docker system. Run the following command:

> **Note**
>
> The `nwdaf-images.tar` file becomes available, once you have untarred the OCNWDAF package zip file. For more information, see Installation Package Download.

```
docker load --input <image_file_name.tar>
```

Example:

```
docker load --input images
```

3. Push the Docker images to the docker repository, run the following command:

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

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

   > **Note**
   >
   > It is recommended to configure the docker certificate before running the push command to access customer registry via HTTPs, otherwise, docker push command may fail.

4. Verify if the image is loaded correctly by running the following command. Run the following command:

   ```
   docker images
   ```

5. Push the Helm charts to the Helm repository. Run the following command:

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

4 OCNWDAF Installation

This chapter describes how to install Oracle Communications Network Data Analytics Function (OCNWDAF) on Oracle Communications Cloud Native Environment (CNE).

The steps are divided into two categories:

- Preinstallation
- Installation Tasks

You are recommended to follow the steps in the given sequence for preparing and installing OCNWDAF.

4.1 Preinstallation

To install OCNWDAF, perform the steps described in this section.

> **Note**
>
> The kubectl commands might vary based on the platform used for deploying CNC Policy. Users are recommended to replace `kubectl` with environment-specific command line tool to configure kubernetes resources through kube-api server. The instructions provided in this document are as per the CNE's version of kube-api server.

4.1.1 Creating Service Account, Role, and RoleBinding

This section describes the procedure to create service account, role, and rolebinding.

> **Important**
>
> The steps described in this section are optional and you can skip it in any of the following scenarios:
>
> - If service accounts are created automatically at the time of OCNWDAF deployment.
> - If the global service account with the associated role and role-bindings is already configured or if you are using any internal procedure to create service accounts. If a service account with necessary rolebindings is already available, then update the `ocnwdaf/values.yaml` with the account details before initiating the installation procedure. In case of incorrect service account details, the installation fails.

Create Service Account

To create the global service account:

1. Create an OCNWDAF service account resource file:

   ```
   vi <ocnwdaf resource file>
   ```

   Example:

   ```
   vi ocnwdaf-sampleserviceaccount-template.yaml
   ```

2. Update the resource file with the release specific information:

   > **Note**
   >
   > Update <helm-release> and <namespace> with its respective OCNWDAF namespace and OCNWDAF helm release name.

   ```
   apiVersion: v1
   kind: ServiceAccount
   metadata:
   name: <helm-release>-serviceaccount
   namespace: <namespace>
   ```

where, `<helm-release>` is the helm deployment name.
`<namespace>` is the name of the Kubernetes namespace of OCNWDAF. All the microservices are deployed in this Kubernetes namespace.

Define Permissions using Role

To define permissions using roles:

1. Create an OCNWDAF roles resource file:

   ```
   vi <ocnwdaf sample role file>
   ```

   Example:

   ```
   vi ocnwdaf-samplerole-template.yaml
   ```

2. Update the resource file with the role specific information:

   ```
   apiVersion: rbac.authorization.k8s.io/v1
   kind: ClusterRole
   metadata:
     name: <helm-release>-role
   rules:
   - apiGroups: [""]
     resources:
     - pods
     - services
     - configmaps
     verbs: ["get", "list", "watch"]
   ```

Create RoleBindings

To bind the roles with the service account:

1. Create an OCNWDAF rolebinding resource file:

   ```
   vi <ocnwdaf sample rolebinding file>
   ```

   Example:

   ```
   vi ocnwdaf-sample-rolebinding-template.yaml
   ```

2. Update the resource file with the role binding specific information:

   ```
   apiVersion: rbac.authorization.k8s.io/v1
   kind: RoleBinding
   metadata:
   name: <helm-release>-rolebinding
   namespace: <namespace>
   roleRef:
   apiGroup: rbac.authorization.k8s.io
   kind: Role
   name: <helm-release>-role
   subjects:
   - kind: ServiceAccount
   name: <helm-release>-serviceaccount
   namespace: <namespace>
   ```

Create resources

Run the following commands to create resources:

```
kubectl -n <namespace> create -f <service account resource file>;
```

```
kubectl -n <namespace> create -f <roles resource file>;
```

```
kubectl -n <namespace> create -f <rolebinding resource file>
```

> **Note**
>
> Once the global service account is added, users must add `global.ServiceAccountName` in the `ocnwdaf/values.yaml` file; otherwise, installation may fail as a result of creating and deleting custom resource definitions (CRD).

4.1.2 Configuring Database, Creating Users, and Granting Permissions

This section explains how a database administrator can create the databases, users, and grant permissions to the users for OCNWDAF.

Perform the following steps to create the OCNWDAF MySQL database and grant permissions to the OCNWDAF user for database operations:

1. Unzip the package *nwdaf-installer.zip*

   ```
   mkdir nwdaf-installer

   unzip nwdaf-installer.zip -d nwdaf-installer/
   ```

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

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

5. Run the following command to log in to the MySQL prompt as a user with root permissions:

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

6. Copy the file content from *nwdaf-release-package/installer/nwdaf-installer/sql/ocn-nwdaf-db-script-23.1.0.0.0.sql* and run it in the current MySQL instance.

4.1.3 Verifying and Creating OCNWDAF Namespace

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

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

```
$ kubectl get namespaces
```

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

> **Note**
>
> This is an optional step. Skip this step if the required namespace already exists.
>
> ```
> $ kubectl create namespace <required namespace>
> ```
>
> Example:
>
> ```
> $ kubectl create namespace oc-nwdaf
> ```

Naming Convention for Namespaces

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

- Starts and ends with an alphanumeric character

- Contains 63 characters or less
- Contains only alphanumeric characters or '-'

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

4.1.4 Verifying Installer

A folder named *installer* is obtained on decompressing the release package, copy this folder to the Kubernetes bastion home. To verify if the *installer*, run the command:

```
[ -d "./nwdaf-release-package/installer/nwdaf-installer" ] && echo "nwdaf-
installer exist"
[ -d "./nwdaf-release-package/installer/cap4c-installer" ] && echo "cap4c-
installer exist"
[ -d "./nwdaf-release-package/installer/nwdaf-cap4c" ] && echo "nwdaf-cap4c
exist"
[ -d "./nwdaf-release-package/installer/nwdaf-ats" ] && echo "nwdaf-ats exist"
```

Sample output:

```
nwdaf-installer exist

nwdaf-cap4c exist

cap4c-installer exist

nwdaf-ats exist
```

4.2 Installation Tasks

This section describes the tasks that the user must follow for installing OCNWDAF.

4.2.1 Install NRF Client

This section describes the procedure to install the NRF client and configure the NRF client parameters.

> **Note**
>
> These configurations are required when an NF has to register with the NRF. Before you proceed with NRF client configuration changes, NRF client service should be enabled.

Installing NRF client

Preparation

Set the required environment variables, run the commands:

```
export K8_NAMESPACE="<nwdaf-kubernetes-namespace>"
export MYSQL_HOST="<db-host>"
export MYSQL_PORT="<db-port>"
export MYSQL_ENGINE="<db-engine>"
```

Configure The Required Databases

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

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

3. Connect to MySQL prompt as a user with root permissions:

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

4. Copy the DDL file content from *nwdaf-release-package/nrf-client-installer/sql/ddl.sql* and run it in current MySQL instance.
5. Copy the DCL file content from *nwdaf-release-package/nrf-client-installer/sql/dcl.sql* and run it in current MySQL instance

Install nrf-client

Run the script `install-nrf-client.sh`

```
sh ./nwdaf-release-package/nrf-client-installer/scripts/install-nrf-client.sh
```

Verify the Installation

Run the following command to verify the dependencies:

```
kubectl get pods -n ${K8_NAMESPACE}
```

Verify the status of all the dependencies, ensure they are **RUNNING**. For example:

```
[service-user@blurr7-bastion-1 nwdaf-nrf-client-installer]$ kubectl get pods -
n ${K8_NAMESPACE}
NAME                                                        READY
STATUS              RESTARTS   AGE
nrfclient-appinfo-5495cb8779-vjdqd                          1/1
Running             0          2m50s
nrfclient-ocnf-nrf-client-nfdiscovery-6c6787996f-f4zdn      1/1
Running             0          2m34s
nrfclient-ocnf-nrf-client-nfdiscovery-6c6787996f-w9d8d      1/1
Running             0          2m50s
nrfclient-ocnf-nrf-client-nfmanagement-7dddb57fb7-bkf7d     1/1
Running             0          2m50s
nrfclient-ocpm-config-6d8f49dd6-vfl5j                       1/1
Running             0          2m50s
```

You should configure the NRF Client parameters in the ocnwdaf-<*version*>.custom-values.yaml file, for complete list of NRF Client parameters see NRF Client Parameters.

4.2.2 Installing OCNWDAF CAP4C

Set Up the Required Environment Variables

Export the following required environment variables:

```
export IMAGE_REGISTRY="<image-registry-uri>"
export K8_NAMESPACE="<nwdaf-kubernetes-namespace>"
export ENCRYPT_KEY="<symetric-encrypt-key>"
```

Replace the <*image-registry-uri*>, <*nwdaf-kubernetes-namespace*>, <*symetric-encrypt-key*> with the appropriate values. For instance, the K8_NAMESPACE should be set to ocn-nwdaf as per the OCNWDAF installation steps provided in previous section to create namespace.

Configure the Database

CAP4C

Follow the procedure to create CAP4C MySQL database and grant user permissions to the required users, to perform the necessary operations on the database:

1. Unzip the *package cap4c-installer.zip*

   ```
   mkdir cap4c-installer

   unzip cap4c-installer.zip -d cap4c-installer/
   ```

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

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

4. Run the following command to log in to the MySQL prompt as a user with root permissions. A user with root permission can create users with required permissions, to perform necessary operations on the database.

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

5. Copy the DDL file content from *nwdaf-release-package/installer/cap4c-installer/sql/ddl.sql* and run it in the current MySQL instance.
6. Copy the DCL file content from *nwdaf-release-package/installer/cap4c-installer/sql/dcl.sql* and run it in the current MySQL instance.

> **Note**
>
> You can change the DB user and password by editing the file *ocn-nwdaf-db-script-23.1.0.0.0.sql* before you run it in the above step, if you change DB user and password ensure that you export the property variables as described in the Verifying Dependencies procedure .

Installing Dependencies

1. Unzip the package *nwdaf-cap4c-installer.zip*

   ```
   mkdir nwdaf-cap4c-installer

   unzip nwdaf-cap4c-installer.zip -d nwdaf-cap4c-installer/
   ```

2. Run `01-install-dependencies.sh` as follows:

   ```
   cd nwdaf-cap4c-installer/scripts

   sh 01-install-applications.sh
   ```

Verifying Dependencies

Run the following command to verify if the dependencies are running:

```
kubectl get pods -n ${K8_NAMESPACE}
```

Verify if the status of all the dependencies (listed below) is **Running**. If the status of any dependency is not **Running**, wait until a maximum of 5 restarts have occurred.

```
NAME                                                              READY
STATUS    RESTARTS   AGE
kafka-sts-0                                                       1/1
Running   0          01s
kafka-sts-1                                                       1/1
Running   0          01s
kafka-sts-2                                                       1/1
Running   0          01s
nwdaf-cap4c-spring-cloud-config-server-deploy-xxxxxxxxx-xxxxx   1/1
Running   0          01s
redis-master-pod                                                  1/1
Running   0          01s
redis-slave-sts-0                                                 1/1
Running   0          01s
redis-slave-sts-1                                                 1/1
Running   0          01s
zookeper-sts-0                                                    1/1
Running   0          01s
```

The following is the list of default variables used to configure OCNWDAF. You can export any variable and it will be replaced in the deployed configuration.

- MYSQL_HOST
- MYSQL_PORT
- KAFKA_BROKERS
- DRUID_HOST
- DRUID_PORT
- REDIS_HOST
- REDIS_PORT
- CAP4C_KAFKA_INGESTOR_DB
- CAP4C_KAFKA_INGESTOR_DB_USER
- CAP4C_KAFKA_INGESTOR_DB_PASSWORD
- CAP4C_MODEL_CONTROLLER_DB
- CAP4C_MODEL_CONTROLLER_DB_USER
- CAP4C_MODEL_CONTROLLER_DB_PASSWORD
- CAP4C_MODEL_EXECUTOR_DB_USER
- CAP4C_MODEL_EXECUTOR_DB_PASSWORD
- CAP4C_STREAM_ANALYTICS_DB
- NWDAF_CAP4C_REPORTING_SERVICE_USER
- NWDAF_CAP4C_REPORTING_SERVICE_PASSWORD
- NWDAF_CAP4C_SCHEDULER_SERVICE_DB
- NWDAF_CAP4C_SCHEDULER_SERVICE_DB_USER
- NWDAF_CAP4C_SCHEDULER_SERVICE_DB_PASSWORD
- NWDAF_CONFIGURATION_HOST
- NWDAF_USER
- NWDAF_DB_PASSWORD

Change the default values for the following variables:

```
export MYSQL_HOST=<any-other-mysql-host>
export MYSQL_PORT=<any-other-mysql-port>
```

Setting up Encrypted Credentials

To encrypt the credentials, run the following commands:

```
export CAP4C_PASSWORD=cap4c_passwd
export CAP4C_MODEL_CONTROLLER_PASSWORD="'{cipher}$(kubectl -n ${K8_NAMESPACE}
exec $(kubectl -n ${K8_NAMESPACE} get pods --no-headers -o custom-
columns=":metadata.name" | grep nwdaf-cap4c-spring-cloud-config-server) --
curl -s localhost:8888/encrypt -d ${CAP4C_PASSWORD})'"

export CAP4C_KAFKA_INGESTOR_DB_PASSWORD="'{cipher}$(kubectl -n $
{K8_NAMESPACE} exec $(kubectl -n ${K8_NAMESPACE} get pods --no-headers -o
custom-columns=":metadata.name" | grep nwdaf-cap4c-spring-cloud-config-
server) -- curl -s localhost:8888/encrypt -d ${CAP4C_PASSWORD})'"
```

```
export CAP4C_MODEL_EXECUTOR_DB_PASSWORD="'{cipher}$(kubectl -n $
{K8_NAMESPACE} exec $(kubectl -n ${K8_NAMESPACE} get pods --no-headers -o
custom-columns=":metadata.name" | grep nwdaf-cap4c-spring-cloud-config-
server) -- curl -s localhost:8888/encrypt -d ${CAP4C_PASSWORD})'"

export NWDAF_CAP4C_REPORTING_SERVICE_PASSWORD="'{cipher}$(kubectl -n $
{K8_NAMESPACE} exec $(kubectl -n ${K8_NAMESPACE} get pods --no-headers -o
custom-columns=":metadata.name" | grep nwdaf-cap4c-spring-cloud-config-
server) -- curl -s localhost:8888/encrypt -d ${CAP4C_PASSWORD})'"

export NWDAF_CAP4C_SCHEDULER_SERVICE_DB_PASSWORD="'{cipher}$(kubectl -n $
{K8_NAMESPACE} exec $(kubectl -n ${K8_NAMESPACE} get pods --no-headers -o
custom-columns=":metadata.name" | grep nwdaf-cap4c-spring-cloud-config-
server) -- curl -s localhost:8888/encrypt -d ${CAP4C_PASSWORD})'"

export NWDAF_DB_PASSWORD=ocn_nwdaf_user_passwd
export NWDAF_DB_PASSWORD="'{cipher}$(kubectl -n ${K8_NAMESPACE} exec $
(kubectl -n ${K8_NAMESPACE} get pods --no-headers -o custom-
columns=":metadata.name" | grep nwdaf-cap4c-spring-cloud-config-server) --
curl -s localhost:8888/encrypt -d ${NWDAF_DB_PASSWORD})'"
```

> **Note**
>
> The *NWDAF_<microservice>_PASSWORD* and *CAP4C_<microservice>_PASSWORD* by default are the same that the DB users created in the section Configure the Database, if you change the password in this step, you must change the passwords in the database before you continue with the installation.

Configuring Dependencies

Run `02-prepare-dependencies.sh`, as follows:

```
cd nwdaf-cap4c-installer/scripts

sh 02-prepare-dependencies.sh
```

Installing CAP4C Images

Run `install-applications.sh`, as follows:

```
cd ~/nwdaf-release-package/installer/cap4c-installer/scripts

sh install-applications.sh
```

Verify if all the services are up and running, run the command:

```
kubectl get pods --namespace=ocn-nwdaf -o wide
```

Sample output:

```
NAME                                                              READY
STATUS              RESTARTS   AGE
nwdaf-cap4c-reporting-service-deploy-59565b4b95-cfls9             1/1
Running             0          3m13s
nwdaf-cap4c-scheduler-service-deploy-67ddc89858-29fcr             1/1
Running             0          3m13s
nwdaf-cap4c-scheduler-service-deploy-67ddc89858-s2gr7             1/1
Running             0          3m13s
cap4c-model-controller-deploy-6469bbccd8-wghkm                    1/1
Running             0          3m30s
cap4c-model-executor-deploy-579969f887-9tdd2                      1/1
Running             0          3m29s
cap4c-stream-analytics-deploy-57c6556865-ddhks                    1/1
Running             0          3m29s
cap4c-stream-analytics-deploy-57c6556865-gp2qs                    1/1
Running             0          3m29s
cap4c-kafka-ingestor-deploy-596498b677-4bmj8                      1/1
Running           0          3m38s
cap4c-kafka-ingestor-deploy-596498b677-xv8lc                      1/1
Running               1          3m38s
```

Install OCNWDAF Images

Run `01-install-applications.sh`, as follows:

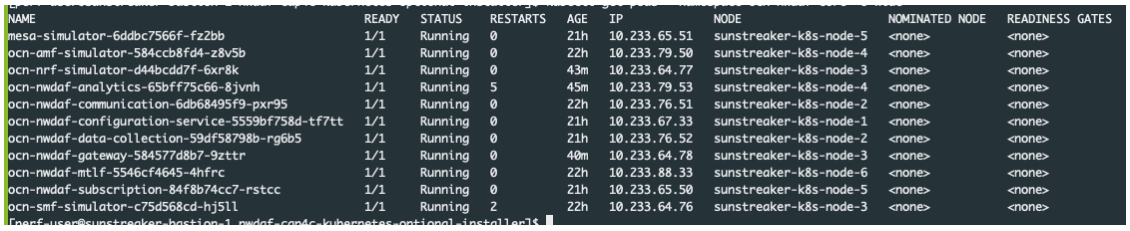
```
cd ~/nwdaf-release-package/installer/nwdaf-installer/scripts

sh install-applications.sh
```

Verify if all the services are up and running, run the command:

```
kubectl get pods --namespace=ocn-nwdaf -o wide
```

Figure 4-1 Sample Output



NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
mesa-simulator-6ddbc7566f-fz2bb	1/1	Running	0	21h	10.233.65.51	sunstreaker-k8s-node-5	<none>	<none>
ocn-amf-simulator-584ccb8fd4-z8v5b	1/1	Running	0	22h	10.233.79.50	sunstreaker-k8s-node-4	<none>	<none>
ocn-nrf-simulator-d44bcdd7f-6xr8k	1/1	Running	0	43m	10.233.64.77	sunstreaker-k8s-node-3	<none>	<none>
ocn-nwdaf-analytics-65bff75c66-8jvnh	1/1	Running	5	45m	10.233.79.53	sunstreaker-k8s-node-4	<none>	<none>
ocn-nwdaf-communication-6db68495f9-pxr95	1/1	Running	0	22h	10.233.76.51	sunstreaker-k8s-node-2	<none>	<none>
ocn-nwdaf-configuration-service-5559bf758d-tf7tt	1/1	Running	0	21h	10.233.67.33	sunstreaker-k8s-node-1	<none>	<none>
ocn-nwdaf-data-collection-59df58798b-rg6b5	1/1	Running	0	21h	10.233.76.52	sunstreaker-k8s-node-2	<none>	<none>
ocn-nwdaf-gateway-584577d8b7-9zttr	1/1	Running	0	40m	10.233.64.78	sunstreaker-k8s-node-3	<none>	<none>
ocn-nwdaf-mtlf-5546cf4645-4hfrc	1/1	Running	0	22h	10.233.88.33	sunstreaker-k8s-node-6	<none>	<none>
ocn-nwdaf-subscription-84f8b74cc7-rstcc	1/1	Running	0	21h	10.233.65.50	sunstreaker-k8s-node-5	<none>	<none>
ocn-smf-simulator-c75d568cd-hj5ll	1/1	Running	2	22h	10.233.64.76	sunstreaker-k8s-node-3	<none>	<none>

All micro services running in K8's and each one is a pod. Each microservice will only have one pod.

OCNWDAF Microservices Port Mapping

Table 4-1 Port Mapping

Service	Port Type	IP Type	Network Type	Service Port	Container Port
ocn-amf-simulator	Internal	ClusterIP	Internal / K8s	8085/TCP	8080/TCP
mesa-simulator	Internal	ClusterIP	Internal / K8s	8097/TCP	8080/TCP
ocn-nrf-simulator	Internal	ClusterIP	Internal / K8s	8084/TCP	8080/TCP
ocn-nwdaf-analytics	Internal	ClusterIP	Internal / K8s	8083/TCP	8080/TCP
ocn-nwdaf-communication	Internal	ClusterIP	Internal / K8s	8082/TCP	8080/TCP
ocn-nwdaf-configuration-service	Internal	ClusterIP	Internal / K8s	8096/TCP	8080/TCP
ocn-nwdaf-data-collection	Internal	ClusterIP	Internal / K8s	8081/TCP	8080/TCP
ocn-nwdaf-gateway	Internal	ClusterIP	Internal / K8s	8088/TCP	8080/TCP
ocn-nwdaf-mtlf	Internal	ClusterIP	Internal / K8s	8093/TCP	8080/TCP
ocn-nwdaf-subscription	Internal	ClusterIP	Internal / K8s	8087/TCP	8080/TCP
ocn-smf-simulator	Internal	ClusterIP	Internal / K8s	8094/TCP	8080/TCP

4.2.3 Seeding Slice Load and Geographical Data for Simulation

To simplify the running of the script, a docker image is created with the *csv_data* and *json_data* embedded into the image.

Verify if the *nwdaf-cap4c-initial-setup-script* pod is running in the cluster, if not, create the values.yml file as below:

```
global:
  projectName: nwdaf-cap4c-initial-setup-script
  imageName: nwdaf-cap4c/nwdaf-cap4c-initial-setup-script
  imageVersion: 2.22.4.0.0
config:
  env:
    APPLICATION_NAME: nwdaf-cap4c-initial-setup-script
    APPLICATION_HOME: /app
```

Run the HELM command:

```
helm install nwdaf-cap4c-initial-setup-script https://artifacthub-
phx.oci.oraclecorp.com/artifactory/ocnwdaf-helm/nwdaf-cap4c-deployment-
template-22.0.0.tgz -f <path_values_file>/values.yml -n <namespace_name>
```

A container will be running inside the K8s cluster, identify the name of the container, run the following command:

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

Sample output:

```
NAME                                                        READY   STATUS
RESTARTS   AGE
nwdaf-cap4c-initial-setup-script-deploy-64b8fbcd9-2vqf9   1/1     Running
0          55s
```

Search for the name and port of the configurator service, to be used later on. Run the command:

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

Sample output:

```
NAME                                                  TYPE        CLUSTER-
IP      EXTERNAL-IP   PORT(S)          AGE
ocn-nwdaf-configuration-service                       ClusterIP
10.233.57.212   <none>        8096/TCP         38d
```

Run the following command to access the container:

```
$ kubectl exec -n <namespace_name> nwdaf-cap4c-initial-setup-script-
deploy-64b8fbcd9-2vqf9 -it bash
```

Once the container is accessed, run the following script to begin the data loading, starting with the loading of **slices**:

> **Note**
>
> Instead of using the IP address the host name is used.

```
$ python3 cells_loader.py -c /app-data/csv_data -f slices -i
<configurator_service_name>.<namespace_name>:<configurator_service_port> -t 2
```

For example:

```
python3 cells_loader.py -c /app-data/csv_data -f slices -i ocn-nwdaf-
configuration-service-internal:8096 -t 2
```

To begin loading the **cells**, run the command:

```
$ python3 cells_loader.py -c /app-data/csv_data -j /app-data/json_data -f
austin_cells -i
<configurator_service_name>.<namespace_name>:<configurator_service_port> -t 1
```

For example:

```
python3 cells_loader.py -c /app-data/csv_data -j /app-data/json_data -f
austin_cells -i ocn-nwdaf-configuration-service-internal:8096 -t 1
```

Note that the **-f** parameter indicates the data source file being used, while **-t** defines the type of load.

Other optional parameters:

- `-c / --csv_directory [full path]`

  This parameter indicates the data directory which contains the specified filename .csv file. It must be the full system path of the directory containing the .csv data file.

- `-j / --json_directory [full path]`

  This parameter indicates the data directory which contains the specified filename .json file. It must be the full system path of the directory containing the .json data file.

- `-i / --hostIp [host IP]`

  This is the IP address of the server where the Configurator service is installed, it must be reachable from the machine that runs this script. The IP address must contain the port when appropriate.

- `-p / --protocol [protocol]`

  This is the protocol used by the Configurator service. Valid values are "http" or "https". Default is "http".

4.2.4 Verifying OCNWDAF Installation

This section describes how to verify if OCNWDAF is installed successfully.

To check the installation status, run any of the following commands:

```
helm3 ls release_name -n <release-namespace>
```

Example:

```
helm3 ls ocnwdaf -n ocnwdaf_namespace
```

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

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

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

Example:

```
kubectl get pod -n ocnwdaf
```

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

Run the following command to get status of services:

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

Example:

```
kubectl get services -n ocnwdaf_namespace
```

> **Note**
>
> Take a backup of the following files that are required during disaster recovery:
>
> - Updated ocnwdaf-custom-values.yaml file
> - Updated helm charts
> - secrets, certificates, and keys that are used during installation

If the installation is not successful or you do not see the status as **Running** for all the pods, contact My Oracle Support (MOS).

4.2.5 Performing Helm Test

Helm Test is a feature that validates the successful installation of OCNWDAF and determines if the NF is ready to take traffic. The pods are tested based on the namespace and label selector configured for the helm test configurations.

> **Note**
>
> Helm Test can be performed only on helm3.

Prerequisite: To perform the helm test, you must have the helm test configurations completed under the "Global Parameters" section of the `custom_values.yaml` file. For more information on parameters, see Global Parameters.

Run the following command to perform the helm test:

```
helm3 test <helm-release_name> -n <namespace>
```

where:

`helm-release-name` is the release name.

`namespace` is the deployment namespace where OCNWDAF is installed.

Example:

```
helm3 test ocnwdaf -n ocnwdaf
```

Sample output:

```
NAME: ocnwdaf
LAST DEPLOYED: Mon Nov 14 11:01:24 2022
NAMESPACE: ocnwdaf
STATUS: deployed
 REVISION: 1
TEST SUITE: ocnwdaf-test
Last Started: Mon Nov 14 11:01:45 2022
Last Completed: Mon Nov 14 11:01:53 2022
Phase: Succeeded
NOTES:
# Copyright 2022 (C), Oracle and/or its affiliates. All rights reserved
```

4.2.6 Configuring OCNWDAF GUI

This section describes how to configure Oracle Communications Networks Data Analytics Function (OCNWDAF) GUI using the following steps:

Configure OCNWDAF GUI in CNC Console

Prerequisite: To configure OCNWDAF GUI in CNC Console, you must have CNC Console installed. For information on how to install CNC Console, refer to *Oracle Communications Cloud Native Configuration Console Installation, Upgrade and Fault Recovery Guide.*

Before installing CNC Console, ensure that the instances parameters are updated in the *occncc_custom_values.yaml* file.

If CNC Console is already installed, ensure all the parameters are updated in the *occncc_custom_values.yaml* file. For information refer to *Oracle Communications Cloud Native Configuration Console Installation, Upgrade and Fault Recovery Guide.*

Access OCNWDAF GUI

To access OCNWDAF GUI, follow the procedure mentioned in the "Accessing CNC Console" section of *Oracle Communications Cloud Native Configuration Console Installation, Upgrade and Fault Recovery Guide*.

5 OCNWDAF Customization

Customizing OCNWDAF

This section describes how to customize OCNWDAF.

The OCNWDAF deployment is customized by overriding the default values of various configurable parameters in the `oc-nwdaf-custom-values.yaml` file.

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

1. Unzip the *Custom_Templates* file available in the extracted documentation release package to get the following files that are used to customize the deployment parameters during installation:
   a. `oc-nwdaf-custom-values.yaml`: This file is used to customize the deployment parameters during installation.
   b. `oc-nwdaf-custom-values-ats.yaml`: This file is used to customize the OCNWDAF deployment with ATS.
   c. `nwdafAlertrules.yaml`: This file is used for Prometheus.

   For more information on how to download the package from My Oracle Support, see the Installation Package Download section.
2. Customize the `oc-nwdaf-custom-values.yaml` file.
3. Save the updated `oc-nwdaf-custom-values.yaml` file in the helm chart directory.

> **Note**
>
> - All parameters mentioned as mandatory must be present in Custom Values YAML file and configured before the deployment.
> - All fixed value parameters listed must be present in the Custom Values YAML file with the exact values as specified in this section.

5.1 Configurable Parameters

This section lists all the OCNWDAF configurable parameters.

5.1.1 Global Parameters

Table 5-1 Global Paramaters

Parameter	Description	Details
`global.test.nfName`	This is a mandatory parameter. The value of *nfName* is specified as *ocnwdaf*. The *nfName* is used as a prefix in the test container name.	**Default value:** ocnwdaf
`global.test.image.name`	This is a mandatory parameter. Name of the test image.	**Default value:** nf_test
`global.test.image.tag`	This is a mandatory parameter. Name of the test image tag.	**Default value:** 22.2.0
`global.test.config.logLevel`	This is an optional parameter. Helm test hook-related configurations. It indicates the logging level of the test container.	**Range:** INFO, DEBUG, FATAL, ERROR, WARN **Default value**: INFO
`global.test.config.timeout`	This is an optional parameter. Helm test hook-related configurations. Beyond this duration, helm test is considered as a failure.	**Default value**: 240
`global.test.complianceEnable`	This is an optional parameter. Helm test hook-related configurations. Sets the test container as compliant.	**Range:** true, false **Default value:** true
`global.test.resources`	This is a mandatory parameter. It specifies the resources to which the test pod needs access.	**Default value:** - deployments/v1 - configmaps/v1 - serviceaccounts/v1 -poddisruptionbudgets/v1 - roles/v1 - statefulsets/v1 - services/v1 - rolebindings/v1

Example of the configurations that should be included under the global section of the *ocnwdaf-custom-values.yaml* file.

```
global:
  test:
    nfName: ocnwdaf
    image:
      name: nf_test
      tag: 22.2.0
    config:
      logLevel: INFO
      timeout: 240
    complianceEnable: true
    resources:
      - deployments/v1
```

```
- configmaps/v1
- serviceaccounts/v1
- poddisruptionbudgets/v1
- roles/v1
- statefulsets/v1
- services/v1
- rolebindings/v1
```

5.1.2 Microservice Parameters

For each microservice, there is an option to specify the following parameters in its own *values.yaml* file to perform the *readinessProbe*.

Table 5-2 Microservice Parameter

Parameter	Description	Details
`readinessProbe.initialDelaySeconds`	This is an optional parameter. Specifies the configurable wait time before performing the first readiness probe by Kubelet.	**Default value**: *20*
`readinessProbe.timeoutSeconds`	This is an optional parameter. Number of seconds after which the probe times out.	**Default value**: 3
`readinessProbe.periodSeconds`	This is an optional parameter. Specifies the time interval for every readiness probe check performed by Kubelet.	**Default value**: 10
`readinessProbe.successThreshold`	This is an optional parameter. Minimum consecutive successes for the probe to be considered successful after having failed.	**Default value**: 1
`readinessProbe.failureThreshold`	This is an optional parameter. When a Pod starts and the probe fails, Kubernetes will try *failureThreshold* times before giving up.	**Default value**: 3

Following is an example of the configurations that should be included in the *values.yaml* file of each microservice.

```
readinessProbe:
  initialDelaySeconds: 20
  timeoutSeconds: 3
  periodSeconds: 10
  successThreshold: 1
  failureThreshold: 3
```

5.1.3 Georedundancy Parameters

Configure the following parameters to enable and configure georedundancy in the custom values file for OCNWDAF:

Table 5-3 REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
ocnwdaf.cluster.namespace	Name space of the deployment **Note** Change this to the namespace of OCNWDAF deployments.	ocn-nwdaf
ocnnwdaf.geored.hooks.database	Database information for the hook	nwdaf_subscription
ocnnwdaf.geored.hooks.table	Table information for the hook	nwdaf_subscription
ocnnwdaf.geored.hooks.column1	Column1 information for the hook	record_owner

Table 5-3 (Cont.) REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
ocnnwdaf.geored.hooks.column2	Column2 information for the hook	current_owner
ocnnwdaf.geored.hooks.image	Image information for the hook	ocnwdaf-docker.dockerhub-phx.oci.oraclecorp.com/nwdaf-cap4c/nwdaf-cap4c-mysql:8.0.30
ocnnwdaf.geored.agent.name	Name of the deployment	ocn-nwdaf_georedagent
ocnnwdaf.geored.agent.replicas	Number of Replicas	1
ocnnwdaf.geored.agent.image.source	Image for GRD Agent **Note** Modify this value if the image is in a different repository.	occne-repo-host:5000/occne/redagent-ms-dev:1.0.31
ocnnwdaf.geored.agent.image.pullPolicy	Image Pull Policy	IfNotPresent

Table 5-3 (Cont.) REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
ocnnwdaf.geored.agent.resources.limits.cpu	CPU Limit	1
ocnnwdaf.geored.agent.resources.limits.memory	Memory Limit	1Gi
ocnnwdaf.geored.agent.resources.request.cpu	CPU Request	1
ocnnwdaf.geored.agent.resources.request.memory	Memory Request	1Gi
ocnnwdaf.geored.agent.service.type	Service Type of the Deployment	ClusterIP
ocnnwdaf.geored.agent.service.port.containerPort	Container Port of the Deployment	9181
ocnnwdaf.geored.agent.service.port.targetPort	Target Port of the Deployment	9181
ocnnwdaf.geored.agent.service.port.name	Name of the Service Port	ocnwdafgeoredagentport
ocnnwdaf.geored.agent.service.prometheusport.containerPort	Container Port of the Prometheus	9000
ocnnwdaf.geored.agent.service.prometheusport.targetPort	Target Port of the Prometheus	9000

Table 5-3 (Cont.) REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
ocnnwdaf.geored.agent.service.prometheusport.name	Name of the Prometheus **Note** Modify the port name based on the prometheus used on the deployed setup.	http-cnc-metrics

Table 5-3 (Cont.) REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_SERVER_HTTP2_ENABLED	Enable/Disable HTTP2	TRUE
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_HEARTBEAT_INTERVAL_MS	Time Interval To check HeartBeat in "ms"	10000
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_HEARTBEAT_THRESHOLD	Number of Time Times to check Heart Beat	5
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_CORE_COMP_THRESHOLD	Number of Time Times to check Heart Beat toward Core Components provides	5
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_SITE_NUMBER	Current Site Number	1
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_SITE_ID	Current Site ID	OCNWDAF-XX-1
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_NUMBER_OF_MATED_SITE	Number of Mated Sites. It is updated based on GRD Sites in sync.	1
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_SELF_ADDRESS	Current Agent Address. The resolvable URL of the OCNNWDAF Gateway service. This address should be reachable outside the cluster .	
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_MICROSERVICE_LIVELINESS_MS	Check Interval for OCNWDAF Microservice in "ms".	10000
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_OCNWDAF_CORE_COMPONENT_LIST	List of OCNWDAF microservices that needs to be verified.	ocn-nwdaf-subscription
ocnnwdaf.geored.agent.env.GEO_RED_SITE_SUBSCRIPTION_OWNERSHIP_TRANSFER_URL	Subscription API for Ownership Transfer	http://ocn-nwdaf-subscription-service-internal:8087/nnwdaf-eventssubscription/v1/subscriptions/updateServingOwner
ocnnwdaf.geored.agent.env.GEO_RED_SITE_DATA_COLLECTION_URL	Data Collection API for Ownership Check	http://ocn-nwdaf-data-collection-service-internal:8081/ra/notify
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_DBTIER_REPLICATION_STATUS_URL	DBTier Monitor Service URL for Replication	Use the Reachable Monitor Service from Deployed CNDB namespace. For example: http://mysql-cluster-db-monitor-svc.{cndbnamspace}.svc.{domainname}:8080/db-tier/status/replication
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_DBTIER_STATUS_URL	DBTier Monitor Service URL for Local	Use the Reachable Monitor Service from Deployed CNDB namespace. For example: http://mysql-cluster-db-monitor-svc.{cndbnamspace}.svc.{domainname}:8080/db-tier/status/local

Table 5-3 (Cont.) REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
ocnnwdaf.georedagent.env.GEO_RED_AGENT_SITE_SECONDARY_SITEID	Secondary Site ID	OCNWDAF-XX-2
ocnnwdaf.georedagent.env.GEO_RED_AGENT_SITE_SECONDARY_ADDRESS	Secondary Site Address	The Resolvable URL of the OCNWDAF Gateway of Secondary Site. This address should be reachable outside the cluster
ocnnwdaf.georedagent.env.GEO_RED_AGENT_SITE_TERTIARY_SITEID	Tertiary Site ID	OCNWDAF-XX-3
ocnnwdaf.georedagent.env.GEO_RED_AGENT_SITE_TERTIARY_ADDRESS	Tertiary Site Address	The Resolvable URL of the OCNWDAF Gateway of Tertiary Site. This address should be reachable outside the cluster
ocnnwdaf.georedagent.env.GEO_RED_AGENT_DB_URL	IP of the Site DBTier	The Cluster IP/External IP of the CNDB
ocnnwdaf.georedagent.env.GEO_RED_AGENT_USERNAME	Name of the DB User with privileges to GRD DB and Subscription DB. The user should have access to both GRD and Subscription Databases	occneuser
ocnnwdaf.georedagent.env.GEO_RED_AGENT_PASSWORD	Password for the DB User	password
ocnnwdaf.georedagent.env.GEO_RED_AGENT_ENABLE	Enable/Disable GRD	false
ocnnwdaf.georedagent.env.GEO_RED_AGENT_DB_PORT	Port of the DB	3306
ocnnwdaf.georedagent.env.GEO_RED_AGENT_DB_NAME	Name of the GRD Database	georedagent
ocnnwdaf.georedagent.env.GEO_RED_AGENT_CONFIG_SERVER	Config Server URL for the Site's OCNWDAF	http://nwdaf-cap4c-spring-cloud-config-server:8888

5.1.4 NRF Client Parameters

To configure NRF client parameters, you should configure the following parameters in the ocnwdaf-<*version*>.custom-values.yaml file. The following table provides details about the NRF client customization parameters in the ocnwdaf-<*version*>.custom-values.yaml file:

Table 5-4 Configurable Parameters - NRF Client Configuration

Parameter	Description	Details
global.deploymentNrfClientService.envNfNamespace	This is a mandatory parameter. Contains the namespace of the services to be monitored by performance service.	Default value: No services are monitored, leave blank. **Note** If no services are to be monitored, envNfNamespace can be left blank.
nrf-client.configmapApplicationConfig.profile	This is a mandatory parameter. Contains configuration parameters that goes into nrf-client's config map. **Note** See config-map table for configurable parameters .	Default Value: Valid profile. Please refer to the ocnwdaf-<*version*>custom-values.yaml example below.

Table 5-4 (Cont.) Configurable Parameters - NRF Client Configuration

Parameter	Description	Details
appinfo.infraServices	This is a conditional parameter. Set this parameter to an empty array if any one of the following conditions are met: • Deploying on occne 1.4 or lesser version • Not deploying on OCCNE • Do not wish to monitor infra services such as db-monitor service	NA
perf-info.configmapPerformance.prometheus	This is a conditional parameter. Specifies Prometheus server URL. **Note** If no value is specified, OCNWDAF reports "0" load to NRF.	Default Value: http://prometheus-server.prometheus:5802
global.leaderPodDbName	Contains the name for Leader Pod Database in the global parameters. This database is unique per site.	Range: Valid database name.
nrf-client-nfmanagement.dbConfig.leaderPodDbName	Contains the name for Leader Pod Database in the nrf-client-nfmanagement.dbConfig.	Range: Valid database name.
global.networkDbName	Contains the network database name in the global parameters.	Default Value: Valid Release Database name. For Example: nwdafReleaseDB
nrf-client-nfmanagement.dbConfig.networkDbName	Contains the network database name in the nrf-client-nfmanagement.dbConfig parameters.	Default Value: networkDbNameRef

Table 5-4 (Cont.) Configurable Parameters - NRF Client Configuration

Parameter	Description	Details
nrf-client-nfmanagement.enablePDBSupport	Set this to true to enable PDB Support. **Note** If this parameter is set to true, helm test will fail. It is an expected behavior, as the mode is active and on standby, leader pod (nrf-client-management) will be in a ready state and follower will not be in read	Range: true or false Default Value: true

Table 5-4 (Cont.) Configurable Parameters - NRF Client Configuration

Parameter	Description	Details
	y state, whic h will lead to failur e in the helm test.	
nrf-client-nfmanagement.replicas	Contains number of NRF-Client replicas.	Default Value: 2

Table 5-5 Configurable parameters of NRF Client Configuration in Config-map

Parameter	Description	Details
primaryNrfApiRoot	Primary NRF hostname and port Hostname/IP:Port.	Range: Valid API root Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x
SecondaryNrfApiRoot	secondary NRF hostname and port Hostname/IP:Port For example: nrf2-api-gateway.svc:80	Range: Valid API root Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x
retryAfterTime	When primary NRF is down, this will be the wait Time (in ISO 8601 duration format) after which request to primary NRF will be retried to detect primary NRF's availability. For example: PT120S	Range: Valid ISO 8601 duration format Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x
nrfClientType	The NfType of the NF registering. This should be set to OCNWDAF.	Default Value: OCNWDAF Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x

Table 5-5 (Cont.) Configurable parameters of NRF Client Configuration in Config-map

Parameter	Description	Details
nrfClientSubscribeTypes	NF Type(s) for which the NF wants to discover and subscribe to the NRF. **Note** Leave blank if OCNWDAF does not require this option.	Range: AMF, NRF, UDM, SMF, AUSF, NEF, PCF, SMSF, NSSF, UDR, LMF, GMLC, 5G_EIR, SEPP, UPF, N3IWF, AF, UDSF, BSF, CHF, NWDAF Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x
appProfiles	NfProfile of OCNWDAF to be registered with NRF.	Range: Valid NF Profile Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x
enableF3	Support for 29.510 Release 15.3.	Range: true or false Default Value: true Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x
enableF5	Support for 29.510 Release 15.5.	Range: true or false Default Value: true Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x
renewalTimeBeforeExpiry	Time(seconds) before the subscription validity expires. For example: 3600 (1hr)	Range: Time in seconds Default Value: 3600 Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x
validityTime	The default validity time (days) for subscriptions. For example: 30 (30 days)	Range: Time in days Default Value: 30 Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x

Table 5-5 (Cont.) Configurable parameters of NRF Client Configuration in Config-map

Parameter	Description	Details
enableSubscriptionAutoRenewal	Enable renewal of subscriptions automatically.	Range: true or false Default Value: true Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.6.x
acceptAdditionalAttributes	Enable additionalAttributes as part of 29.510 Release 15.5.	Range: true or false Default Value: false Applicable to: OCNWDAF
enableVirtualNrfResolution	Enable virtual NRF session retry by alternate routing service.	Range: true or false Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.9.0
virtualNrfFqdn	Virtual NRF FQDN used to query static list of route.	Default Value: nrf.oracle.com Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.9.0
virtualNrfScheme	Scheme to be used with the virtual FQDN.	Range: http or https Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.9.0
requestTimeoutGracePeriod	An additional grace period where no response is received from the NRF. This additional period shall be added to the requestTimeout value. This will ensure that the egress-gateway shall first timeout, and send an error response to the NRF-client.	Range: Integer value Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.9.0
nrfRetryConfig	Configurations required for the NRF Retry mechanism.	Default Value: Valid configuration. Please refer to the ocnwdaf-<version>custom-values.yaml example below. Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.9.0
healthCheckConfig	Configurations required for the Health check of NRFs.	Default Value: Valid configuration. Please refer to the ocnwdaf-<*version*>custom-values.yaml example below. Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.9.0
supportedDataSetId	The data-set value to be used in queryParams for NFs autonomous or on-demand discovery.	Range: SUBSCRIPTION, POLICY, EXPOSURE, APPLICATION. Applicable to: OCNWDAF Added, Deprecated, or Updated: Added in Release 1.7.1

Table 5-6 NRF Client Parameters

Parameter	Description	Detail
nrfclient.nrf-client.configmapApplicationConfig.profile	This is a mandatory parameter. It contains configuration parameters that goes into nrf-client's config map. **Note** See configmap table for configurable parameters	Applicable to: OCNWDAF
nrfclient.nrf-client	This is a mandatory parameter. Microservice level control if specific microservice need to be disabled.	Applicable to: OCNWDAF
config-server	This is a mandatory parameter. Details of Config-server microservice.	Applicable to: OCNWDAF
config-server.enabled	This is a mandatory parameter. Engineering-parameter	Applicable to: OCNWDAF
config-server.envMysqlDatabase	This is a mandatory parameter. It specifies MySQL Config Server Database Name.	Default Value: Provisional DB name Applicable to: OCNWDAF
config-server.dbConfig.dbEngine	This is a mandatory parameter. Database hook Configuration	Default Value: Database Engine name Applicable to: OCNWDAF
appinfo.enabled	This is a mandatory parameter. Use to enable or disable appinfo microservices.	Range: true or false Default Value: true Applicable to: OCNWDAF
appinfo.debug	This is a mandatory parameter. Represent appinfo image name. Note: Registry is taken from global section: • image: oc-app-info • appinfo image tag • imageTag: 23.1.0 • Set Log Level to DEBUG. • If false, Log Level shall be INFO	Range: true or false Default Value: false Applicable to: OCNWDAF

Table 5-6 (Cont.) NRF Client Parameters

Parameter	Description	Detail
serviceMeshCheck	Not supported in this OCNWDAF release.	Default Value: Leave blank. Not supported in this OCNWDAF release.
istioSidecarQuitUrl	Not supported in this OCNWDAF release.	Default Value: Leave blank. Not supported in this OCNWDAF release.
istioSidecarReadyUrl	Not supported in this OCNWDAF release.	Default Value: Leave blank. Not supported in this OCNWDAF release.

Table 5-7 NRF Client NF Discovery Parameters

Parameter	Description	Details
nrf-client.nrf-client-nfdiscovery	This is a mandatory parameter. Contains deployment specific configuration for Nrf-Client Discovery Microservice	Applicable to: OCNWDAF
configmapApplicationConfig	This is a mandatory parameter.	Applicable to: OCNWDAF
cpuRequest	This is a mandatory parameter.	Default Value: 2 Applicable to: OCNWDAF
cpuLimit	This is a mandatory parameter.	Default Value: 2 Applicable to: OCNWDAF
memoryRequest	This is a mandatory parameter.	Default Value: 1Gi Applicable to: OCNWDAF
memoryLimit	This is a mandatory parameter.	Default Value: 1Gi Applicable to: OCNWDAF
minReplicas	This is a mandatory parameter.	Default Value: 1 Applicable to: OCNWDAF
maxReplicas	This is a mandatory parameter.	Default Value: 5 Applicable to: OCNWDAF
averageCpuUtil	This is a mandatory parameter.	Default Value: 80 Applicable to: OCNWDAF
commonCfgServer.configServerSvcName	This is a mandatory parameter.	Default Value: nsconfig Applicable to: OCNWDAF
commonCfgServer.port	This is a mandatory parameter.	Default Value: 8080 Applicable to: OCNWDAF
commonCfgClient.enabled	This is a mandatory parameter.	Default Value: *cfgClientEnabled Applicable to: OCNWDAF
nrf-client.nrf-client-nfdiscovery.dbConfig	This is a mandatory parameter. Contains database credentials of Nrf-Client Discovery parameters.	Default Value: Valid database host (Example: ocnwdaf-nsdb.db) Applicable to: OCNWDAF
dbConfig.dbHost	This is a mandatory parameter. DB credential of nrf-client-nfdiscovery parameters.	Applicable to: OCNWDAF
dbConfig.dbPort	This is a mandatory parameter. Databse credential of nrf-client-nfdiscovery parameters.	Default Value: 3306 Applicable to: OCNWDAF

Table 5-7 (Cont.) NRF Client NF Discovery Parameters

Parameter	Description	Details
dbConfig.secretName	This is a mandatory parameter.Contains name of the secret.	Default Value: Provisional DB secret Applicable to: OCNWDAF
dbConfig.dbName	This is a mandatory parameter. Contains name of the database.	Default Value: Provisional DB name Applicable to: OCNWDAF
dbConfig.dbUNameLiteral	This is a mandatory parameter. Contains the name of the Key configured for "DB Username" in Secret with following name: "<dbConfig.secretName>".	Applicable to: OCNWDAF
dbConfig.dbPwdLiteral	This is a mandatory parameter. Contains the name of the Key configured for "DB Password" in Secret with following name: "<dbConfig.secretName>".	Applicable to: OCNWDAF
dbConfig.dbEngine	This is a mandatory parameter. Database engine name.	Default Value: NDBCLUSTER Applicable to: OCNWDAF
serviceMeshCheck	Not supported in this OCNWDAF release.	Default Value: Leave blank. Not supported in this OCNWDAF release.
istioSidecarQuitUrl	Not supported in this OCNWDAF release.	Default Value: Leave blank. Not supported in this OCNWDAF release.
istioSidecarReadyUrl	Not supported in this OCNWDAF release.	Default Value: Leave blank. Not supported in this OCNWDAF release.

Table 5-8 NRF Client NF Management Parameters

Parameter	Description	Details
nrf-client-nfmanagement	This is a mandatory parameter. Contains deployment specific configuration for nrf-client-nfmanagement microservice	Applicable to: OCNWDAF
configmapApplicationConfig	This is a mandatory parameter.	Applicable to: OCNWDAF
replicas	This is a mandatory parameter.	Default Value: 1 Applicable to: OCNWDAF
cpuRequest	This is a mandatory parameter.	Default Value: 1 Applicable to: OCNWDAF
cpuLimit	This is a mandatory parameter.	Default Value: 1 Applicable to: OCNWDAF
memoryRequest	This is a mandatory parameter.	Default Value: 1Gi Applicable to: OCNWDAF
memoryLimit	This is a mandatory parameter.	Default Value: 1Gi Applicable to: OCNWDAF
commonCfgServer.configServerSvcName	This is a mandatory parameter.	Default Value: nsconfig Applicable to: OCNWDAF

Table 5-8 (Cont.) NRF Client NF Management Parameters

Parameter	Description	Details
commonCfgServer.port	This is a mandatory parameter.	Default Value: 8080 Applicable to: OCNWDAF
commonCfgClient.enabled	This is a mandatory parameter.	Default Value: *cfgClientEnabled Applicable to: OCNWDAF
dbConfig	This is a mandatory parameter. Contains database credentials of Nrf-Client Discovery parameters,	Default Value: Valid dbConfig set of properties. Please refer to the ocnwdaf-<version>custom-values.yaml example. Applicable to: OCNWDAF
dbConfig.dbHost	This is a mandatory parameter. DB credential of nrf-client-nfdiscovery parameters.	Default Value: Valid database host (Example: ocnwdaf-nsdb.db) Applicable to: OCNWDAF
dbConfig.dbPort	This is a mandatory parameter. Database credential of nrf-client-nfdiscovery parameters.	Default Value: 3306 Applicable to: OCNWDAF
dbConfig.secretName	This is a mandatory parameter. Contains name of the secret.	Default Value: Provisional DB secret Applicable to: OCNWDAF
dbConfig.dbName	This is a mandatory parameter. Contains name of the database.	Default Value: Provisional DB name Applicable to: OCNWDAF
dbConfig.dbUNameLiteral	This is a mandatory parameter. Contains the name of the Key configured for "DB Username" in Secret with following name: "<dbConfig.secretName>".	Applicable to: OCNWDAF
dbConfig.dbPwdLiteral	This is a mandatory parameter. Contains the name of the Key configured for "DB Password" in Secret with following name: "<dbConfig.secretName>".	Applicable to: OCNWDAF
bConfig.dbEngine	This is a mandatory parameter. Database engine name.	Default Value: NDBCLUSTER Applicable to: OCNWDAF
serviceMeshCheck	Not supported in this OCNWDAF release.	Default Value: Leave blank. Not supported in this OCNWDAF release.
istioSidecarQuitUrl	Not supported in this OCNWDAF release.	Default Value: Leave blank. Not supported in this OCNWDAF release.
istioSidecarReadyUrl	Not supported in this OCNWDAF release.	Default Value: Leave blank. Not supported in this OCNWDAF release.

Here is a sample configuration for NRF client in ocnwdaf-<*version*>custom-values.yaml file:

```
# Please add below in global section of custom values.yaml of NF deployment.
global:
  # The value of nfName is specified as ocnf which is stands of Oracle NF.
  # nfName is used as a prefix in serivce names of nrf client's service and
other services it connects to for eg appinfo, config server etc.
```

```
  nfName: ocnf
  # Global control to enable/disable deployment of NF Discovery service,
enable it if on demand discovery of NF is required.
  nrfClientNfDiscoveryEnable: true
  # Global control to enable/disable deployment of NF Management service.
  nrfClientNfManagementEnable: true
  # Global Parameter to enable/disable DEBUG Container tool
  extraContainers: "DISABLED"
  extraContainersTpl: |
    - command:
        - /bin/sleep
        - infinity
      image: {{ printf "%s/%s:%s" (include "getdockerregistry.name" .)
"ocdebugtool/ocdebug-tools" "22.3.1" }}
      imagePullPolicy: Always
      name: {{ printf "%s-tools-%s" (include "getprefix" .) (include
"getsuffix" .) | trunc 63 | trimPrefix "-" | trimSuffix "-"  }}
      resources:
        requests:
          ephemeral-storage: "2Gi"
          cpu: "0.5"
          memory: "1Gi"
        limits:
          ephemeral-storage: "4Gi"
          cpu: "1"
          memory: "2Gi"
      securityContext:
        allowPrivilegeEscalation: true
        capabilities:
          drop:
          - ALL
          add:
          - NET_RAW
          - NET_ADMIN
        readOnlyRootFilesystem: false
        runAsUser: 7000
  # Global parameter to mention if alternate-route service is
available(deployed) or not.
  alternateRouteServiceEnable: false
  altServiceHTTP2Enabled: true
  altServiceReqTimeout: 3000
  altServiceLookupInterval: 3000
  # Metrics name for OSO
  cncMetricsName: cnc-metrics
  # Jaeger tracing host
  envJaegerAgentHost: ''
  # Jaeger tracing port
  envJaegerAgentPort: 6831
  # Provide value for NodePort
  nrfClientNodePort: 0
  # Docker registry from which config-server images, nrf-client images will
be pulled
  #dockerRegistry: ocnrf-registry.us.oracle.com:5000
  dockerRegistry: cgbu-cnc-comsvc-release-docker.dockerhub-
phx.oci.oraclecorp.com
  # Readiness-Detector image details with tag
```

```
  #Vendor name
  vendor: "Oracle"
  # application name
  applicationName: "nrf-client"
  # prefix for Metrics
  metricPrefix: &metricPrefix ""
  # suffix for Metrics
  metricSuffix: &metricSuffix ""
  # Common service port
  nrfClientCommonServicePort: '9090'
  # Prometheus configuration
  prometheusScrapingConfig:
    enabled: false
    path: "/actuator/prometheus"
  nfInstanceId: 'fe7d992b-0541-4c7d-ab84-c6d70b1b01b1'
  configServerEnable: true
  # Config-Server Service. Shall be used as {{ ReleaseName }}-
configServerFullNameOverride
  configServerFullNameOverride: ocpm-config
  # Mysql Host
  envMysqlHost: &mySqlHostRef 10.233.5.39 # Changed
  envMysqlSecondaryHost: ''
  # Mysql Port
  envMysqlPort: &mySqlPortRef '3306'
  envMysqlSecondaryPort: ''
  # Mysql Secret Name
  dbCredSecretName: &dbCredSecretNameRef 'ocnf-db-pass'
  # Mysql Config Server Databse Name
  # Global Control to disable appinfo service
  appinfoServiceEnable: true
  # Global Control to disable performance info service
  performanceServiceEnable: true
  egressGatewayHost: ocn-nrf-simulator-service.ocnwdaf-ns # Changed
  # Deployment Specific configuration
  deploymentNrfClientService:
    # Services to be monitored by performance service
    # If no services are to be monitored,
envNfNamespace,envNfType,envConsumeSvcName can be left blank
    envNfNamespace: ''
    envNfType: ''
    envConsumeSvcName: ''
    # Egress gateway Host. Shall be used as {{ ReleaseName }}-
envEgressGatewayFullnameOverride
    envEgressGatewayFullnameOverride: egressgateway
    # Egress gateway Port
    envEgressGatewayPort: "8080" # Changed
    # Callback URI to receive Notifications from NRF
    nfApiRoot: http://ocnrf-ingressgateway.ocnrf:80
    nodeSelectorEnabled: false
    nodeSelectorKey: zone
    nodeSelectorValue: app
  # K8s Secret containing Database/user/password for DB Hooks for creating
tables
  privilegedDbCredSecretName: &privDbCredSecretNameRef 'ocnf-privileged-db-
pass'
  releaseDbName: &dbNameRef 'ocnf_config_server'
```

```
  networkDbName: &networkDbNameRef 'nrfNetworkDB'
  # Database Name for LeaderPod
  # Every NF has to configure a different LeaderPod for each site
  # so that when the sites are geo-redundant, the entries in the table do not
crash.
  leaderPodDbName: &leaderPodDbRef 'leaderPodDb'

  # Service Account Name
  serviceAccountName: ""

  # Application name that is added in logs as labels
  app_name: "nrfclient"
  supportedVersions:
    - autoscaling/v2
    - autoscaling/v2beta2
    - autoscaling/v2beta1
    - autoscaling/v1
    - policy/v1
    - policy/v1beta1
  #Resource Values for Hook Jobs
  hookJobResources:
    limits:
      cpu: 2
      memory: 2Gi
    requests:
      cpu: 1
      memory: 1Gi


# Details of perf-info microservices
perf-info:
   image: occnp/oc-perf-info
   imageTag: 22.4.1 # Changed from 22.3.2
   # Service namespace for perf-info
   service_namespace: ocnrf
   # Replicas for perf Info
   replicas: 2
   # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
   imagepullPolicy: IfNotPresent
   service:
     type: ClusterIP
     port: 5905
   resources: {}
     # We usually recommend not to specify default resources and to leave
this as a conscious
     # choice for the user. This also increases chances charts run on
environments with little
     # resources, such as Minikube. If you do want to specify resources,
uncomment the following
     # lines, adjust them as necessary, and remove the curly braces after
'resources:'.
     # limits:
     #  cpu: 100m
     #  memory: 128Mi
     # requests:
     #  cpu: 100m
```

```
    #  memory: 128Mi
  nodeSelector: {}
  tolerations: []
  affinity: {}
  commonCfgClient:
    enabled: false
  commonCfgServer:
  # Do not comment this line in case you want to deploy both config-server
and APIGW in same namespace simultaneously
  # Otherwise comment this line and use 'host'
    configServerSvcName: 'common-config-server'
    host: 'common-config-server'
    port: '80'
    pollingInterval: 5000
  dbConfig:
    dbHost: *mySqlHostRef
    dbPort: *mySqlPortRef
    secretName: *dbCredSecretNameRef
    dbName: *dbNameRef
    # Name of the Key configured for "DB Username" in Secret with following
name: "<dbConfig.secretName>"
    dbUNameLiteral: mysql-username
    # Name of the Key configured for "DB Password" in Secret with following
name: "<dbConfig.secretName>"
    dbPwdLiteral: mysql-password
  ingress:
     enabled: false
  configmapPerformance:
    prometheus: http://prometheus-server.prometheus:5802

# Microservice level control if specific microservice need to be disabled
nrf-client:
  # This config map is for providing inputs to NRF-Client
  configmapApplicationConfig:
     &configRef
     # Config-map to provide inputs to Nrf-Client
     # primaryNrfApiRoot - Primary NRF Hostname and Port
     # SecondaryNrfApiRoot - Secondary NRF Hostname and Port
     # nrfRouteList - Can be used to specify routes with priority instead of
primary and secondary NRF API root
     # useNrfRouteList - Can be used to specify whether nrfRouteList should
be used instead of primary and secondary NRF API root
     # nrfScheme - Scheme of primary and secondary NRF http or https
     # retryAfterTime - Default downtime(in Duration) of an NRF detected to
be unavailable.
     # nrfClientType - The NfType of the NF registering
     # nrfClientSubscribeTypes - the NFType for which the NF wants to
subscribe to the NRF.
     # appProfiles - The NfProfile of the NF to be registered with NRF.
     # enableF3 - Support for 29.510 Release 15.3
     # enableF5 - Support for 29.510 Release 15.5
     # registrationRetryInterval - Retry Interval after a failed autonomous
registration request
     # subscriptionRetryInterval - Retry Interval after a failed autonomous
subscription request
     # discoveryRetryInterval - Retry Interval after a failed autonomous
```

```
discovery request
      # discoveryRefreshInterval - The default interval after which
autonomous discovery requests will be resent
      # discoveryDurationBeforeExpiry - This value specifies the rate at
which the NF shall resend discovery requests to NRF. The value shall be
configured in terms of percentage(1-100). if the discovery ValidityPeriod is
60s, then the discovery requests shall be sent at
discoveryDurationBeforeExpiry * 60/100
      # renewalTimeBeforeExpiry - Time Period(seconds) before the
Subscription Validity time expires.
      # validityTime - The default validity time(days) for subscriptions.
      # enableSubscriptionAutoRenewal - Enable Renewal of Subscriptions
automatically.
      # nfHeartbeatRate - This value specifies the rate at which the NF shall
heartbeat with the NRF. The value shall be configured in terms of
percentage(1-100). if the heartbeatTimer is 60s, then the NF shall heartbeat
at nfHeartBeatRate * 60/100
      # acceptAdditionalAttributes - Enable additionalAttributes as part of
29.510 Release 15.5
      # retryForCongestion - The duration(seconds) after which nrf-client
should retry to a NRF server found to be congested.
      # supportedDataSetId - The data-set value to be used in queryParams for
NFs autonomous/on-demand discovery. e.g. data-set=POLICY
      # enableVirtualNrfResolution- enable virtual NRF session retry by
Alternate routing service
      # virtualNrfFqdn - virtual NRF FQDN used to query static list of route
      # virtualNrfScheme - http or https
      # useAlternateScpOnAlternateRouting - enable use SCP on alternate
routing service
      # subscriberNotificationRetry - number of health status notification
retries to a subscriber
      # requestTimeoutGracePeriod - grace period(seconds) for timeout period
until no response is received from NRF
      # nrfRetryConfig - configurations required for the NRF Retry mechanism
      #serviceRequestType - type of service operation for which configuration
is applicable
      #primaryNRFRetryCount - number of retries for primary instance
      #nonPrimaryNRFRetryCount - number of retries for non-primary instance
      #alternateNRFRetriesCount - number of retry attempts
      #errorReasonsForFailure - httpStatusCode or error conditions for which
retry shall be attempted
      #requestTimeout - timeout period(seconds) where no response is received
from NRF
      #gatewayErrorCodes - httpStatusCode sent by gateway for which retry
shall be attempted
      # healthCheckConfig - configurations required for the Health check of
NRFs
      #healthCheckCount - consecutive success/failure operation count
(default -1 means disabled)
      #healthCheckInterval - interval duration in seconds to perform health
check
      #requestTimeout - timeout period(seconds) where no response is received
from NRF
      #errorReasonsForFailure - httpStatusCode or error conditions for which
the NRF is considered as unhealthy
      #gatewayErrorCodes - httpStatusCode sent by gateway for the NRF shall
```

```
be considered unhealthy
      #enableDiscoveryRefresh - Feature flag to enable Automatic Discovery
Refresh
      #enableRediscoveryIfNoProdNFs - Feature flag to enable rediscovery when
No Producer NFs are available
      #offStatesForRediscoveryIfNoProdNFs - Comma separated value for states
to consider Producer NFs as not available

      #appProfiles=[{"nfInstanceId":"fe7d992b-0541-4c7d-ab84-
c6d70b1b01b1","nfType":"NWDAF","nfStatus":"REGISTERED","plmnList":null,"nsiLis
t":null,"fqdn":"pcf.oracle.com","interPlmnFqdn":null,"ipv4Addresses":null,"ipv
6Addresses":null,"priority":null,"capacity":null,"load":null,"locality":null,"
udrInfo":null,"udmInfo":null,"ausfInfo":null,"amfInfo":null,"smfInfo":null,"up
fInfo":null,"pcfInfo":null,"bsfInfo":null,"customInfo":null,"recoveryTime":nul
l,"nfServices":[{"serviceInstanceId":"03063893-
cf9e-4f7a-9827-067f6fa9dd01","serviceName":"nnrf-nfmanagement","versions":
[{"apiVersionInUri":"v1","apiFullVersion":"1.2.2","expiry":"2019-08-03T18:55:0
8.871+0000"}],"scheme":"http","nfServiceStatus":"REGISTERED","fqdn":"nrf.oracl
e.com","interPlmnFqdn":null,"apiPrefix":"bsf-
service","defaultNotificationSubscriptions":null,"allowedPlmns":null,"allowedN
fTypes":["PCF","NEF",
"NWDAF"],"allowedNfDomains":null,"allowedNssais":null,"priority":null,"capacit
y":null,"load":null,"recoveryTime":null,"supportedFeatures":null}],"sNssais":n
ull}]

      profile: |-
         [appcfg]
         primaryNrfApiRoot=ocn-nrf-simulator-service.ocnwdaf-ns:8080
         secondaryNrfApiRoot=
         nrfRouteList=[{"nrfApi":"nrfDeployName-
nrf-1:8080","scheme":"http","weight":100,"priority":1}]
         useNrfRouteList=false
         nrfScheme=http
         retryAfterTime=PT120S
         nrfClientType=NRF
         nrfClientSubscribeTypes=PCF
         appProfiles=[{"nfInstanceId":"fe7d992b-0541-4c7d-ab84-
c6d70b1b01b1","nfType":"NWDAF","nfStatus":"REGISTERED","plmnList":null,"nsiLis
t":null,"fqdn":"ocn-nwdaf-
communication","interPlmnFqdn":null,"ipv4Addresses":null,"ipv6Addresses":null,
"priority":null,"capacity":null,"load":null,"locality":null,"udrInfo":null,"ud
mInfo":null,"ausfInfo":null,"amfInfo":null,"smfInfo":null,"upfInfo":null,"pcfI
nfo":null,"bsfInfo":null,"customInfo":null,"recoveryTime":null,"nfServices":
[{"serviceInstanceId":"03063893-
cf9e-4f7a-9827-067f6fa9dd01","serviceName":"nnrf-nfmanagement","versions":
[{"apiVersionInUri":"v1","apiFullVersion":"1.2.2","expiry":"2019-08-03T18:55:0
8.871+0000"}],"scheme":"http","nfServiceStatus":"REGISTERED","fqdn":"nrf.oracl
e.com","interPlmnFqdn":null,"apiPrefix":"bsf-
service","defaultNotificationSubscriptions":null,"allowedPlmns":null,"allowedN
fTypes":["PCF","NEF",
"NWDAF"],"allowedNfDomains":null,"allowedNssais":null,"priority":null,"capacit
y":null,"load":null,"recoveryTime":null,"supportedFeatures":null}],"sNssais":n
ull}]
         enableF3=true
         enableF5=true
         registrationRetryInterval=5000
```

```
            subscriptionRetryInterval=5000
            enableDiscoveryRefresh=false
            discoveryRetryInterval=5000
            discoveryRefreshInterval=10
            discoveryDurationBeforeExpiry=90
            renewalTimeBeforeExpiry=3600
            enableRediscoveryIfNoProdNFs=false

offStatesForRediscoveryIfNoProdNFs=SUSPENDED,UNDISCOVERABLE,DEREGISTERED
            validityTime=30
            enableSubscriptionAutoRenewal=true
            nfHeartbeatRate=80
            acceptAdditionalAttributes=true
            retryForCongestion=5
            supportedDataSetId=
            enableVirtualNrfResolution=false
            virtualNrfFqdn=nrf.oracle.com
            virtualNrfScheme=http
            useAlternateScpOnAlternateRouting=
            subscriberNotificationRetry=2
            requestTimeoutGracePeriod=2
            enableDiscoveryRefresh=false

nrfRetryConfig=[{"serviceRequestType":"ALL_REQUESTS","primaryNRFRetryCount":1,
"nonPrimaryNRFRetryCount":1,"alternateNRFRetryCount":-1,"errorReasonsForFailur
e":
["503","504","500","SocketTimeoutException","JsonProcessingException","Unknown
HostException","NoRouteToHostException","IOException"],"gatewayErrorCodes":
["503"],"requestTimeout":10},
{"serviceRequestType":"AUTONOMOUS_NFREGISTER","primaryNRFRetryCount":1,"nonPri
maryNRFRetryCount":1,"alternateNRFRetryCount":-1,"errorReasonsForFailure":
["503","504","500","SocketTimeoutException","JsonProcessingException","Unknown
HostException","NoRouteToHostException","IOException"],"gatewayErrorCodes":
["503"],"requestTimeout":10}]

healthCheckConfig={"healthCheckCount":-1,"healthCheckInterval":5,"requestTimeo
ut":10,"errorReasonsForFailure":
["503","504","500","SocketTimeoutException","IOException"],"gatewayErrorCodes"
:["503", "504", "500"]}

   # Deployment specific configuration for Nrf-Client Discovery Microservice
   nrf-client-nfdiscovery:
      global:
          #ephemeralStorage value in MB. Value 0 means ephemeral-storage will
not be set during deployment.
          logStorage: 0 #default calculated value 70
          crictlStorage: 0 #default calculated value 1
          ephemeralStorageLimit: 0 #default calculated value 1024
          maxUnavailable: "25%"
      configmapApplicationConfig: *configRef
      # NRF Client Microservice image name
      image: nrf-client
      # NRF Client Microservice image tag
      imageTag: '22.3.3'
      envJaegerSamplerParam: '1'
      envJaegerSamplerType: ratelimiting
```

```
      envJaegerServiceName: nrf-client-nfdiscovery
      # Resource Details
      cpuRequest: 2
      cpuLimit: 2
      memoryRequest: 1Gi
      memoryLimit: 1Gi
      # Min replicas to scale to maintain an average CPU utilization
      minReplicas: 1
      # Max replicas to scale to maintain an average CPU utilization
      maxReplicas: 2
      averageCpuUtil: 80
      type: ClusterIP
      # Set to true if the discovery results should be cached.
      cacheDiscoveryResults: false
      # Discovery Service Port
      envDiscoveryServicePort: 5910
      # Management Service Port
      envManagementServicePort: 5910
      #Restart Policy for hooks
      hookRestartPolicy: Never
      # prefix for Metrics
      metricPrefix: *metricPrefix
      # suffix for Metrics
      metricSuffix: *metricSuffix
      #The sidecar (istio url) when deployed in serviceMesh
      # Default value: http://127.0.0.1:15000/quitquitquit
      istioSidecarQuitUrl: ""
      # Default value: http://127.0.0.1:15000/ready
      istioSidecarReadyUrl: ""
      # Flag to enable service Mesh
      serviceMeshCheck: false
      # Flag to switch between "HELM" based or "REST" based nfProfile
configuration
      nfProfileConfigMode: "HELM"
      # Flag to enable/ disable the feature
      # Allowed Values: DISABLED, ENABLED, USE_GLOBAL_VALUE
      extraContainers: USE_GLOBAL_VALUE
      commonCfgClient:
         enabled: false
      commonCfgServer:
        # Do not comment this line in case you want to deploy both config-
server and APIGW in same namespace simultaneously
        # Otherwise comment this line and use 'host'
        configServerSvcName: 'common-config-server'
        host: 'common-config-server'
        port: '80'
        pollingInterval: 5000
        nfProfileUri: 'nf/nf-common-component/v1/nrf-client-nfmanagement/
nfProfileList'
      dbConfig:
        dbHost: *mySqlHostRef
        dbPort: *mySqlPortRef
        secretName: *privDbCredSecretNameRef
        dbName: *dbNameRef
        # Name of the Key configured for "DB Username" in Secret with
following name: "<dbConfig.secretName>"
```

```
        dbUNameLiteral: mysql-username
        # Name of the Key configured for "DB Password" in Secret with
following name: "<dbConfig.secretName>"
        dbPwdLiteral: mysql-password

   # Deployment specific configuration for Nrf-Client Management Microservice
   nrf-client-nfmanagement:
      global:
         #ephemeralStorage value in MB. Value 0 means ephemeral-storage will
not be set during deployment.
         logStorage: 0 #default calculated value 70
         crictlStorage: 0 #default calculated value 1
         ephemeralStorageLimit: 0 #default calculated value 1024
         minAvailable: 1
      configmapApplicationConfig: *configRef
      # NRF Client Microservice image name
      image: nrf-client
      # NRF Client Microservice image tag
      imageTag: '22.3.3'
      envJaegerSamplerParam: '1'
      envJaegerSamplerType: ratelimiting
      envJaegerServiceName: nrf-client-nfmanagement
      enablePDBSupport: false
      # Resource Details
      replicas: 1
      cpuRequest: 1
      cpuLimit: 1
      memoryRequest: 1Gi
      memoryLimit: 1Gi
      type: ClusterIP
      #Restart Policy for hooks
      hookRestartPolicy: Never
      # prefix for Metrics
      metricPrefix: *metricPrefix
      # suffix for Metrics
      metricSuffix: *metricSuffix
      #The sidecar (istio url) when deployed in serviceMesh
      # Default value: http://127.0.0.1:15000/quitquitquit
      istioSidecarQuitUrl: ""
      # Default value: http://127.0.0.1:15000/ready
      istioSidecarReadyUrl: ""
      # Flag to enable service Mesh
      serviceMeshCheck: false
      # Flag to switch between "HELM" based or "REST" based nfProfile
configuration
      nfProfileConfigMode: "HELM"
      # Flag to switch between "HELM" based or "REST" based nrfRoute
configuration
      nrfRouteConfigMode: "HELM"
      # Flag to enable/ disable the feature
      # Allowed Values: DISABLED, ENABLED, USE_GLOBAL_VALUE
      extraContainers: USE_GLOBAL_VALUE
      commonCfgClient:
         enabled: false
      commonCfgServer:
        # Do not comment this line in case you want to deploy both config-
```

```
server and APIGW in same namespace simultaneously
        # Otherwise comment this line and use 'host'
        configServerSvcName: 'common-config-server'
        host: 'common-config-server'
        port: '80'
        pollingInterval: 5000
      dbConfig:
        dbHost: *mySqlHostRef
        dbPort: *mySqlPortRef
        secretName: *privDbCredSecretNameRef
        dbName: *dbNameRef
        networkDbName: *networkDbNameRef
        # Database for leaderPod selection
        leaderPodDbName: *leaderPodDbRef
        # Name of the Key configured for "DB Username" in Secret with
following name: "<dbConfig.secretName>"
        dbUNameLiteral: mysql-username
        # Name of the Key configured for "DB Password" in Secret with
following name: "<dbConfig.secretName>"
        dbPwdLiteral: mysql-password

# Details of Config-server microservice
config-server:
   enabled: true
   image: occnp/oc-config-server
   imageTag: 22.3.2
   fullNameOverride: "config-server"
   envJaegerServiceName: pcf-config
   # This is the NfInstanceId of NF that will get deployed. This shall be
used in the profile being registered.
   nfInstanceId: 'fe7d992b-0541-4c7d-ab84-c6d70b1b01b1'
   # Mysql Config Server Databse Name
   envMysqlDatabase: ocnf_config_server
   # Replicas for Config server - This is exact value without scaling
   replicas: 1
   nodeSelectorEnabled: false
   nodeSelectorKey: zone
   nodeSelectorValue: app
   # Resource details
   resources:
    limits:
      cpu: 1
      memory: 1Gi
    requests:
      cpu: 0.5
      memory: 1Gi
   servicePcfConfig:
       type: NodePort

# Details of appinfo microservices
appinfo:
   enabled: true
   image: occnp/oc-app-info
   imageTag: 22.3.2
   pullPolicy: IfNotPresent
    # Replicas for Appinfo - This is exact value without scaling
```

```
   replicas: 1
   # Set Log Level to DEBUG. If false, Log Level shall be INFO
   debug: true
   serviceAccountName: ''

   commonCfgClient:
     enabled: false
   commonCfgServer:
   # Do not comment this line in case you want to deploy both config-server
and APIGW in same namespace simultaneously
   # Otherwise comment this line and use 'host'
     configServerSvcName: 'common-config-server'
     host: 'common-config-server'
     port: '80'
     pollingInterval: 5000
   dbConfig:
     dbHost: *mySqlHostRef
     dbPort: *mySqlPortRef
     secretName: *dbCredSecretNameRef
     dbName: *dbNameRef
     # Name of the Key configured for "DB Username" in Secret with following
name: "<dbConfig.secretName>"
     dbUNameLiteral: mysql-username
     # Name of the Key configured for "DB Password" in Secret with following
name: "<dbConfig.secretName>"
     dbPwdLiteral: mysql-password
   # Service to be monitored by appinfo
   # nFType in core_services must consist of nfType used in nrfclient profile.
   #Examples-1 NRF with all services listed
   #core_services:
   #  nrf:
   #  - "ocnrf-nfRegistration"
   #  - "ocnrf-nfSubscription"
   #Example-2 NRF without listing any services
   core_services: {}
   # Infrastructure services
   # If using occne 1.4 or if you don't want to monitor infra services such
as db-monitor service then the below mentioned
   # attribute 'infraServices' should be uncommented and empty array should
be passed as already mentioned.
   # If infraServices is not set, by default appinfo shall monitor status of
db-monitor-svc and db-replication-svc.
   #infraServices: []
```

6
Uninstalling OCNWDAF

To uninstall OCNWDAF, uninstall every Helm chart by running the following command:

> **Note**
>
> This command can be run from any path.

```
# Uninstall OCNWDAF
helm uninstall <helm-name> --namespace=<namespace-name>

# Example
helm uninstall ocn-nwdaf-analytics --namespace=ocn-nwdaf
```

6.1 Verify Uninstallation

To verify the OCNWDAF uninstallation, run the following command:

```
$ kubectl get all -n <release-namespace>
```

In case of successful uninstallation, no OCNWDAF resource is displayed in the command output.

If the command output displays the OCNWDAF resources or objects, then perform the following procedure:

1. Run the following command to delete all the objects:
   - To delete all the Kubernetes objects:

     ```
     kubectl delete all --all -n <release-namespace>
     ```

   - To delete all the configmaps:

     ```
     kubectl delete cm --all -n <release-namespace>
     ```

> **Caution**
>
> The command deletes all the Kubernetes objects of the specified namespace. RBAC resources and service accounts are automatically created before the helm installation in the same namespace, and these resources are required, then do not delete them.

2. Run the following command to delete the specific resources:

   ```
   kubectl delete <resource-type> <resource-name> -n <release-namespace>
   ```

3. Run the following command to delete the Kubernetes namespace:

   ```
   kubectl delete namespace <release-namespace>
   ```

7
Fault Recovery

This chapter explains how to perform fault recovery for OCNWDAF deployment.

7.1 Introduction

This section describes the procedures to perform disaster recovery for Oracle Communications Networks Data Analytics Function (OCNWDAF) deployment. The OCNWDAF operators can take database backup and restore it either on the same or a different cluster. The OCNWDAF stores data in the following storages:

- MySQL NDB Cluster
- Apache Kafka

> **Note**
>
> This section describes the recovery procedures to restore OCNWDAF databases only. To restore all the databases that are part of DBTier, see Oracle Communications Cloud Native Core cnDBTier Installation, Upgrade, and Fault Recovery Guide.

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

7.2 Fault Recovery Impact

This section provides an overview of Fault recovery scenarios and the impacted areas.

The following table provides information about the impacted areas during Oracle Communications Network data Analytics Function Fault recovery:

Table 7-1 Fault Recovery Impacted Area

Scenario	Requires Fault Recovery or re-install of Kafka?	Requires Fault Recovery or re-install of MySQL?	Other
Deployment Failure	No	No	Uninstall and re-install Helm.
cnDBTier Corruption	Yes	Yes	Restore MySQL from backup and ensure it is not re-installed.
When DBTier failed in a Single Site	No	Yes	Restore MySQL from backup and ensure it is not re-installed.

Table 7-1 (Cont.) Fault Recovery Impacted Area

Scenario	Requires Fault Recovery or re-install of Kafka?	Requires Fault Recovery or re-install of MySQL?	Other
Configuration Database Corruption	No	Yes	Backup and restore of configuration database is required.
Site Failure	No	No	Uninstall and re-install Helm.

7.3 Prerequisites

Ensure that the following prerequisites are met before performing the disaster recovery procedures:

- Ensure that cnDBTier is in a healthy state and available. Run the following command to check the status of cnDBTier service:

  ```
  kubectl -n <namespace> exec <management node pod> -- ndb_mgm -e show
  ```

- Enable the automatic backup on DBTier by scheduling regular backups. The regular backups help in disaster recovery with the following tasks:
  - Restore stable version of the network function databases
  - Minimize significant loss of data due to upgrades or roll back failures
  - Minimize loss of data due to system failure
  - Minimize loss of data due to data corruption or deletion due to external input
  - Migrate network function database information from one site to another
  - Docker images used during the previous installation or upgrade must be retained in the external data source

7.4 Fault Recovery Scenarios

This chapter describes the fault recovery procedures for various scenarios.

7.4.1 Deployment Failure

This scenario describes how to recover OCNWDAF when the deployment corrupts.

To recover OCNWDAF:

1. Run the following command to uninstall OCNWDAF:

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

   Example:
   helm uninstall oc-nwdaf --namespace oc-nwdaf
   ```

For more information about uninstalling OCNWDAF, see the **Uninstalling OCNWDAF** chapter in *Oracle Communication Networks Data Analytics Function Installation Guide.*

2. Install OCNWDAF as described in the **Installing OCNWDAF** chapter in *Oracle Communication Networks Data Analytics Function Installation Guide.* Use the back up of custom values file to reinstall the OCNWDAF.

Restore OCNWDAF, cnDBTier, and OCNWDAF database (DB) as described in Restoring OCNWDAF and cnDBTier.

7.4.2 cnDBTier Corruption

This section describes how to recover database when the data replication is broken due to database corruption and cnDBTier has failed in single site.

When the database corrupts, the database on all the other sites may also corrupt due to data replication. It depends on the replication status after the corruption has occurred. If the data replication is broken due to database corruption, then DBTier fails in either single or multiple sites (not all sites). And if the data replication is successful, then database corruption replicates to all the DBTier sites and DBTier fails in all sites.

If corrupted database replicated to mated sites, refer to the procedure When DBTier failed in a Single Site.

7.4.3 When DBTier failed in a Single Site

This section describes how to recover database when the data replication is broken due to database corruption and DBTier has failed in a single site.

To recover database:

1. Uninstall OCNWDAF helm chart. For information about uninstalling OCNWDAF, see the **Uninstalling OCNWDAF** chapter in the *Oracle Communications Networks Data Analytics Function Installation Guide*.
2. For DBTier disaster recovery:
   a. Create on-demand backup from mated site that has health replication with failed site. For more information about DBTier backup, see the **Create On-demand Database Backup** chapter in the *Oracle Communications Cloud Native Core DBTier Disaster Recovery Guide.*
   b. Use the backup data from mate site for restore. For more information about DBTier restore, see the **Restore Georeplication Failure** chapter in *Oracle Communications Cloud Native Core DBTier Disaster Recovery Guide.*
3. Install OCNWDAF helm chart. For more information about installing OCNWDAF, see the **Installing OCNWDAF** chapter in the *Oracle Communication Networks Data Analytics Function Installation Guide.*

7.4.4 Configuration Database Corruption

This scenario describes how to recover OCNWDAF when its configuration database corrupts.

The configuration database is stored in a site exclusive database along with its tables. Thus, corruption of configuration database impacts only a particular site.

To recover OCNWDAF configuration database, user has to restore the DB backup.

1. Transfer the *<backup_ filename >.sql.gz* file to the SQL node where user want to restore it.

2. Log into MySQL NDB Cluster's SQL node on the new DB cluster and create a new database where the database needs to be restored.
3. For details on creating database, user and adding permissions, see **Configuring Database, Creating Users, and Granting Permissions** section in the **Preinstallation** Chapter in *Oracle Communications Networks Data Analytics Function Installation Guide*.

> **Note**
>
> The database name created in the above step should be same as the database name created in the following step.

4. To restore the database to the new database created use the following command:

```
gunzip < <backup_filename>.sql.gz | mysql -h127.0.0.1 –u <username> -p
<backup-database-name>
```

Enter the password when prompted.

Example:

```
gunzip < OCNWDAFdbBackup.sql.gz | mysql -h127.0.0.1 -u dbuser -p OCNWDAFdb
```

7.4.5 Site Failure

This section describes how to perform disaster recovery when the site has a software failure.

It is assumed that the user has DBTier and OCNWDAFinstalled on multiple sites with automatic data replication and backup enabled.

To recover the failed sites:

1. Run the Cloud Native Environment (CNE) installation procedure to install a new cluster. For more information, see *Oracle Communications Cloud Native Environment (OCCNE) Installation Guide*.
2. For DBTier disaster recovery:
   a. Take on-demand backup from the mate site that has health replication with the failed site or sites. For more information about on-demand backup, see the **Create On-demand Database** Backup chapter in the *Oracle Communications Cloud Native Core DBTier Disaster Recovery Guide.*
   b. Use the backup data from the mate site to restore the database. For more information about database restore, see the **Restore Georeplication Failure chapter** in the *Oracle Communications Cloud Native Core DBTier Disaster Recovery Guide.*
3. Install OCNWDAF helm chart. For more information about installing OCNWDAF, see the **Installing OCNWDAF**chapter in the *Oracle Communication Networks Data Analytics Function Installation Guide.*

7.4.6 Kafka Issues

Kafka related issues (and solution) are described in the following sections.

Scenario 1

OCNWDAFmicroservices need to consume Kafka messages again from the beginning. This scenario can occur if any of the OCNWDAF microservices stopped consuming messages or failed to process them correctly. For example, when a DB downtime occurs, none of the messages consumed by the microservices will be processed correctly and we need to "retry" them.

Current Kafka configuration in OCNWDAFis based on transactions. Restart the microservice that needs to consume such messages and the messages from the last offset in Kafka are consumed. After restarting the service, the messages backlog is not consumed, change the microservice configuration stored in the Spring Cloud Configuration.

Solution

1. Ensure the microservice consumer configuration is `auto-offset-reset: earliest`. This will process the topic partition at the earliest offset (current setup value).
2. Redeploy the microservice.

Scenario 2

OCNWDAFmicroservices must ignore all messages in Kafka. This scenario can occur when the data sources like AMF, SMF, UDM, and so on are generating wrong data and overloading OCNWDAF.

Solution 1

Delete all messages in the topic, follow the steps below:

- Delete and recreate the topic.

OR

- Update the retention Kafka retention policy
  1. Stop the consumer microservice.
  2. Check the current retention policy, run the command:

     ```
     kafka-configs --zookeeper <zkhost>:2181 --describe --entity-type topics
     --entity-name <topic name>
     ```

  3. Update the retention policy in Kafka to a very low value (for example,1 min per instance):

     ```
     kafka-configs.sh --zookeeper <zkhost>:2181 --entity-type topics --alter
     --entity-name <topic name> --add-config retention.ms=60000
     ```

  4. Wait for Kafka service to delete the messages.
  5. Set the retention policy value.
  6. Redeploy the microservice.

Solution 2

Follow the procedure below, to retain the wrong messages and allow the Kafka service to delete the messages according to the retention policy.

1. Change Kafka consumer configuration to `auto-offset-reset: latest`. This will process the messages from the latest partition offset.

2. Redeploy the microservice.

7.4.7 Microservice Deployment Issue

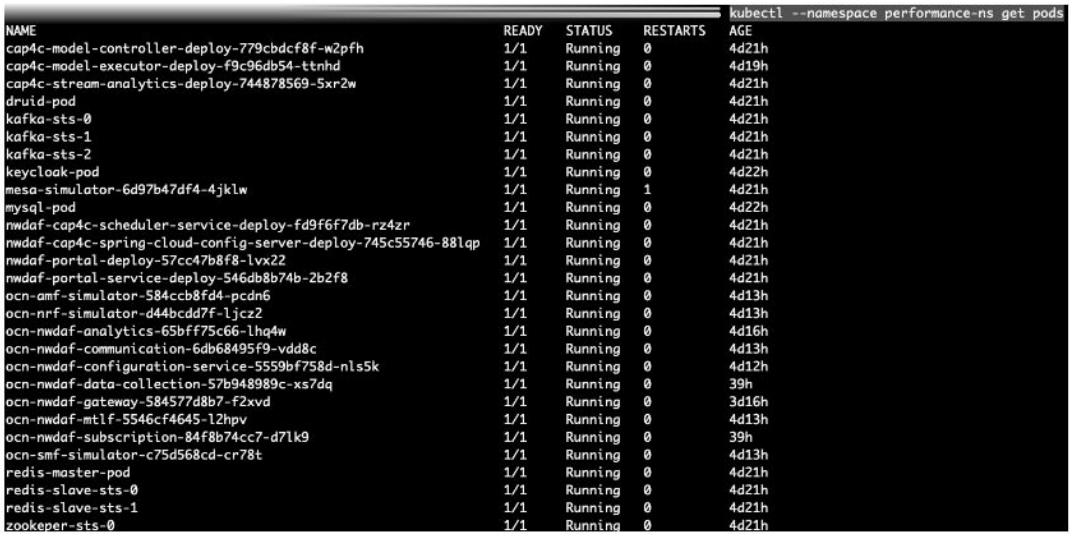
Follow the procedure below to resolve microservice deployment issue:

1. Run the following command to obtain information on the pods deployed and their status:

   ```
   kubectl --namespace [namespace name] get pods --sort-
   by='.status.containerStatuses[0].restartCount'
   ```

   Sample Output:

   Figure 7-1 Sample Output



```
kubectl --namespace performance-ns get pods
NAME                                                  READY   STATUS    RESTARTS   AGE
cap4c-model-controller-deploy-779cbdcf8f-w2pfh        1/1     Running   0          4d21h
cap4c-model-executor-deploy-f9c96db54-ttnhd           1/1     Running   0          4d19h
cap4c-stream-analytics-deploy-744878569-5xr2w         1/1     Running   0          4d21h
druid-pod                                             1/1     Running   0          4d21h
kafka-sts-0                                           1/1     Running   0          4d21h
kafka-sts-1                                           1/1     Running   0          4d21h
kafka-sts-2                                           1/1     Running   0          4d21h
keycloak-pod                                          1/1     Running   0          4d22h
mesa-simulator-6d97b47df4-4jklw                       1/1     Running   1          4d21h
mysql-pod                                             1/1     Running   0          4d22h
nwdaf-cap4c-scheduler-service-deploy-fd9f6f7db-rz4zr  1/1     Running   0          4d21h
nwdaf-cap4c-spring-cloud-config-server-deploy-745c55746-88lqp  1/1     Running   0          4d21h
nwdaf-portal-deploy-57cc47b8f8-lvx22                  1/1     Running   0          4d21h
nwdaf-portal-service-deploy-546db8b74b-2b2f8          1/1     Running   0          4d21h
ocn-amf-simulator-584ccb8fd4-pcdn6                    1/1     Running   0          4d13h
ocn-nrf-simulator-d44bcdd7f-ljcz2                     1/1     Running   0          4d13h
ocn-nwdaf-analytics-65bff75c66-lhq4w                  1/1     Running   0          4d16h
ocn-nwdaf-communication-6db68495f9-vdd8c              1/1     Running   0          4d13h
ocn-nwdaf-configuration-service-5559bf758d-nls5k      1/1     Running   0          4d12h
ocn-nwdaf-data-collection-57b948989c-xs7dq            1/1     Running   0          39h
ocn-nwdaf-gateway-584577d8b7-f2xvd                    1/1     Running   0          3d16h
ocn-nwdaf-mtlf-5546cf4645-l2hpv                       1/1     Running   0          4d13h
ocn-nwdaf-subscription-84f8b74cc7-d7lk9               1/1     Running   0          39h
ocn-smf-simulator-c75d568cd-cr78t                     1/1     Running   0          4d13h
redis-master-pod                                      1/1     Running   0          4d21h
redis-slave-sts-0                                     1/1     Running   0          4d21h
redis-slave-sts-1                                     1/1     Running   0          4d21h
zookeper-sts-0                                        1/1     Running   0          4d21h
```

2. Observe the **Restarts** column, the ideal value is "0", indicating the pods have not re-started. If the column **Status** is not **Running** and the value of the **Restarts** is constantly increasing implies an underlying problem with the microservice or any dependency.
3. Refer to the *Oracle Communications Networks Data Analytics Troubleshooting Guide* to resolve the issue.
4. If required, re-deploy the microservice.

In a Kubernetes setup, at least one pod of each microservice must be deployed so, if the problem root cause is due to a dependency, re-deploying the microservice is not required. It is sufficient to fix the dependency and wait for Kubernetes to retry the deployment.

If there are any changes in the Spring Cloud Config properties of the microservice, refreshing the property through its actuator with a POST request to `http://{host}:{port}/actuator/refresh`. If the issue is not resolved (not all properties support online refresh), redeploy the microservice.

7.5 Backup and Restore

This chapter describes the Backup and Restore procedures.

7.5.1 OCNWDAF Database Backup and Restore

Introduction

Perform this procedure to take a backup of the OCNWDAF database (DB) and restore the database on a different cluster. This procedure is for on-demand backup and restore of OCNWDAF DB. The commands used for these procedures are provided by the MYSQL Network Database(NDB) cluster.

Prerequisite

Ensure that the MYSQL NDB cluster is in a healthy state, and each of its database node is in running state. Run the following command to check the status of cnDBTier service:

```
kubectl -n <namespace> exec <management node pod> -- ndb_mgm -e show
```

Where,

- `<namespace>` is the namespace where cnDBTier is deployed
- `<management node pod>` is the management node pod of cnDBTier

Example:

```
[cloud-user@vcne2-bastion-1 ~]$ kubectl -n nwdafsvc exec ndbmgmd-0 -- ndb_mgm
-e show
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @10.233.86.202  (mysql-8.0.22 ndb-8.0.22, Nodegroup: 0, *)
id=2    @10.233.81.144  (mysql-8.0.22 ndb-8.0.22, Nodegroup: 0)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.81.154  (mysql-8.0.22 ndb-8.0.22)
id=50   @10.233.86.2  (mysql-8.0.22 ndb-8.0.22)

[mysqld(API)]   2 node(s)
id=56   @10.233.81.164  (mysql-8.0.22 ndb-8.0.22)
id=57   @10.233.96.39  (mysql-8.0.22 ndb-8.0.22)

[cloud-user@vcne2-bastion-1 ~]$
```

OCNWDAF DB Backup

If the OCNWDAF database backup is required, do the following:

1. Log in to any of the SQL node or API node, and then run the following command to take dump of the database:

   ```
   kubectl exec -it <sql node> -n <namespace> bash
   mysqldump --quick -h127.0.0.1 –u <username> -p  <databasename>| gzip >
   <backup_filename>.sql.gz
   ```

   Where,

- <sql node> is the SQL node of cnDBTier
- <namespace> is the namespace where cnDBTier is deployed
- <username> is the database username
- <databasename> is the name of the database that has to be backed up
- <backup_filename> is the name of the backup dump file

2. Enter the OCNWDAF database name and password in the command when prompted. Example:

```
kubectl exec -it ndbmysqld-0 -n nwdaf bash
mysqldump --quick -h127.0.0.1 -u dbuser -p nwdafdb | gzip >
NWDAFdbBackup.sql.gz
```

> **Note**
>
> Ensure that there is enough space on the directory to save the backup file.

OCNWDAF RESTORE

If OCNWDAF database restore is required, do the following:

1. Transfer the <backup_ filename>.sql.gz file to the SQL node where you want to restore it.
2. Log in to the SQL node of the MYSQL NDB cluster on the new DB cluster and create a new database where the database needs to be restored
3. Create database, database user, and grant permissions as described in *Oracle Communication Networks Data Analytics Function Installation Guide*

> **Note**
>
> The database name created in this step should be the same as the database name created in the next sub step. Also, the Kubernetes secret should be the same as in the `values.yaml` file used for installing OCNWDAF.

4. To restore the database to the new database created, run the following command:

```
gunzip < <backup_filename>.sql.gz | mysql -h127.0.0.1 –u <username> -p
<databaseName >
```

Example:

```
gunzip < nwdafdbBackup.sql.gz | mysql -h127.0.0.1 -u dbuser -p newnwdafdb
```

5. Enter the password when prompted.

7.5.2 Restoring OCNWDAF and cnDBTier

Perform this procedure to restore OCNWDAF, cnDBTier, and OCNWDAF database (DB).

Prerequisites:

- Take a backup of the `custom_values.yaml` file that was used for installing OCNWDAF.
- Take a backup of the OCNWDAFdatabase and restore the database as described in Restoring OCNWDAF and cnDBTier Perform this task once every evening.

If OCNWDAFdeployment is corrupted, do the following:

1. Run the following command to uninstall the corrupted OCNWDAFdeployment:

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

   Where,

   - `<release_name>` is a name used to track this installation instance.
   - `<namespace>` is the release number.

   Example:

   ```
   helm uninstall ocnwdaf --namespace nwdafsvc
   ```

2. Install OCNWDAFusing the backed up copy of the `custom_values.yaml` file. For information about installing OCNWDAFusing Helm, see *Oracle Communication Networks Data Analytics Function Installation Guide*

If cnDBTier and NF deployment is corrupted, do the following:

1. Install cnDBTier as described in *Oracle Communication Cloud Native Core DBTier Installation and Upgrade Guide*
2. Restore OCNWDAFDB as described in Restoring OCNWDAF and cnDBTier
3. Install OCNWDAFusing the backed up copy of the `custom_values.yaml` file. For information about installing OCNWDAF, see *Oracle Communication Networks Data Analytics Function Installation Guide*

If DB Secret or cnDBTier mysql FQDN or IP or PORT is changed, do the following:

1. In the backed up copy of the `custom_values.yaml` file, update these values: `dbHost/dbPort/dbAppUserSecretName/dbPrivilegedUserSecretName`.

   ```
   # DB Connection Service IP Or Hostname with secret name.
     database:
       dbHost: "mysql-connectivity-service.nwdafsvc"
       dbPort: "3306"
       # K8s Secret containing Database/user/password for all services of
   OCNWDAF interacting with DB.
       dbAppUserSecretName: "seppuser-secret"
       # K8s Secret containing Database/user/password for DB Hooks for
   creating tables
       dbPrivilegedUserSecretName: "nwdafprivilegeduser-secret"
   ```

2. Run the following Helm upgrade command to update the secrets for OCNWDAF control plane microservices, such as configuration, subscription, audit, and notification, to connect to the DB with changed DB Secret or DB FQDN, IP, or Port:

   ```
   helm upgrade <release name> -f <custom_values.yaml> --namespace
   <namespace> <helm-repo>/chart_name --version <helm_version>
   ```

Example:

```
helm upgrade ocnwdaf -f /home/cloud-user/1.12/values.yaml /home/cloud-user/
1.11.0/ocnwdaf --namespace nwdafsvc
```

7.5.3 Kafka Backup and Restore

Follow the backup procedures described below:

- Backup Zookeeper State Data
- Backup Kafka Topics and Messages

Once the backup procedures are completed, follow the restore procedures:

- Restore Zookeeper State Data
- Restore Kafka Data and Messages

Finally, follow the procedure to Verify the Restoration.

Backup Zookeeper State Data

The Zookeeper stores its data in the directory specified by the `dataDir` field in the `~/kafka/config/zookeeper.properties` configuration file. Read the value of this field to determine the directory to backup. By default, `dataDir` points to the `/tmp/zookeeper` directory.

> **Note**
>
> If the `dataDir` points to some other directory in your installation, use that value instead of `/tmp/zookeeper` in the following commands.

1. Run the command `~/kafka/config/zookeeper.properties`.

   Sample output:

   ```
   # the directory where the snapshot is stored.
   dataDir=/tmp/zookeeper
   # the port at which the clients will connect
   clientPort=2181
   # disable the per-ip limit on the number of connections since this is a
   non-production config
   maxClientCnxns=0
   ```

2. Create a compressed archive file of the contents in the directory. Run the command:

   ```
   tar -czf /home/kafka/zookeeper-backup.tar.gz /tmp/zookeeper/*
   ```

3. Run the command `ls`, to verify if the `tar` file is created. The output will display the compressed file `zookeeper-backup.tar.gz`.
4. A backup of the Zookeeper State Data is successfully created.

Backup Kafka Topics and Messages

Kafka stores topics, messages, and internal files in the directory that the `log.dirs` field specifies in the `~/kafka/config/server.properties` configuration file. Read the value of this

field to determine the directory to backup. By default, `log.dirs` points to the `/tmp/kafka-logs` directory.

> **Note**
>
> If the `log.dirs` points to some other directory in your installation, use that value instead of `/tmp/kafka-logs` in the following commands.

1. Run the command `~/kafka/config/server.properties`.

   Sample output:

   ```
   ############################# Log Basics #############################

   # A comma separated list of directories under which to store log files
   log.dirs=/tmp/kafka-logs

   # The default number of log partitions per topic. More partitions allow
   greater
   # parallelism for consumption, but this will also result in more files
   across
   # the brokers.
   num.partitions=1

   # The number of threads per data directory to be used for log recovery at
   startup and flushing at shutdown.
   # This value is recommended to be increased for installations with data
   dirs located in RAID array.
   num.recovery.threads.per.data.dir=1
   ```

2. Stop the Kafka service so that the data in the `log.dirs` is in a consistent state. Run `exit` to become a non-root user, then run the command:

   ```
   sudo systemctl stop kafka
   ```

3. Once the Kafka service is stopped, login as Kafka root user. Run the command:

   ```
   sudo -iu kafka
   ```

   > **Note**
   >
   > Always stop or start the Kafka and Zoo Keeper services as non-root sudo user, as in the Apache Kafka installation prerequisite the **kafka** user is restricted as a security precaution. This prerequisite disables sudo access for the **kafka** user, hence commands fail to execute.

4. Create a compressed archive file of the contents in the directory. Run the command:

   ```
   tar -czf /home/kafka/kafka-backup.tar.gz /tmp/kafka-logs/*
   ```

5. Run the command `ls`, to verify if the `tar` file is created. The output will display the compressed file `kafka-backup.tar.gz`.

6. Start the Kafka service, run `exit` to become a non-root user, then run the command:

   ```
   sudo systemctl start kafka
   ```

7. Once the Kafka service is started, login as Kafka root user. Run the command:

   ```
   sudo -iu kafka
   ```

8. A backup of the Kafka Data is successfully created.

Restore Zookeeper State Data

1. To prevent data directories from receiving invalid data during the restoration period, stop both Zookeeper and Kafka. Run the following commands:
   - Run `exit` to become a non-root user, then run the command:

     ```
     sudo systemctl stop kafka
     ```

   - Stop the Zookeeper service:

     ```
     sudo systemctl stop zookeeper
     ```

2. Login as Kafka root user. Run the command:

   ```
   sudo -iu kafka
   ```

3. Delete the existing cluster data directory, run the command:

   ```
   rm -r /tmp/zookeeper/*
   ```

4. Restore the backedup data, run the command:

   ```
   tar -C /tmp/zookeeper -xzf /home/kafka/zookeeper-backup.tar.gz --strip-components 2
   ```

   > **Note**
   >
   > The `-C` flag specifies the `tar` to change to the directory `/tmp/zookeeper` before extracting the data. Specifying `--strip 2` flag in the command ensures `tar` extracts the archive's contents in the directory `/tmp/zookeeper/` and not in any another directory (such as `/tmp/zookeeper/tmp/zookeeper/`) inside it.

5. Cluster state data is successfully restored.

Restore Kafka Data and Messages

1. Delete the existing Kafka directory, run the command:

   ```
   rm -r /tmp/kafka-logs/*
   ```

2. After directory deletion, the Kafka installation is similar to a fresh installation with no topics or messages present in it. Restore the backed up data, extract the files:

   ```
   tar -C /tmp/kafka-logs -xzf /home/kafka/kafka-backup.tar.gz --strip-components 2
   ```

   The `-C` flag specifies `tar` to change to the directory `/tmp/kafka-logs` before extracting the data. Specifying `--strip 2` flag ensures that the archive's contents are extracted in `/tmp/kafka-logs/` and not in another directory (such as `/tmp/kafka-logs/kafka-logs/`) inside it.

3. Run `exit` to become a non-root user, then run the command:

   ```
   sudo systemctl start kafka
   ```

   Start the Zookeeper service

   ```
   sudo systemctl start zookeeper
   ```

4. Login as Kafka root user. Run the command:

   ```
   sudo -iu kafka
   ```

5. Kafka data is successfully restored.

Verify the Restoration

1. Once Kafka is successfully running, run the following command to read the messages:

   ```
   ~/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic <TopicName> --from-beginning
   ```

   > **Note**
   >
   > If Kafka has not started up properly, you might encounter the following warning:
   >
   > ```
   > Output
   > [2018-09-13 15:52:45,234] WARN [Consumer clientId=consumer-1, groupId=console-consumer-87747] Connection to node -1 could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
   > ```

   Always wait for Kafka to start fully before performing this step.

2. Retry the previous step if you encountered a warning, or run the following command (as non-root sudo user) to restart Kafka:

   ```
   sudo systemctl restart kafka
   ```

3. The following message is displayed if restoration is successful:

```
Output
<Topic Message>
```

> **Note**
>
> If you do not see this message, check if you missed out procedure steps in the previous section and run them.

A successful verification implies, data has been backed up and restored correctly in a single Kafka installation.