

Oracle® Communications

Networks Data Analytics Function Installation and Fault Recovery Guide



Release 23.3.0

F84697-02

November 2023

ORACLE®

Copyright © 2022, 2023, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

1	Introduction	
1.1	Overview	1
1.2	References	2
2	Installing OCNWDAF	
2.1	Prerequisites	1
2.1.1	Software Requirements	1
2.1.2	Environment Setup Requirements	3
2.1.3	Resource Requirements	4
2.2	Installation Sequence	6
2.2.1	Preinstallation Tasks	6
2.2.1.1	Downloading the OCNWDAF package	6
2.2.1.2	Pushing the Images to Customer Docker Registry	8
2.2.1.3	Verifying and Creating OCNWDAF Namespace	11
2.2.2	Installation Tasks	11
2.2.2.1	Update OCNWDAF Preinstaller Files	12
2.2.2.2	Setup Encrypted Credentials	12
2.2.2.3	Configure Database Flag	12
2.2.2.4	Configuring Service Mesh	13
2.2.2.5	Configuring Routing Rules in Ingress Gateway	15
2.2.2.6	Install Ingress and Egress Gateways	19
2.2.2.7	Installing OCNWDAF Package	19
2.2.3	Postinstallation Tasks	26
2.2.3.1	Verifying Installation	26
2.2.3.2	Performing Helm Test	27
2.2.3.3	Configuring OCNWDAF GUI	28
2.2.3.4	Taking a Backup	38
3	OCNWDAF Customization	
3.1	Configurable Parameters	1
3.1.1	Global Parameters	2
3.1.2	Microservice Parameters	3

3.1.3	Georedundancy Parameters	4
3.1.4	NRF Client Parameters	7
3.1.5	Ingress Gateway Parameters	25

4 Uninstalling OCNWDAF

4.1	Verify Uninstallation	1
-----	-----------------------	---

5 Fault Recovery

5.1	Introduction	1
5.2	Fault Recovery Impact	1
5.3	Prerequisites	2
5.4	Fault Recovery Scenarios	2
5.4.1	Deployment Failure	2
5.4.2	cnDBTier Corruption	3
5.4.3	When cnDBTier failed in a Single Site	3
5.4.4	Configuration Database Corruption	3
5.4.5	Site Failure	4
5.4.6	Kafka Issues	4
5.4.7	Microservice Deployment Issue	5
5.5	Backup and Restore	6
5.5.1	OCNWDAF Database Backup and Restore	6
5.5.2	Restoring OCNWDAF and cnDBTier	8
5.5.3	Kafka Backup and Restore	9

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
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
NF	Network Function
NRF	Oracle Communications Cloud Native Core, Network Repository Function
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
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 23.3.x in Oracle Communications Network Data Analytics Function (OCNWDAF) Installation and Fault Recovery Guide.

Release 23.3.0.0.1 - F84697-02, November 2023

The following sections are updated for this release:

- [Configuring OCNWDAF GUI](#)
- Added note in the section [Pushing the Images to Customer Docker Registry](#)

Release 23.3.0 - F84697-01, September 2023

The following sections are updated for this release:

- [Software Requirements](#): Updated with the latest software versions supported by OCNWDAF.
- [Environment Setup Requirements](#): Updated the Environment setup requirements for 23.3.0 OCNWDAF.
- [Downloading the OCNWDAF package](#): Release version updated.
- [Pushing the Images to Customer Docker Registry](#)
- [Installing OCNWDAF Package](#): Docker image versions updated.
- [Verifying and Creating OCNWDAF Namespace](#): Sample command added.

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.
- 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.

Note

The performance and capacity of the OCNWDAF system may vary based on the call model, Feature/Interface configuration, and underlying CNE and hardware environment.

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 Configuration Console Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Environment Installation, Upgrade, 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

Installing OCNWDAF

This chapter provides information about installing Oracle Communications Networks Data Analytics Function (OCNWDAF) in a cloud native environment.

2.1 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.

2.1.1 Software Requirements

This section describes the software requirements for installing OCNWDAF:

Oracle Communications Cloud Native Environment Specification

Oracle Communications Network Data Analytics Function (OCNWDAF) 23.3.0 can be installed on Oracle Cloud Infrastructure (OCI), Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) 1.9.x, 1.10.x, 22.4.x, 23.1.x, and 23.2.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 Core, Cloud Native Environment (CNE) 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, 1.23.x, 1.24.x
HELM	3.1.2, 3.5.0, 3.6.3, 3.8.0
Podman	2.2.1, 3.2.3, 3.3.1
cnDBTier	23.2.0, 23.2.1, 22.4.1
CNC Console	23.3.0, 23.2.0, 23.2.1

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

- To check Kubernetes version, run the following command:

```
kubect1 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	9.1.7	KPIs
prometheus	2.39.1	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.

2.1.2 Environment Setup Requirements

This section provides information about 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 about installation procedure, see the *Oracle Communications Cloud Native Core cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

If the environment has cnDBTier 23.2.0 installation, follow the instruction below:

- If cnDBTier 23.2.0 release is installed, set the `ndb_allow_copying_alter_table` parameter to 'ON' in the cnDBTier custom values `dbtier_23.2.0_custom_values_23.2.0.yaml` file and perform cnDBTier upgrade before install, upgrade, or any fault recovery procedure is performed for OCNWDAF. Set the parameter to its default value, 'OFF' once the activity is completed and perform the cnDBTier upgrade to apply the parameter changes.
- To perform cnDBTier upgrade, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

2.1.3 Resource Requirements

This section lists the resource requirements to install and run OCNWDAF.

OCNWDAF Services

The following table lists the resource requirement for OCNWDAF services:

Table 2-3 Core Microservices Resource Requirements

Microservice Name	POD Replica		CPU/POD		Memory/POD (in GB)		Ephemeral Storage	
	Min	Max	Min	Max	Min	Max	Min (Mi)	Max (GB)
ocn-nwdaf-analytics-info-service	1	2	1	2	1	2	78.1	1
nwdaf-ingress-gateway	1	2	1	2	1	2	78.1	1
nwdaf-cap4c-spring-cloud-config-server	1	1	2	2	1	1	78.1	1
nwdaf-egress-gateway	1	2	1	2	1	2	78.1	1
ocn-nwdaf-data-collection-service	2	4	2	4	2	4	78.1	1
ocn-nwdaf-data-collection-controller	2	1	2	2	1	1	78.1	1
ocn-nwdaf-subscription-service	1	2	1	2	1	2	78.1	1
ocn-nwdaf-mtlf-service	1	2	1	2	1	2	78.1	1
cap4c-configuration-manager-service	1	2	1	2	1	2	78.1	1
ocn-nwdaf-cap4c-model-controller	1	2	4	8	1	2	78.1	1

Table 2-3 (Cont.) Core Microservices Resource Requirements

Microservice Name	POD Replica		CPU/POD		Memory/POD (in GB)		Ephemeral Storage	
ocn-nwdaf-cap4c-model-executor	2	4	2	4	1	2	78.1	1
ocn-nwdaf-cap4c-stream-analytics	2	4	4	8	1	2	78.1	1
ocn-nwdaf-cap4c-portal	1	2	2	4	1	2	78.1	1
ocn-nwdaf-cap4c-portal-service	1	2	2	4	1	2	78.1	1
ocn-nwdaf-cap4c-scheduler-service	1	2	1	2	1	2	78.1	1
cap4c-stream-transformer	1	2	1	2	1	2	78.1	1
cap4c-api-gateway	1	2	1	2	1	2	78.1	1
ocn-nwdaf-cap4c-kafka-ingestor	2	4	1	2	1	2	78.1	1
ocn-nwdaf-cap4c-reporting-service	1	2	1	2	1	2	78.1	1
ocn-nwdaf-geo-redundancy-agent	1	1	1	1	1	2	78.1	1
Total	25	45	32	59	21	40		

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 run. 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 2-4 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

2.2 Installation Sequence

This section describes preinstallation, installation, and postinstallation tasks for OCNWDAF.

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

2.2.1 Preinstallation Tasks

Before installing OCNWDAF, perform the tasks described in this section.

① 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.

2.2.1.1 Downloading the OCNWDAF package

This section provides information about how to download OCNWDAF package.

To download the OCNWDAF package from [My Oracle Support](#):

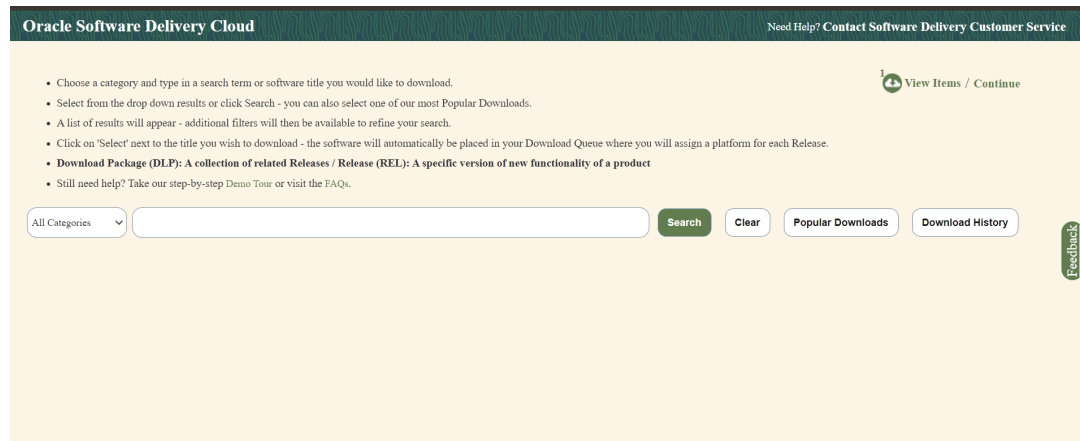
1. Log in to [My Oracle Support](#) using the appropriate credentials.
2. Click **Patches & Updates** to locate the patch.
3. In the **Patch Search** console, select the **Product or Family (Advanced)** option.
4. Enter Oracle Communications Cloud Native Core Network Data Analytics Function in the **Product** field. Select the product from the **Product** drop-down
5. From the **Release** drop-down, select "**Oracle Communications Cloud Native Core Network Data Analytics Function <release_number>**" Where, <release_number> indicates the required release number of OCNWDAF.
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-23.3.0.0.zip*

To download the package from the [edelivery](#) portal, perform the following steps:

1. Login to the [edelivery](#) portal with your credentials. The following screen appears:

Figure 2-1 edelivery portal

2. Select the **Download Package** option, from **All Categories** drop down list.
3. Enter **Oracle Communications Cloud Native Core Network Data Analytics Data Function** in the search bar.
4. List of release packages available for download are displayed on the screen. Select the release package you want to download, the package automatically gets downloaded.

Untar the Package ZIP File

Run the following command to untar or unzip the OCNWDAF package zip file to the specific repository:

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

or

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

After the package and its content is extracted, the following directory structure is displayed:

```
# Root
- images
  - tar of images
  - sha 256 of images
- troubleshooting/
  - nfDataCapture.sh
- installer/
  - ocn-nwdaf-helmChart/
    - helmChart/
      - templates
      - charts
      - values.yaml
      - charts.yaml
      - nwdaf-pre-installer.tar.gz
  - simulator-helmChart/
    - templates
    - charts
    - values.yaml
    - charts.yaml
```

- nwdaf-ats/
 - ocn-ats-nwdaf-tool
 - nginx
 - templates
 - ocnwdaf_tests
 - data
 - features
- scripts/
 - gen_certs.sh
 - readme.txt

Note

The *readme.txt* file under the scripts folder is for the *gen_certs.sh* script.

2.2.1.2 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 2-5 Docker Images for OCNWDAF

Service Name	Docker Image Name	Image Tag
NWDAF Analytics Info Service	ocn-nwdaf-analytics	23.3.0.0.0
NWDAF MTLF Service	ocn-nwdaf-mtlf-service	23.3.0.0.0
NWDAF Subscription Service	ocn-nwdaf-subscription-service	23.3.0.0.0
AMF NF Simulator Service	ocn-amf-simulator-service	23.3.0.0.0
SMF NF Simulator Service	ocn-smf-simulator-service	23.3.0.0.0
NRF NF Simulator Service	ocn-nrf-simulator-service	23.3.0.0.0
OAM Simulator Service	ocn-oam-simulator-service	23.3.0.0.0
Mesa Simulator Service (Data Generator)	mesa-simulator	23.3.0.0.0
cap4c ML model controller	cap4c-model-controller	23.3.0.0.0
cap4c ML model executor	cap4c-model-executor	23.3.0.0.0
cap4c stream analytics	cap4c-stream-analytics	23.3.0.0.0
kafka to mysql serializer	cap4c-kafka-ingestor	23.3.0.0.0
Reporting service	nwdaf-cap4c-reporting-service	23.3.0.0.0
kafka	nwdaf-cap4c-kafka	3.4.0
nwdaf-cap4c-scheduler	nwdaf-cap4c-scheduler-service	23.3.0.0.0
nwdaf-cap4c-spring-cloud-config-server	nwdaf-cap4c-spring-cloud-config-server	23.3.0.0.0
nwdaf-portal	nwdaf-portal	23.3.0.0.0
nwdaf-portal-service	nwdaf-portal-service	23.3.0.0.0
redis	nwdaf-cap4c-redis	7.0.4
zookeeper	nwdaf-cap4c-zookeeper	3.8.0
nwdaf-cap4c-initial-setup-script	nwdaf-cap4c-initial-setup-script	23.2.0
ocats-nwdaf	ocats-nwdaf	23.3.0.0.0

Table 2-5 (Cont.) Docker Images for OCNWDAF

Service Name	Docker Image Name	Image Tag
ocats-nwdaf-notify	ocats-nwdaf-notify	23.3.0.0.0
ocats-nwdaf-notify-nginx	ocats-nwdaf-notify-nginx	23.3.0.0.0
Helm Test	nf-test	22.2.0
geo redundancy agent	ocn-nwdaf-geo-redundacy-agent	23.1.0.0.0
nwdaf-egress-gateway	ocingress_gateway	23.1.3
nwdaf-ingress-gateway	ocegress_gateway	23.1.3
nrf client configuration server	oc-config-server	22.4.0
nrf client app info	oc-app-info	22.4.0
nrf client perf info	oc-perf-info	22.4.0
nrf client	nrf-client	22.4.0
NWDAF Data Collection Controller	ocn-nwdaf-data-collection-controller	23.3.0.0.0
cap4c-configuration-manager-service	cap4c-configuration-manager-service	23.3.0.0.0
cap4c-stream-transformer	cap4c-stream-transformer	23.3.0.0.0
nwdaf-cap4c-nginx	nwdaf-cap4c-nginx	1.20
cap4c-api-gateway	cap4c-api-gateway	23.3.0.0.0
nwdaf-cap4c-java	nwdaf-cap4c-java	17.0
ocnwdaf-pre-install-hook-image	ocnwdaf-pre-install-hook-image	1.1.3
nwdaf-cap4c-mysql	nwdaf-cap4c-mysql	8.0.30

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. If the images of the above services are already present in the artifact, then proceed with the [Preinstallation Tasks](#) tasks.
3. (Optional) If the images of the above services are not present in the artifact, then the user has to run the following commands to manually load, tag, and push the images. Run the following command:

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

Example:

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

Note

If you face issues in loading the images, verify if format of images is correct. For example:

The format is `cap4c-api-gateway-23.3.0.0.1.tar` and not `cap4c-api-gateway:23.3.0.0.1.tar`

Run the following command to change the format:

```
mv <source image name> <destination image name>
```

For example:

```
mv cap4c-api-gateway:23.3.0.0.1.tar cap4c-api-gateway-23.3.0.0.1.tar
```

4. 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 through HTTPS, otherwise, docker push command may fail.

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

```
docker images
```

6. (Optional) Push the Helm charts to the Helm repository. Run the following command:

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

Untar the Preinstaller

(Optional) To extract the `nwdaf-pre-installer.tar.gz` file outside the `/helmchart` directory, run the following command:

```
tar xzC <path to extract> -f nwdaf-pre-installer.tar.gz
```

Verify the file structure of the extracted file:

- etc/
 - nwdaf-cap4c-spring-cloud-config-prod-properties/
 - kafka-topics.txt
- scripts/
 - util/

- kubernetes-util.sh
- helm-util.sh
- generic-util.sh
- prepare-dependencies.sh

2.2.1.3 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 the namespace is not available, create the namespace using the following command:

```
$ 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 namespace:

- 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.

To export the installation *namespace* name as environment variable, run the following command:

```
export K8_NAMESPACE="<namespace>"
```

2.2.2 Installation Tasks

This section explains how to install OCNWDAF.

Note

Before installing OCNWDAF, you must complete [Prerequisites](#) and [Preinstallation Tasks](#).

2.2.2.1 Update OCNWDAF Preinstaller Files

Note

This is an optional procedure.

To update the preinstaller file, perform the following steps:

1. Make the required changes in *config* files present in the extracted *nwdaf-pre-installer* directory and create a fresh tar file by running the following command:

```
tar -zcvf nwdaf-pre-installer.tar.gz nwdaf-pre-installer/
```

2. Replace the existing tar file in the */helmChart* directory with the new tar file.

2.2.2.2 Setup Encrypted Credentials

To set up encrypted credentials, perform the following steps:

1. To update the secret values (username and password), replace the existing values with updated values after encoding the values using Base64 encoding method. Listed below are the secrets files:
 - *ocnwdafe-hooks-secret.yaml* under */helmchart/templates/* directory
 - *simulators-hooks-secret.yaml* under */simulator-helmChart/templates/* directory
2. To read the secret values, decode the present values using Base64 decoding method.

2.2.2.3 Configure Database Flag

Note

This is an optional step. Perform this step based on customer requirement.

Update the *dbConfigStatus* flag in *values.yaml* file under */helmchart* with any of the following values (the default value is *alldb*):

- *alldb*: This is the default value of the flag. Set this flag to create a fresh database by removing the existing database. If this flag is present, proceed with the installation of the services.
- *nodb*: This flag disables the *dbCreation* hooks for the installation of the Helm chart. Set this flag to install the services if the database is present without deleting any data.

- *nwdafdb*: This flag is used to create or reinstall the database only for OCNWDAF services. Set this flag to run the *dbCreation* hook only for OCNWDAF services (standard installation is followed for the remaining services).
- *cap4cdb*: This flag is used to create or reinstall the database only for CAP4C services. Set this flag to run the *dbCreation* hook only for CAP4C services (standard installation is followed for the remaining services).

① Note

If there is a requirement to install only OCNWDAF or only CAP4C services, set the *dbConfigStatus* flag to create the required DB and the charts that are not needed can be set to 'enabled: false' in the "*values.yaml*" under "*/helm chart*".

For example, if a user wants to install CAP4C services only with its database, then set the *dbConfigStatus* flag to 'cap4cdb', and set the value of all the OCNWDAF FE services that are not required to 'enabled: false' and proceed with the installation procedure.

2.2.2.4 Configuring Service Mesh

① Note

This configuration step is optional and only applies when a service mesh is available.

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

- Service discovery
- Routing and traffic configuration
- Encryption and authentication/authorization
- Metrics and monitoring

① Note

To configure OCNWDAF to support a service mesh, the service mesh must be available in the cluster in which OCNWDAF is installed.

Enable or Disable Service Mesh

To enable or disable service mesh support, update the Istio sidecar section in the *values.yaml* file.

For example:

```
#####

#ISTIO SIDECAR INJECTION #

#####

istio:

    ## NOTE: The label of the namespace will take precedence over the
    injection field that is set here. If mesh is to be disabled, make sure the
    namespace has no istio-injection label or set to disabled if present

    injection: false
    readinessCheck: &readinessCheck false
```

For more information, see [Global Parameters](#).

Update the following NRF client parameters:

- `istioSidecarQuitUrl`
- `istioSidecarReadyUrl`
- `serviceMeshCheck`

For more information, see [NRF Client Parameters](#).

Update the following Ingress Gateway Parameters in the `values.yaml` file:

- `serviceMeshCheck`

Table 2-6 Ingress Gateway Parameter

Parameter	Description	Detail
<code>serviceMeshCheck</code>	This is a mandatory parameter. This flag must be set to <code>true</code> if a Service Mesh is present in the environment where OCNWDAF is deployed. If this parameter is set to <code>true</code> load balancing is handled by the Service Mesh.	Range: True or False Default value: False Applicable to: OCNWDAF

Update the following Egress Gateway parameters in the `values.yaml` file:

- `serviceMeshCheck`

Table 2-7 Egress Gateway Parameter

Parameter	Description	Detail
serviceMeshCheck	This is a mandatory parameter. This flag must be set to <code>true</code> if a Service Mesh is present in the environment where OCNWDAF is deployed. If this parameter is set to <code>true</code> load balancing is handled by the Service Mesh.	Range: True or False Default value: False Applicable to: OCNWDAF

After Service Mesh is enabled and deployed, the proxy containers run along with the OCNWDAF application pods.

① Note

The gateways and other services inside the Service Mesh are not accessible from outside the Service Mesh. In order to use OCNWDAF with a Service Mesh, ensure that the dependencies (such as, `cnDBTier` or analytics consumers) are deployed within the Service Mesh.

2.2.2.5 Configuring Routing Rules in Ingress Gateway

The routing rules are configured in the Ingress Gateway `values.yaml` file. Once the routing rules are configured, the Ingress Gateway reroutes the incoming traffic to the microservices based on the configured routing rules.

Ingress Gateway `values.yaml` file:

```
- id: prodcon
  uri: http://10.123.158.150:31457
  path: /relinquishOwnership
  order: 1
  #Below field is used to provide an option to enable/disable route level
  xfccHeaderValidation, it will override global configuration for
  xfccHeaderValidation.enabled
  metadata:
    # requestTimeout is used to set timeout at route level. Value should be
    in milliseconds.
    requestTimeout: 4000
    requiredTime: 3000
    xfccHeaderValidation:
      validationEnabled: false
    oauthValidator:
      enabled: false
    svcName: "prodcon-1"
    configurableErrorCodes:
      enabled: false
    errorScenarios:
      - exceptionType: "NOT_FOUND_EXCEPTION"
        errorProfileName: "ERR_NOT_FOUND"
      - exceptionType: "UNKNOWN_HOST_EXCEPTION"
```

```

    errorProfileName: "ERR_UNKNOWN_HOST"
  - exceptionType: "CONNECT_EXCEPTION"
    errorProfileName: "ERR_400"
  - exceptionType: "XFCC_HEADER_NOT_PRESENT_OR_EMPTY"
    errorProfileName: "ERR_1300"
  - exceptionType: "GLOBAL_RATELIMIT"
    errorProfileName: "ERR_RATE_LIMIT"

# Server header configuration if defined at Route level(irrespective of
being enabled/disabled) will take precedence over the Global conf. Uncomment
only if needed at Route level.
#serverHeaderDetails:
#  enabled: false
#  errorCodeSeriesId: E2 # If not defined here, value at Global level
will be used as fallback. Value need to be one among "errorCodeSeriesList"
resource defined later.
filters:
  controlledShutdownFilter:
    applicableShutdownStates:
      - "PARTIAL_SHUTDOWN"
      - "COMPLETE_SHUTDOWN"
    unsupportedOperations:
      - "GET"
      - "PUT"

#Below are Request Custom Headers
customReqHeaderEntryFilter:
  headers:
    - methods:
        - ALL
      headersList:
        - headerName: x-entry-headeReq-1
          defaultVal: script:shm-02,x-exit-new-req
          source: incomingReq
          sourceHeader: x-entry-current-user
        - headerName: x-entry-current-user
          defaultVal: 123
          source: incomingReq
          sourceHeader: test
customReqHeaderExitFilter:
  headers:
    - methods:
        - ALL
      headersList:
        - headerName: x-exit-headeReq-1
          defaultVal: abc
          source: incomingReq
          sourceHeader: x-exit-current-user
        - headerName: x-exit-current-user
          defaultVal: 123
          source: incomingReq
          sourceHeader: sbi-timer-feature
    - methods:
        - GET
        - POST
      headersList:

```

```

- headerName: x-exit-headeReq-3
  defaultVal: abc
  source: incomingReq
  sourceHeader: x-exit-new-req
  override: false
- headerName: x-exit-headeReq-4
  defaultVal: 123
  source: incomingReq
  sourceHeader: x-exit-headeReq-1
  override: false
- methods:
  - DELETE
  - GET
headersList:
- headerName: x-exit-headerReq-5
  defaultVal: abc
  source: incomingReq
  sourceHeader: x-exit-headerReq-new
  override: false
- headerName: x-exit-headerReq-6
  defaultVal: 123
  source: incomingReq
  sourceHeader: x-exit-headerReq-temp
  override: false
# Below are Response Custom Headers
customResHeaderEntryFilter:
headers:
- methods:
  - ALL
headersList:
- headerName: x-entry-headerRes-1
  defaultVal: abc
  source: incomingReq
  sourceHeader: x-entry-headeReq-1
  override: false
- headerName: sbi-timer-feature-Res
  defaultVal: 123
  source: incomingReq
  sourceHeader: x-exit-new-req
customResHeaderExitFilter:
headers:
- methods:
  - ALL
headersList:
- headerName: x-exit-headerRes-1
  defaultVal: abc
  source: incomingReq
  sourceHeader: x-exit-headerReq-1
  override: false
- headerName: sbi-timer-feature
  defaultVal: 123
  source: incomingRes
  sourceHeader: x-exit-headerRes-1
- methods:
  - GET
  - PUT

```

```

      headersList:
        - headerName: x-exit-headeRes-3
          defaultVal: abc
          source: incomingRes
          sourceHeader: x-exit-SourceRes-a
          override: true
        - headerName: x-exit-headeRes-4
          defaultVal: 123
          source: incomingReq
          sourceHeader: x-exit-SourceRes-b
          override: false
    - methods:
      - DELETE
      headersList:
        - headerName: x-exit-headeRes-5
          defaultVal: abc
          source: incomingRes
          sourceHeader: ""
          override: false
        - headerName: x-exit-headeRes-6
          defaultVal: 123
          source: incomingRes
          sourceHeader: ""
          override: false
    #Below field is used for blacklisting(removing) a request header at route
    level.
    removeRequestHeader:
      - name: myheader1
      - name: myheader3
    #Below field is used for blacklisting(removing) a response header at
    route level.
    removeResponseHeader:
      - name: myresponseheader1
      - name: myresponseheader3

```

The following Ingress Gateway *routesConfig* parameters are updated:

- id: prodcon
- uri: http://10.123.158.150:31457
- path: /relinquishOwnership

For more information on the customizable Ingress Gateway parameters, see [Ingress Gateway Parameters](#).

Note

It is recommended to retain the default values of other *routesConfig* parameters.

2.2.2.6 Install Ingress and Egress Gateways

Run the following command to install the configured Helm charts of both the Ingress and Egress Gateways:

```
helm install <chart-name> <chart-path> -n <namespace>
```

After a successful installation, both the Ingress and Egress Gateway pods must be running in your namespace.

Sample output:

Figure 2-2 Sample Output

```
pod/nwdaf-egress-gateway-5d7f4475b6-d7x62      1/1      Running   0
pod/nwdaf-egress-gateway-5d7f4475b6-vs578      1/1      Running   0
pod/nwdaf-ingress-gateway-6d8df94bbf-kjr5x      1/1      Running   0
pod/nwdaf-ingress-gateway-6d8df94bbf-kqv2v      1/1      Running   0
```

2.2.2.7 Installing OCNWDAF Package

To install the OCNWDAF package, perform the following steps:

1. Update the values in the `<replace here>` tag in the `values.yaml` file under the `<release directory>/ocn-nwdaf-helmChart/helmChart/` directory according to the setup.

```
...
cluster:
  name: &clusterName '<replace here>'
  namespace: &nameSpace '<replace here>'
  dbConfig:
    MYSQL_HOST: &mysqlHost '<replace here>'
    MYSQL_PORT: &mysqlPort '<replace here>'
    MYSQL_ENGINE: &mysqlEngine '<replace here>'
    CNDBTIER_NAMESPACE: &cndbNameSpace '<replace here>'
    CNDBTIER_SQL_POD_NAME: &cndbSQLPodName '<replace here>'
  image:
    registry: &imageRegistry '<replace here>'
  gateway:
    registry: '<replace here>'
  initContainer:
    imagePullPolicy: *imagePullPolicy IfNotPresent
...
```

2. Update the `<replace here>` tag under the "KAFKA_BROKERS" variable under "NWDAF CONFIGURATION VARIABLES" section with the proper kafka broker. **Note:** Replace respective OCNWDAF namespaces and cluster names if example values are to be used.
3. Set the `Subcharts` flag in the centralized `values.yaml` file under the `<release directory>/ocn-nwdaf-helmChart/helmChart/` directory. The allowed values are true or false. The services with the flag set to "false" are not deployed.
4. Optionally, update any other parameter in centralized or subchart `values.yaml` files. For example, Prometheus monitoring details or hooks environment variables in the centralized `values.yaml` under the `ocn-nwdaf-helmChart/helmChart` directory. Any

microservice specific values like image name or tag, environment variables in microservices subchart *values.yaml* file.

The following list is the default variables used to configure OCNWDAF, these variables are present in the centralized *values.yaml* files and in the secrets:

- MYSQL_HOST
- MYSQL_PORT
- KAFKA_BROKERS
- 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

5. Install OCNWDAF, run the following Helm installation command:

```
helm install <installation name> <path to the chart directory> -  
n $K8_NAMESPACE --timeout <timeout>m
```

For example:

```
helm install nwdafe helmChart/ -n ocnwdaf-ns --timeout 30m
```

Note

The parameter `--timeout` is optional. It is recommended to use this parameter to avoid any installation failure due to slow internet or CPU speeds. Use appropriate value for this parameter depending on the speed of image pull from the nodes of the setup. The recommended timeout value is 30 minutes.

Mandatory Installation Instruction

Note

Some services are release name dependent, use "nwdaf" for <installation name> in the Helm install command.

For example:

```
[cloud-user@ocne224-cluster-bastion-1]$ helm install nwdaf helmChart/ -n  
nwdaf-test --timeout 30m
```

Sample output after installation is complete:

```
[cloud-user@ocne224-cluster-bastion-1]$ helm install nwdaf helmChart/ -  
n $K8_NAMESPACE --timeout 30m  
W0404 04:44:48.456730 3847781 warnings.go:70] autoscaling/v2beta2  
HorizontalPodAutoscaler is deprecated in v1.23+, unavailable in v1.26+; use  
autoscaling/v2 HorizontalPodAutoscaler  
W0404 04:44:48.459573 3847781 warnings.go:70] autoscaling/v2beta2  
HorizontalPodAutoscaler is deprecated in v1.23+, unavailable in v1.26+; use  
autoscaling/v2 HorizontalPodAutoscaler  
W0404 04:51:41.957767 3847781 warnings.go:70] autoscaling/v2beta2  
HorizontalPodAutoscaler is deprecated in v1.23+, unavailable in v1.26+; use  
autoscaling/v2 HorizontalPodAutoscaler  
W0404 04:51:41.963127 3847781 warnings.go:70] autoscaling/v2beta2  
HorizontalPodAutoscaler is deprecated in v1.23+, unavailable in v1.26+; use  
autoscaling/v2 HorizontalPodAutoscaler  
NAME: nwdaf  
LAST DEPLOYED: Tue Apr 4 04:44:47 2023  
NAMESPACE: nwdaf-test  
STATUS: deployed  
REVISION: 1  
TEST SUITE: None
```

Verify if all the dependencies are in **Running** state (if any pod is not in **Running** state wait for a maximum of five restarts).

Run the following command to view all the resources present in the namespace:

```
kubectl get all -n $K8_NAMESPACE
```

For example:

```
[cloud-user@ocne224-cluster-bastion-1 ~]$ kubectl get all -n $K8_NAMESPACE
```

Figure 2-3 Sample Output

```
Every 2.0s: kubectl get all -n tantest
```

NAME	READY	STATUS	RESTARTS	AGE
pod/cap4c-api-gateway-deploy-5b85fd5cb-d4478	1/1	Running	0	43m
pod/cap4c-configuration-manager-service-deploy-6f74687888-kpkzk	1/1	Running	0	43m
pod/cap4c-kafka-ingestor-deploy-7dff676b57-kwz55	1/1	Running	0	43m
pod/cap4c-model-controller-deploy-7869468774-6cndg	1/1	Running	0	43m
pod/cap4c-model-executor-sts-0	1/1	Running	0	43m
pod/cap4c-stream-analytics-deploy-948897f9-bxl5z	1/1	Running	0	43m
pod/cap4c-stream-transformer-deploy-6c6bccf4b8-9w6mg	1/1	Running	0	15m
pod/kafka-mirror-maker-cronjob-28184148-hf4d4	0/1	Completed	0	42m
pod/kafka-mirror-maker-cronjob-28184156-2dpgv	0/1	Completed	0	34m
pod/kafka-mirror-maker-cronjob-28184158-729tx	1/1	Running	0	31m
pod/kafka-sts-0	1/1	Running	0	50m
pod/kafka-sts-1	1/1	Running	0	50m
pod/kafka-sts-2	1/1	Running	0	50m
pod/mesa-simulator-deploy-5b89ff549-m54dg	1/1	Running	0	38m
pod/nrfclient-appinfo-84d967c47c-fh8nb	1/1	Running	0	43m
pod/nrfclient-ocnf-nrf-client-nfdisccovery-546cdf78bc-h2vcg	1/1	Running	0	43m
pod/nrfclient-ocnf-nrf-client-nfdisccovery-546cdf78bc-mk9hl	1/1	Running	0	43m
pod/nrfclient-ocnf-nrf-client-nfmanagement-59ddc64b9c-vtphr	1/1	Running	0	43m
pod/nrfclient-ocpm-config-5b68669898-zr4j8	1/1	Running	0	43m
pod/nwdaf-cap4c-reporting-service-deploy-7568f56f65-hpnfp	1/1	Running	0	43m
pod/nwdaf-cap4c-scheduler-service-deploy-7df8c4cf98-mh8hg	1/1	Running	0	43m
pod/nwdaf-cap4c-spring-cloud-config-server-deploy-688965748d-xxzrl	1/1	Running	0	50m
pod/nwdaf-egress-gateway-7bd96cc656-26n6p	1/1	Running	0	43m
pod/nwdaf-egress-gateway-7bd96cc656-pk8l6	1/1	Running	0	43m
pod/nwdaf-ingress-gateway-6776974ddc-lmvdv	1/1	Running	0	43m
pod/nwdaf-ingress-gateway-6776974ddc-vb87b	1/1	Running	0	43m
pod/nwdaf-portal-deploy-5d6bfcf7d6-f2xjp	1/1	Running	0	43m
pod/nwdaf-portal-service-deploy-c5cf96f5f-fkg5j	1/1	Running	0	43m
pod/ocn-amf-simulator-service-deploy-69444c9748-gh5n2	1/1	Running	0	38m
pod/ocn-nrf-simulator-service-deploy-7cd55b78dc-s8gzv	1/1	Running	0	45m
pod/ocn-nwdaf-analytics-info-deploy-d4b95c597-kt68q	1/1	Running	0	43m
pod/ocn-nwdaf-data-collection-service-deploy-5844549765-g58nd	1/1	Running	0	17m
pod/ocn-nwdaf-datacollection-controller-deploy-688645fbbb-69l2s	1/1	Running	0	43m
pod/ocn-nwdaf-mtlf-service-deploy-5c4978f54d-qpwrr	1/1	Running	0	43m
pod/ocn-nwdaf-subscription-service-deploy-774d65b448-gz7n4	1/1	Running	0	43m
pod/ocn-oam-simulator-55556b69d7-9xdkw	1/1	Running	0	38m
pod/ocn-smf-simulator-service-deploy-7c699cb7db-jnt7j	1/1	Running	0	38m
pod/redis-master-pod	1/1	Running	0	43m
pod/redis-slave-sts-0	1/1	Running	0	43m
pod/redis-slave-sts-1	1/1	Running	0	43m
pod/zookeeper-sts-0	1/1	Running	0	50m

OCNWDaf Microservices Port Mapping

Table 2-8 Port Mapping

Service	Port Type	IP Type	Network Type	Service Port	Container Port
ocn-nwdaf-analytics	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
nwdaf-egress-gateway	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
nwdaf-ingress-gateway	External	NodePort	External/ K8s	80/TCP	8081/TCP
ocn-nwdaf-data-collection	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
ocn-nwdaf-mtlf	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
ocn-nwdaf-subscription	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
ocn-nwdaf-analytics-info	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
ocn-nwdaf-configuration	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
ocn-nwdaf-georedagent	Internal	ClusterIP	Internal / K8s	9181/TCP	9181/TCP
cap4c-kafka-ingestor	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP

Table 2-8 (Cont.) Port Mapping

Service	Port Type	IP Type	Network Type	Service Port	Container Port
cap4c-model-controller	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
cap4c-model-executor	Internal	ClusterIP	Internal / K8s	9092/TCP	9092/TCP
cap4c-stream-transformer	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
cap4c-stream-analytics	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
cap4c-api-gateway	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
nwdaf-cap4c-reporting-service	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
nwdaf-cap4c-scheduler-service	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
nwdaf-portal	External	NodePort	External / K8s	80/TCP	
nwdaf-portal-service	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
cap4c-configuration-manager-service	Internal	ClusterIP	Internal / K8s	9000/TCP	9000/TCP

Note

For NodePort services, Kubernetes allocates the Service Port.

Installation of Simulator Chart

Follow the procedure below to install the simulator chart:

1. Update the values in the `<replace here>` tag present in `values.yaml` under `/simulator-helmchart/` based on the setup:

```
...
cluster:
  name: &clusterName '<replace here>'
  namespace: &nameSpace '<replace here>'
  dbConfig:
    MYSQL_HOST: &mysqlHost '<replace here>'
    MYSQL_PORT: &mysqlPort '<replace here>'
    MYSQL_ENGINE: &mysqlEngine '<replace here>'
    CNDBTIER_NAMESPACE: &cndbNameSpace '<replace here>'
    CNDBTIER_SQL_POD_NAME: &cndbSQLPodName '<replace here>'
...
```

2. Set the *Subcharts* flag in the centralized *values.yaml* file under the */simulator-helmchart* directory. The allowed values are true or false. The services with the flag set to false are not deployed.
3. Optionally, update any other parameter in centralized or subchart *values.yaml* files. For example, Prometheus monitoring details or hooks environment variables in the centralized *values.yaml* under the */simulator-helmchart* directory. Any microservice specific values like image name or tag, environment variables in microservices subchart *values.yaml* file.
4. Install simulators, run the following Helm installation command:

```
helm install <installation name> <path to the chart directory> -  
n $K8_NAMESPACE --timeout <timeout>h
```

For example:

```
helm install simulators simulator-helmchart/ -n ocnwdaf-ns --timeout 30m
```

Note

The parameter `--timeout` is optional. It is recommended to use this parameter to avoid any installation failure due to slow internet or CPU speeds. Use appropriate value for this parameter depending on the speed of image pull from the nodes of the Bastion host. The recommended timeout value is 30 minutes.

Sample of the terminal screen once the installation starts:

```
[cloud-user@ocne224-cluster-bastion-2 ocn-nwdaf-helmChart]$ helm install  
simulators simulator-helmChart/ -n ttest --timeout 30m  
W0511 10:38:19.670067 2848359 warnings.go:70]  
spec.template.spec.containers[0].env[61].name: duplicate name  
"SPRING_KAFKA_CONSUMER_PROPERTIES_MAX_POLL_INTERVAL_MS"  
NAME: simulators  
LAST DEPLOYED: Thu May 11 10:38:12 2023  
NAMESPACE: ttest  
STATUS: deployed  
REVISION: 1  
TEST SUITE: None
```

5. Run the following command to verify if all the dependencies are in **Running** state (if any pod is not in **Running** state wait for a maximum of five restarts):

```
kubectl get all -n $K8_NAMESPACE
```

Sample output:

Figure 2-4 Sample Output

```
Every 2.0s: kubectl get all -n tantest blurr8
```

NAME	READY	STATUS	RESTARTS	AGE
pod/cap4c-api-gateway-deploy-5b85fd5cb-d4478	1/1	Running	0	43m
pod/cap4c-configuration-manager-service-deploy-6f74687888-kpkzk	1/1	Running	0	43m
pod/cap4c-kafka-ingestor-deploy-7dff676b57-kwz55	1/1	Running	0	43m
pod/cap4c-model-controller-deploy-7869468774-6cndg	1/1	Running	0	43m
pod/cap4c-model-executor-sts-0	1/1	Running	0	43m
pod/cap4c-stream-analytics-deploy-948897f9-bxl5z	1/1	Running	0	43m
pod/cap4c-stream-transformer-deploy-6c6bccf4b8-9w6mg	1/1	Running	0	15m
pod/kafka-mirror-maker-cronjob-28184148-hf4d4	0/1	Completed	0	42m
pod/kafka-mirror-maker-cronjob-28184156-2dpgv	0/1	Completed	0	34m
pod/kafka-mirror-maker-cronjob-28184158-729tx	1/1	Running	0	31m
pod/kafka-sts-0	1/1	Running	0	50m
pod/kafka-sts-1	1/1	Running	0	50m
pod/kafka-sts-2	1/1	Running	0	50m
pod/mesa-simulator-deploy-5b89ff549-m54dg	1/1	Running	0	38m
pod/nrfclient-appinfo-84d967c47c-fh8nb	1/1	Running	0	43m
pod/nrfclient-ocnf-nrf-client-nfdisccovery-546cdf78bc-h2vcg	1/1	Running	0	43m
pod/nrfclient-ocnf-nrf-client-nfdisccovery-546cdf78bc-mk9hl	1/1	Running	0	43m
pod/nrfclient-ocnf-nrf-client-nfmanagement-59ddc64b9c-vtphr	1/1	Running	0	43m
pod/nrfclient-ocpm-config-5b68669898-zr4j8	1/1	Running	0	43m
pod/nwdaf-cap4c-reporting-service-deploy-7568f56f65-hpnfp	1/1	Running	0	43m
pod/nwdaf-cap4c-scheduler-service-deploy-7df8c4cf98-mh8hg	1/1	Running	0	43m
pod/nwdaf-cap4c-spring-cloud-config-server-deploy-688965748d-xxzrl	1/1	Running	0	50m
pod/nwdaf-egress-gateway-7bd96cc656-26n6p	1/1	Running	0	43m
pod/nwdaf-egress-gateway-7bd96cc656-pk8l6	1/1	Running	0	43m
pod/nwdaf-ingress-gateway-6776974ddc-lmvdv	1/1	Running	0	43m
pod/nwdaf-ingress-gateway-6776974ddc-vb87b	1/1	Running	0	43m
pod/nwdaf-portal-deploy-5d6bfcf7d6-f2xjp	1/1	Running	0	43m
pod/nwdaf-portal-service-deploy-c5cf96f5f-fkg5j	1/1	Running	0	43m
pod/ocn-amf-simulator-service-deploy-69444c9748-gh5n2	1/1	Running	0	38m
pod/ocn-nrf-simulator-service-deploy-7cd55b78dc-s8gzv	1/1	Running	0	45m
pod/ocn-nwdaf-analytics-info-deploy-d4b95c597-kt68q	1/1	Running	0	43m
pod/ocn-nwdaf-data-collection-service-deploy-5844549765-g58nd	1/1	Running	0	17m
pod/ocn-nwdaf-datacollection-controller-deploy-688645fbbb-69l2s	1/1	Running	0	43m
pod/ocn-nwdaf-mtlf-service-deploy-5c4978f54d-qpwrr	1/1	Running	0	43m
pod/ocn-nwdaf-subscription-service-deploy-774d65b448-gz7n4	1/1	Running	0	43m
pod/ocn-oam-simulator-55556b69d7-9xdkw	1/1	Running	0	38m
pod/ocn-smf-simulator-service-deploy-7c699cb7db-jnt7j	1/1	Running	0	38m
pod/redis-master-pod	1/1	Running	0	43m
pod/redis-slave-sts-0	1/1	Running	0	43m
pod/redis-slave-sts-1	1/1	Running	0	43m
pod/zookeeper-sts-0	1/1	Running	0	50m

6. The following services with port mapping are deployed:

Table 2-9 Port Mapping

Service	Port Type	IP Type	Network Type	Service Port	Container Port
ocn-nrf-simulator	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
ocn-amf-simulator	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
mesa-simulator	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
ocn-smf-simulator	Internal	ClusterIP	Internal / K8s	8080/TCP	8080/TCP
ocn-oam-simulator	Internal	ClusterIP	Internal / K8s	8085/TCP	8085/TCP

Configure Service Parameters

In the *values.yaml* under */helmchart/* select the services to deploy, the Helm chart parameters are listed below:

```
nrfclient.enabled: true
ocn-nrf-simulator.enabled: true
nwdaf-cap4c-zookeeper.enabled: true
```

```
nwdaf-cap4c-kafka.enabled: true
nwdaf-cap4c-redis.enabled: true
nwdaf-cap4c-spring-cloud-config-server.enabled: true
nwdaf-cap4c-scheduler-service.enabled: true
nwdaf-cap4c-reporting-service.enabled: true
nwdaf-cap4c-stream-analytics.enabled: true
nwdaf-cap4c-model-executor.enabled: true
nwdaf-cap4c-model-controller.enabled: true
nwdaf-cap4c-kafka-ingestor.enabled: true
ocn-nwdaf-configuration-service.enabled: true
ocn-nwdaf-subscription.enabled: true
ocn-nwdaf-data-collection.enabled: true
ocn-nwdaf-mtls.enabled: true
ocn-nwdaf-analytics.enabled: true
ocnNwdafGeoredagent.enabled: false
nwdaf-portal-service.enabled: true
nwdaf-portal.enabled: true
common-services-gateways.enabled: true
cap4cDeployTemp.enabled: false
ocn-nwdaf-datacollection-controller.enabled: true
```

In the *values.yaml* under */simulator-helmChart/* select the simulators to deploy, the simulator Helm chart parameters are listed below:

```
ocn-smf-simulator.enabled: true
ocn-mesa-simulator.enabled: true
ocn-amf-simulator.enabled: true
ocn-oam-simulator.enabled: true
```

2.2.3 Postinstallation Tasks

This section explains the postinstallation tasks for OCNWDAF.

2.2.3.1 Verifying Installation

To verify the installation:

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

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

Where,

<helm-release> is the Helm release name of OCNWDAF. <namespace>

<namespace> is the namespace of OCNWDAF deployment.

For example:

```
helm status ocndaf -n ocndaf
```

If the deployment is successful, then the STATUS is displayed as deployed.

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

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

Where,

<namespace> is the namespace of OCNWDAF deployment.

The STATUS column of all the pods must be 'Running'.

The READY column of all the pods must be n/n, where n is the number of containers in the pod.

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

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

Where,

<namespace> is the namespace of OCNWDAF deployment.

If the installation is unsuccessful or the status of all the pods is not in RUNNING state, , perform the troubleshooting steps provided in *Oracle Communications Networks Data Analytics Function Troubleshooting Guide*.

2.2.3.2 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

```

2.2.3.3 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 CNC Console's custom values.yaml file.

Follow the steps listed below:

1. Set the Image Repository

In the CNC Console's custom values.yaml file, set the Image Repository to the repository where the images are located. The parameter *dockerRegistry* is found in line number 10 of the global parameters section:
For example:

```
dockerRegistry: ocnwdaf-docker.dockerhub-phx.oc1.oraclecorp.com
```

2. Update the Cluster Domain

Update the cluster's DNS domain based on the deployment. The parameter *clusterDomain* is located in line number 16 of the global parameters section:
For example:

```
clusterDomain: &clusterDomain "sunstreaker"
```

To identify the cluster name, run the following command:

```
kubectl -n kube-system get configmap kubeadm-config -o yaml | grep -i
dnsDomain
```

3. Load Balancer Configuration

If a Load Balancer is used, use the following configuration:

The annotation *metallb.universe.tf/address-pool: signaling/oam* is required in global section if MetalLB in CNE 1.8.x onwards is used

```
# Line 25:
customExtension:
```

```

lbServices:
  labels: {}
  annotations:
    # The annotation metallb.universe.tf/address-pool: signaling/oam is
    # required if MetalLB in CNE 1.8.x is used
    metallb.universe.tf/address-pool: oam
    service.beta.kubernetes.io/oci-load-balancer-internal: "true"

```

4. Set Database Details

Use the following configuration:

```

# DB Details by fqdn
dbHost: &mysqlHostRef "mysql-connectivity-service.<namespace_name>"
dbPort: &mysqlPortRef "3306"
secretName: &mysqlSecretNameRef cncc-db-secret

# DB Details by external ip:
dbHost: &mysqlHostRef 10.233.34.56 <- External ip from mysql-
connectivity-service
dbPort: &mysqlPortRef 3306
secretName: &mysqlSecretNameRef cncc-db-secret

```

5. Activate Cluster IP for Load Balancer

Set the parameter *useClusterIpForLbServices* to true.

Use the following configuration:

```

# Use ClusterIP for LoadBalancer(LB) services.
# The LB services are assigned LoadBalancer service type in k8s service
# definition. Set this flag to true to assign ClusterIP service type.
useClusterIpForLbServices: true

```

6. Update Automatic route generation for CNC Console Manager and Agent Deployment

Update the Automatic route generation for CNC Console Manager and Agent Deployment sections either using an external IP or a Load Balancer:

Using an external IP

```

self:
  cnccId: Cluster1

  mCnccIams:
    - id: Cluster1

    # IP of one of external IP of Cluster nodes
    ip: <external_k8s_node_ip>

    # IAM app port
    port: <service_node_port>

```

```

mCnccCores:

  - id: Cluster1

aCnccs:

  - id: Cluster1

    role: Cluster1

    # Path to acore ingress service "service-
name.namespace.svc.clustername"

    fqdn: nwdaf-cncc-acore-ingress-
gateway.<namespace_name>.svc.<cluster_domian>

    # cncc app port

    port: 80

instances:

  - id: OCCNE-NWDAF-UI-instance1

    # Set type to NWDAF-UI

    type: NWDAF-UI

    owner: Cluster1

    # Path to nwdaf portal UI service "service-
name.namespace.svc.clustername"

    fqdn: nwdaf-portal.<namespace_name>.svc.<cluster_domian>

    # Portal UI port

    port: 80

    # Path to NWDAF on kubernetes cluster "clustername/namespace/ocnwdafe"

    apiPrefix: /<cluster_domain>/<namespace_name>/ocnwdafe

  - id: OCCNE-NWDAF-API-instance1

    # Set type to NWDAF-API

    type: NWDAF-API

    owner: Cluster1

    # Path to nwdaf API service "service-name.namespace.svc.clustername"

    fqdn: cap4c-api-gateway.<namespace_name>.svc.<cluster_domain>

    # Portal API port

```



```

    port: 8080

    # Path to NWDAF on kubernetes cluster "clustername/namespace/
    ocnwdafapi"

    apiPrefix: /<cluster_domain>/<namespace_name>/ocnwdafapi

```

For example:

```

self:
  cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.123.158.150
      port: 30085
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.blurr7
      port: 30076
  instances:
    - id : Cluster1-nwdaf-instance1
      # Set type to NWDAF-UI
      type: NWDAF-UI
      owner: Cluster1
      fqdn: nwdaf-portal.ocnwdaf-ns.svc.blurr7
      port: 80
      apiPrefix: /blurr7/ocnwdaf-ns/ocnwdaf
    - id : Cluster1-nwdaf-instance1
      type: NWDAF-API
      owner: Cluster1
      fqdn: cap4c-api-gateway.ocnwdaf-ns.svc.blurr7
      port: 8080
      apiPrefix: /blurr7/ocnwdaf-ns/ocnwdafapi

```

Using Load Balancer

```

self:

  cnccId: Cluster1

  mCnccIams:

    - id: Cluster1

      # IP of one of Load Balancer IPs of Cluster nodes

      ip: 10.75.245.212

  mCnccCores:

    - id: Cluster1

```

```

aCnccs:
  - id: Cluster1
    role: Cluster1
    # Path to acore ingress service "service-
name.namespace.svc.clustername"
    fqdn: cncc-acore-ingress-gateway.cncc.svc.sunstreaker
    # cncc app port
    port: 80

instances:
  - id: OCCNE-NWDAF-UI-instance1
    # Set type to NWDAF-UI
    type: NWDAF-UI
    owner: Cluster1
    # Path to nwdafe portal UI service "service-
name.namespace.svc.clustername"
    fqdn: nwdafe-portal.ocnwdafe-ns.svc.sunstreaker
    # Portal UI port
    port: 80
    # Path to NWDAF on kubernetes cluster "clustername/namespace/
ocnwdafe"
    apiPrefix: /sunstreaker/ocnwdafe-ns/ocnwdafe
  - id: OCCNE-NWDAF-API-instance1
    # Set type to NWDAF-API
    type: NWDAF-API
    owner: Cluster1
    # Path to nwdafe API service "service-
name.namespace.svc.clustername"
    fqdn: cap4c-api-gateway.ocnwdafe-ns.svc.sunstreaker
    # Portal API port
    port: 8080

```

```

# Path to NWDAF on kubernetes cluster "clustername/namespace/
ocnwdaface"

apiPrefix: /sunstreaker/ocnwdaface-ns/ocnwdaface

```

For example:

```

self:
  cnccId: Cluster1
  mCnccIams:
    - id: Cluster1
      ip: 10.75.245.212
  mCnccCores:
    - id: Cluster1
  aCnccs:
    - id: Cluster1
      role: Cluster1
      fqdn: cncc-acore-ingress-gateway.cncc.svc.sunstreaker
      port: 80
  instances:
    - id: OCCNE-NWDAF-UI-instance1
      type: NWDAF-UI
      owner: Cluster1
      fqdn: nwdaface-portal.ocnwdaface-ns.svc.sunstreaker
      port: 80
      apiPrefix: /sunstreaker/ocnwdaface-ns/ocnwdaface
    - id: OCCNE-NWDAF-API-instance1
      type: NWDAF-API
      owner: Cluster1
      fqdn: cap4c-api-gateway.ocnwdaface-ns.svc.sunstreaker
      port: 8080
      apiPrefix: /sunstreaker/ocnwdaface-ns/ocnwdafaceapi

```

7. Move to the CNC Console IAM Attributes Section

Move to the CNC Console IAM attributes section and update the values as follows:

Update port to the same used by IAM

LOC 244: publicHttpSignalingPort: 30085

If Static node port needs to be set, then set staticNodePortEnabled flag to true and provide value for staticNodePort.

Else random node port will be assigned by K8

```

staticNodePortEnabled: true

staticHttpNodePort: 30085

staticHttpsNodePort: 30053

```

8. Move to the CNC Console Core Attributes Section

Move to the CNC Console Core attributes section and update the values as follows:

Update port to the same used by cncc core

LOC 244: publicHttpSignalingPort: 30085

If Static node port needs to be set, then set staticNodePortEnabled flag to true and provide value for staticNodePort.

Else random node port will be assigned by K8

```
staticNodePortEnabled: true
```

```
staticHttpNodePort: 30085
```

```
staticHttpsNodePort: 30053
```

To identify the external IPs during installation, run the following command:

```
kubectl get pods -n ocnwdaf-ns -owide | grep portal
```

9. Helm Install

Run the Helm install command in the folder where the custom yaml file is located. For example:

```
helm install cncc occncc-xx.x.x.tgz -f occncc_custom_values_xx.x.x.yaml -n cncc
```

10. Monitor the Installation

To monitor the installation process, run the following command:

```
watch kubectl get pods -n cncc
```

You can access the IAM and CNC Console once the pods are in up and running state.

11. Verify IAM

Verify if IAM is running. For example:

IAM: `http://10.123.158.150:30085/`

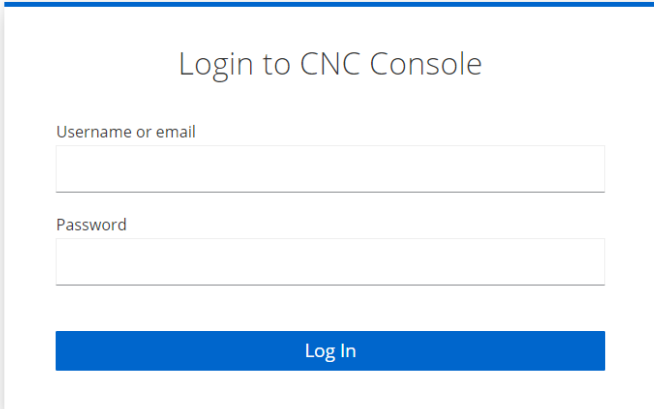
Default user: admin

Default password: password

12. Login to the CNC Console.

Provide the **Username** and **Password**.

Figure 2-5 Login

The image shows the Oracle logo at the top. Below it is a login form titled "Login to CNC Console". The form has two input fields: "Username or email" and "Password". Below these fields is a blue button labeled "Log In".

Click **Login**.

Integrate OCNWDAF and CNC Console

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

1. Login to the CNC Console
2. Click **Clients** option, In the **Settings** tab update the **Root URL** field with the IP on which CNC Console is running and the port defined on *mcore-ingress-gateway* service.

Figure 2-6 Clients

The screenshot shows the Oracle Identity Cloud Service console. On the left is a dark sidebar with a 'Clients' menu item highlighted. The main area is titled 'Clients > cncc'. Below the title is a tabbed interface with 'Settings' selected. The settings form includes fields for Client ID (cncc), Name, Description, Enabled (ON), Always Display in Console (OFF), Consent Required (OFF), Login Theme (cncc), Client Protocol (openid-connect), Access Type (public), Standard Flow Enabled (ON), Implicit Flow Enabled (OFF), Direct Access Grants Enabled (OFF), OAuth 2.0 Device Authorization Grant Enabled (OFF), Front Channel Logout (OFF), and Root URL (http://10.123.158.150:30075).

Click **Save**

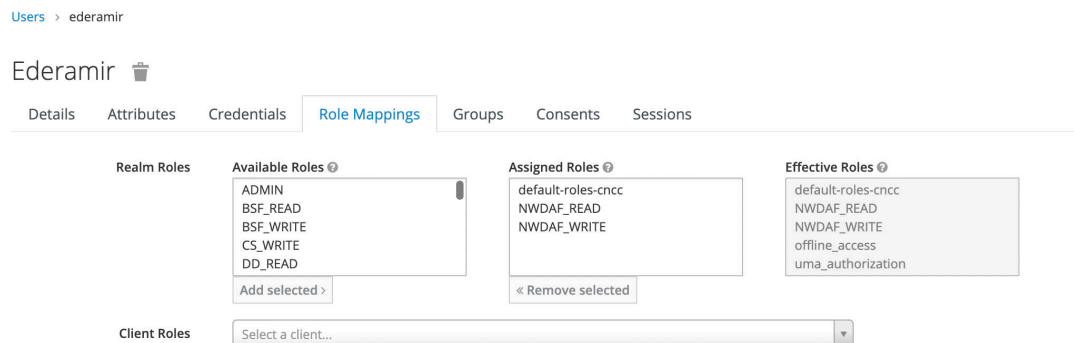
3. To add new user, click **Users** and click **Add User**. Provide a **Username** and fill the form. Click **Save**.

Figure 2-7 Add User

The screenshot shows the 'Add user' form. It includes fields for ID, Created At, Username (with a red asterisk and the text 'newuser' entered), Email, First Name, Last Name, User Enabled (ON), Email Verified (OFF), Groups (a dropdown menu showing 'Select existing group...' and 'No group selected'), and Required User Actions (a dropdown menu showing 'Select an action...'). At the bottom are 'Save' and 'Cancel' buttons.

- Click **Users**, select the newly added user, go to the **Role Mappings** tab.
For example:

Figure 2-8 Role Mapping



Add **NWDAF_READ** and **NWDAF_WRITE** to the **Assigned Roles**.

- Use the **Reset Password** screen to create a new password for the user.

Figure 2-9 Reset Password

Reset Password

Password	
Password Confirmation	
Temporary ?	☒ ON
Reset Password	

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*.

Uninstall CNC Console

To uninstall CNC Console, run the following commands:

```
helm delete cncc -n cncc
```

Delete all jobs, run the following command:

```
kubectl delete jobs --all -n cncc
```

2.2.3.4 Taking a Backup

Take a backup of the following files, which are required during fault recovery:

- Updated NWDAF-custom-values.yaml file.
- Updated Helm charts.
- Secrets, certificates, and keys that are used during installation.

3

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 `values.yaml` file.

To customize the values yaml file, perform the following steps:

1. Extract 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. `values.yaml`: This file is used to customize the OCNWDAF deployment with ATS.
 - b. `nwdafAlertrules.yaml`: This file is used for Prometheus.

For more information about how to download the package from My Oracle Support, see the Installation Package Download section.

2. Customize the `ocn-nwdaf-helmChart/helmChart/values.yaml` file.
3. Save the updated `ocn-nwdaf-helmChart/helmChart/values.yaml` file in the Helm chart directory.

Note

- All parameters mentioned as mandatory must be present in `ocn-nwdaf-helmChart/helmChart/values.yaml` file and configured before the deployment.
- All fixed value parameters listed must be present in the `ocn-nwdaf-helmChart/helmChart/values.yaml` file with the exact values as specified in this section.
- The properties in the *values.yaml* of all the sub charts are configurable, but it is not advisable to update the properties. Update the properties of the sub charts only when there is a specific customer requirement.

3.1 Configurable Parameters

This section lists all the OCNWDAF configurable parameters.

3.1.1 Global Parameters

Table 3-1 Global Paramaters

Parameter	Description	Details
<code>global.test.nfName</code>	This is a mandatory parameter. The value of <i>nfName</i> is specified as <i>ocnwdaf</i> . The <i>nfName</i> is used as a prefix in the test container name.	Default value: <code>ocnwdaf</code>
<code>global.test.image.name</code>	This is a mandatory parameter. Name of the test image.	Default value: <code>nf_test</code>
<code>global.test.image.tag</code>	This is a mandatory parameter. Name of the test image tag.	Default value: <code>23.3.0</code>
<code>global.test.config.logLevel</code>	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
<code>global.test.config.timeout</code>	This is an optional parameter. Helm test hook-related configurations. Beyond this duration, helm test is considered as a failure.	Default value: 240
<code>global.test.complianceEnable</code>	This is an optional parameter. Helm test hook-related configurations. Sets the test container as compliant.	Range: true, false Default value: true
<code>global.test.resources</code>	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: 23.3.0
    config:
      logLevel: INFO
      timeout: 240
      complianceEnable: true
    resources:
      - deployments/v1
```

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

Istio Sidecar Injection Parameters

These parameters are used to enable Service Mesh support:

Note

The label of the namespace takes precedence over the configured injection parameter. If Service Mesh has to be disabled, ensure the injection parameter is not configured in the namespace or it is disabled.

Table 3-2 Istio Sidecar Paramaters

Parameter	Description	Details
injection	This parameter is used to enable Service Mesh support.	Range: true, false Default value: False
readinessCheck	Flag to enable or disable the feature. If disabled then there is no need to configure.	Range: true, false Default value: False

3.1.2 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 3-3 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

Table 3-3 (Cont.) Microservice Parameter

Parameter	Description	Details
readinessProbe.failureThreshold	This is an optional parameter. When a Pod starts and the probe fails, Kubernetes will try <i>failureThreshold</i> 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
```

3.1.3 Georedundancy Parameters

Configure the following parameters to enable and configure georedundancy in the *values.yaml* file for OCNWDAF:

Table 3-4 REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
ocnwdaf.cluster.namespace	Name space of the deployment Note: Change this to the name space 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
ocnnwdaf.geored.hooks.column2	Column2 information for the hook	current_owner
ocnnwdaf.geored.hooks.image	Image information for the hook	ocnwdaf-docker.dockerhub-phx.oc.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
ocnnwdaf.geored.agent.resources.limits.cpu	CPU Limit	1

Table 3-4 (Cont.) REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
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	ocnnwdafgeoredagentport
ocnnwdaf.geored.agent.service.prometheusport.containerPort	Container Port of the Prometheus	9000
ocnnwdaf.geored.agent.service.prometheusport.targetPort	Target Port of the Prometheus	9000
ocnnwdaf.geored.agent.service.prometheusport.name	Name of the Prometheus Note: Modify the port name based on the Prometheus on the deployed setup.	http-cnc-metrics
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_COMPONENT_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 OCNWDaf 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

Table 3-4 (Cont.) REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
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	cnDBTier Monitor Service URL for Replication	Use the Reachable Monitor Service from Deployed CNDB namespace. For example: http://mysql-cluster-db-monitor-svc.{cnbdbnamespace}.svc.{domainname}:8080/db-tier/status/replication
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_DBTIER_STATUS_URL	cnDBTier Monitor Service URL for Local	Use the Reachable Monitor Service from Deployed CNDB namespace. For example: http://mysql-cluster-db-monitor-svc.{cnbdbnamespace}.svc.{domainname}:8080/db-tier/status/local
ocnnwdaf.geored.agent.env.GEO_RED_AGENT_SITE_SECONDARY_SITEID	Secondary Site ID	OCNWDAF-XX-2
ocnnwdaf.geored.agent.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.geored.agent.env.GEO_RED_AGENT_SITE_TERTIARY_SITEID	Tertiary Site ID	OCNWDAF-XX-3
ocnnwdaf.geored.agent.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.geored.agent.env.GEO_RED_AGENT_DB_URL	IP of the Site cnDBTier	The Cluster IP/External IP of the CNDB
ocnnwdaf.geored.agent.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

Table 3-4 (Cont.) REDUNDANCY AGENT CONFIGURATION

Parameter	Description	Default Value
ocnwdaf.geored.agent.env.GEO_RED_AGENT_PASSWORD	Password for the DB User	password
ocnwdaf.geored.agent.env.GEO_RED_AGENT_ENABLE	Enable/disable GRD	false
ocnwdaf.geored.agent.env.GEO_RED_AGENT_DB_PORT	Port of the database	3306
ocnwdaf.geored.agent.env.GEO_RED_AGENT_DB_NAME	Name of the GRD database	georedagent
ocnwdaf.geored.agent.env.GEO_RED_AGENT_CONFIG_SERVER	Config Server URL for the Site's OCNWDAF	http://nwdaf-cap4c-spring-cloud-config-server:8888

3.1.4 NRF Client Parameters

To configure NRF client parameters, you should configure the following parameters in the values.yaml file. The following table provides details about the NRF client customization parameters in the values.yaml file:

Table 3-5 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.
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

Table 3-5 (Cont.) Configurable Parameters - NRF Client Configuration

Parameter	Description	Details
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
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 ready state, which will lead to failure in the helm test.	Range: true or false Default Value: true
nrf-client-nfmanagement.replicas	Contains number of NRF-Client replicas.	Default Value: 2

Table 3-6 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 3-6 (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
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

Table 3-6 (Cont.) Configurable parameters of NRF Client Configuration in Config-map

Parameter	Description	Details
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 3-7 NRF Client Parameters

Parameter	Description	Detail
nrfclient.nrf-client.configmapApplicationC onfig.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

Table 3-7 (Cont.) NRF Client Parameters

Parameter	Description	Detail
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: 23.3.0 - Set Log Level to DEBUG. - If false, Log Level shall be INFO	Range: true or false Default Value: false Applicable to: OCNWDAF
serviceMeshCheck	This is a mandatory parameter. This flag needs to be set to true if a Service Mesh is present in the environment where OCNWDAF is deployed.	Range: True or False. Default value: False Applicable to: OCNWDAF
istioSidecarQuitUrl	This is a mandatory parameter. It needs to be set with correct URL format http://127.0.0.1:<istio management port>/quitquitquit" if Service Mesh is present in the environment where OCNWDAF is deployed.	Default value: http://127.0.0.1:15000/quitquitquit" Applicable to: OCNWDAF
istioSidecarReadyUrl	This is a mandatory parameter. It needs to be set with correct URL format http://127.0.0.1:<istio management port>/ready" if Service Mesh is present in the environment where OCNWDAF is deployed.	Default value: http://127.0.0.1:<istio management port>/ready" Applicable to: OCNWDAF

Table 3-8 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

Table 3-8 (Cont.) NRF Client NF Discovery Parameters

Parameter	Description	Details
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. 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
dbConfig.dbEngine	This is a mandatory parameter. Database engine name.	Default Value: NDBCLUSTER Applicable to: OCNWDAF
serviceMeshCheck	This is a mandatory parameter. This flag needs to be set to true if a Service Mesh is present in the environment where OCNWDAF is deployed.	Range: True or False. Default value: False Applicable to: OCNWDAF

Table 3-8 (Cont.) NRF Client NF Discovery Parameters

Parameter	Description	Details
istioSidecarQuitUrl	This is a mandatory parameter. It needs to be set with correct URL format <code>http:// 127.0.0.1:<istio management port>/quitquitquit</code> if Service Mesh is present in the environment where OCNWDAF is deployed.	Default value: <code>http:// 127.0.0.1:15000/ quitquitquit</code> Applicable to: OCNWDAF
istioSidecarReadyUrl	This is a mandatory parameter. It needs to be set with correct URL format <code>http:// 127.0.0.1:<istio management port>/ready</code> if Service Mesh is present in the environment where OCNWDAF is deployed.	Default value: <code>http:// 127.0.0.1:<istio management port>/ready</code> Applicable to: OCNWDAF

Table 3-9 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
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 <code>ocnwdaf-<version>custom-values.yaml</code> example. Applicable to: OCNWDAF
dbConfig.dbHost	This is a mandatory parameter. DB credential of nrf-client-nfdiscovery parameters.	Default Value: Valid database host (Example: <code>ocnwdaf-nsdb.db</code>) Applicable to: OCNWDAF

Table 3-9 (Cont.) NRF Client NF Management Parameters

Parameter	Description	Details
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	This is a mandatory parameter. This flag needs to be set to true if a Service Mesh is present in the environment where OCNWDAF is deployed.	Range: True or False. Default value: False Applicable to: OCNWDAF
istioSidecarQuitUrl	This is a mandatory parameter. It needs to be set with correct URL format http:// 127.0.0.1:<istio management port>/quitquitquit" if Service Mesh is present in the environment where OCNWDAF is deployed.	Default value: http:// 127.0.0.1:15000/ quitquitquit" Applicable to: OCNWDAF
istioSidecarReadyUrl	This is a mandatory parameter. It needs to be set with correct URL format http:// 127.0.0.1:<istio management port>/ready" if Service Mesh is present in the environment where OCNWDAF is deployed.	Default value: http:// 127.0.0.1:<istio management port>/ready" Applicable to: OCNWDAF

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 service 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 Database 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
serviceName: ""

# 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 $\text{discoveryDurationBeforeExpiry} * 60/100$

- # 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 $\text{nfHeartBeatRate} * 60/100$
- # 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,"nsiList":null,"fqdn":"pcf.oracle.com","interPlmnFqdn":null,"ipv4Addresses":null,"ipv6Addresses":null,"priority":null,"capacity":null,"load":null,"locality":null,"udrInfo":null,"udmInfo":null,"ausfInfo":null,"amfInfo":null,"smfInfo":null,"upfInfo":null,"pcfInfo":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":"2019-08-03T18:55:08.871+0000"}],"scheme":"http","nfServiceStatus":"REGISTERED","fqdn":"nrf.oracle.com","interPlmnFqdn":null,"apiPrefix":"bsf-service","defaultNotificationSubscriptions":null,"allowedPlmns":null,"allowedNFTypes":["PCF","NEF","NWDAF"],"allowedNfDomains":null,"allowedNssais":null,"priority":null,"capacity":null,"load":null,"recoveryTime":null,"supportedFeatures":null}],"sNssais":null}]
```

```
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,"nsiList":null,"fqdn":"ocn-nwdaf-
communication","interPlmnFqdn":null,"ipv4Addresses":null,"ipv6Addresses":null,"priority":null,"capacity":null,"load":null,"locality":null,"udrInfo":null,"udmInfo":null,"ausfInfo":null,"amfInfo":null,"smfInfo":null,"upfInfo":null,"pcfInfo":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":"2019-08-03T18:55:08.871+0000"}],"scheme":"http","nfServiceStatus":"REGISTERED","fqdn":"nrf.oracle.com","interPlmnFqdn":null,"apiPrefix":"bsf-service","defaultNotificationSubscriptions":null,"allowedPlmns":null,"allowedNFTypes":["PCF","NEF","NWDAF"],"allowedNfDomains":null,"allowedNssais":null,"priority":null,"capacity":null,"load":null,"recoveryTime":null,"supportedFeatures":null}],"sNssais":null}]

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,"errorReasonsForFailure":
["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 Database 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 nrclient 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: []

```

3.1.5 Ingress Gateway Parameters

Listed below are the customizable Ingress Gateway parameters:

Table 3-10 Routes Config Customizable Parameters

Parameter	Description	Details
globalRemoveRequestHeader[0].name	This field is used for blacklisting (removing) a request header at global level. Hence, it will be applied to all routes configured. Additional header can be configured by adding a new element in the next line and so on.	Default Value: Value to be updated accordingly

Table 3-10 (Cont.) Routes Config Customizable Parameters

Parameter	Description	Details
globalRemoveResponseHeader[0].name	This field is used for blacklisting(removing) a response header at global level. Hence, it will be applied to all routes configured. Additional header can be configured by adding a new element in the next line and so on.	Default Value: Value to be updated accordingly
routesConfig[0].id	This is a mandatory parameter. Id of the route.	Default Value: Value to be updated accordingly
routesConfig[0].uri	This is a mandatory parameter. Service name of the internal microservice of this NF.	Default Value: Value to be updated accordingly
routesConfig[0].path	This is a mandatory parameter. Provide the path to be matched.	Default Value: Value to be updated accordingly
routesConfig[0].order	This is a mandatory parameter. Provide the order of the execution of this route.	Default Value: Value to be updated accordingly
routesConfig[0].metadata.requestTimeout	requestTimeout is used to set timeout at route level. Value should be in milliseconds. If present then it will override global request timeout	Default Value: 0
routesConfig[0].metadata.xfccHeaderValidation.validationEnabled	This is used to provide an option to enable or disable route level xfccHeaderValidation, it will override global configuration for xfccHeaderValidation.enabled	Default Value: False
routesConfig[0].metadata.oauthValidator.enabled		Default Value: Value to be updated accordingly
routesConfig[0].metadata.ccaHeaderValidation.enabled	This is used to provide an option to enable or disable route level ccaHeaderValidation, it will override global configuration for ccaHeaderValidation.enabled	Default Value: False
routesConfig[0].filters.invalidRouteFilter.errorCodeOnInvalidRoute	This is a mandatory parameter. If invalidRouteFilter filter is configured, then keep the 'order' value highest compared to other routes. Comment the parameters related to invalidRouteFilter if configurable error code is not required for invalid route. Configurable error code for invalid route.	Default Value: Value to be updated accordingly
routesConfig[0].filters.invalidRouteFilter.errorCauseOnInvalidRoute	This is a mandatory parameter. Error cause for invalid route.	Default Value: Value to be updated accordingly
routesConfig[0].filters.invalidRouteFilter.errorTitleOnInvalidRoute	This is a mandatory parameter. Error title for invalid route.	Default Value: Value to be updated accordingly

Table 3-10 (Cont.) Routes Config Customizable Parameters

Parameter	Description	Details
<code>routesConfig[0].filters.invalidRouteFilter.errorDescriptionOnInvalidRoute</code>	This is a mandatory parameter. Error description for invalid route.	Default Value: Value to be updated accordingly
<code>routesConfig[0].filters.removeRequestHeader[0].name</code>	This field is used for blacklisting(removing) a request header at route level. Additional header can be configured by adding a new element in the next line and so on.	The value of "name" attribute denotes the name of the request header which is to be blacklisted/removed at route level. Add a new entry in next line for every header to be removed.
<code>routesConfig[0].filters.removeResponseHeader[0].name</code>	Below field is used for blacklisting(removing) a response header at route level. Additional header can be configured by adding a new element in the next line and so on.	The value of "name" attribute denotes the name of the response header which is to be blacklisted/removed at route level. Add a new entry in next line for every header to be removed.
<code>routesConfig[0].metadata.svcName</code>	The following parameter configurable per route in route-metadata is used to track Overload Control data. If this parameter is not configured in route metadata then svc name from <code>routesConfig[0].uri</code> field is used as the required key to track Overload Control data.	The value of "svcName" attribute denotes the back-end service tag to be used as the required key (configurable per route) to track Overload Control data instead of using back-end service name from <code>routesConfig[0].uri</code> as the required key
<code>routesConfig[0].metadata.serverHeaderDetails.enabled</code>	Server header configuration if defined at Route level (irrespective of being enabled/disabled) will take precedence over the Global conf.	Default Value: Value to be updated accordingly
<code>routesConfig[0].metadata.serverHeaderDetails.errorCodeSeriesId</code>	If not defined here, value at Global level will be used as fallback. Value need to be one among "errorCodeSeriesList" resource defined.	Default Value: Value to be updated accordingly
<code>routesConfig[0].metadata.requiredTime</code>	If <code>isSbiTimerEnabled</code> is true then this is the minimum time required to process an incoming request. If the timeout falls below this then request will be rejected.	Default Value: Value to be updated accordingly

4

Uninstalling OCNWDAF

To perform an automated uninstall, run the following command:

```
helm uninstall <installation name> -n $K8_NAMESPACE
```

This command removes all the resources including the PVCs.

Sample Output:

```
[cloud-user@ocne224-cluster-bastion-1]$ helm uninstall nwdafe -
n $K8_NAMESPACE
W0404 05:04:57.056877 3860334 warnings.go:70] autoscaling/v2beta2
HorizontalPodAutoscaler is deprecated in v1.23+, unavailable   in v1.26+;
use autoscaling/v2 HorizontalPodAutoscaler
W0404 05:04:57.065008 3860334 warnings.go:70] autoscaling/v2beta2
HorizontalPodAutoscaler is deprecated in v1.23+, unavailable   in v1.26+;
use autoscaling/v2 HorizontalPodAutoscaler
release "nwdafe" uninstalled
```

To perform a manual uninstall, run the following commands:

```
helm uninstall <installation name> -n <namespace> --no-hooks
```

```
kubectl delete all --all -n $K8_NAMESPACE && kubectl delete secret --all -
n $K8_NAMESPACE
```

Caution

The `kubectl delete` command deletes all the Kubernetes objects of the specified namespace.

For troubleshooting procedures, see, *Oracle Communications Networks Data Analytics Function Troubleshooting Guide*.

4.1 Verify Uninstallation

To verify the OCNWDAF uninstallation, run the following command:

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

In case of successful uninstallation, no OCNWDAF resources are displayed in the command output.

Sample Output:

```
[cloud-user@occne224-cluster-bastion-1 ~]$ kubectl get all -n $K8_NAMESPACE  
No resources found in nwdaf-test namespace.
```

5

Fault Recovery

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

5.1 Introduction

This section describes the procedures to perform fault 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 storage's:

- MySQL NDB Cluster
- Apache Kafka

① Note

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

① 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.

5.2 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 5-1 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 cnDBTier failed in a Single Site	No	Yes	Restore MySQL from backup and ensure it is not re-installed.

Table 5-1 (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.

5.3 Prerequisites

Ensure that the following prerequisites are met before performing the fault 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 cnDBTier by scheduling regular backups. The regular backups help in fault 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

5.4 Fault Recovery Scenarios

This chapter describes the fault recovery procedures for various scenarios.

5.4.1 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](#).

5.4.2 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 cnDBTier fails in either single or multiple sites (not all sites). And if the data replication is successful, then database corruption replicates to all the cnDBTier sites and cnDBTier fails in all sites.

If corrupted database replicated to mated sites, refer to the procedure [When cnDBTier failed in a Single Site](#).

5.4.3 When cnDBTier failed in a Single Site

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

To recover database:

1. Uninstall OCNWDAF helm chart. For information about uninstalling OCNWDAF, see the section [Uninstalling OCNWDAF](#).
2. For cnDBTier fault recovery:
 - a. Create on-demand backup from mated site that has health replication with failed site. For more information about cnDBTier backup, see the **Create On-demand Database Backup** chapter in the *Oracle Communications Cloud Native Core cnDBTier, Installation, Upgrade, and Fault Recovery Guide*.
 - b. Use the backup data from mate site for restore. For more information about cnDBTier restore, see the **Restore Georeplication Failure** chapter in *Oracle Communications Cloud Native Core cnDBTier, Installation, Upgrade, and Fault Recovery Guide*.
3. Install OCNWDAF Helm chart. For more information about installing OCNWDAF, see the section [Installing OCNWDAF](#).

5.4.4 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
```

5.4.5 Site Failure

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

It is assumed that the user has cnDBTier and OCNWDAF installed on multiple sites with automatic data replication and backup enabled.

To recover the failed sites:

1. Run the Cloud Native Environment (CNE) installation procedure to install a new cluster. For more information, see *Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) Installation Guide, Upgrade, and Fault Recovery Guide*.
2. For cnDBTier fault recovery:
 - a. Take on-demand backup from the mate site that has health replication with the failed site or sites. For more information about on-demand backup, see the **Create On-demand Database Backup** chapter in the *Oracle Communications Cloud Native Core, cnDBTier Installation Guide, Upgrade, and Fault 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, cnDBTier Installation Guide, Upgrade, and Fault Recovery Guide*.
3. Install OCNWDAF Helm chart. For more information about installing OCNWDAF, see the [Installing OCNWDAF](#) section.

5.4.6 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 OCNWDAF is 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

OCNWDAF microservices 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 <zookeeper>: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 <zookeeper>: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.

5.4.7 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 5-1 Sample Output

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-584ceb8fd4-pcdn6	1/1	Running	0	4d13h
ocn-nrf-simulator-d44bcd7f-ljcz2	1/1	Running	0	4d13h
ocn-nwdaf-analytics-65bfff75c66-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-5546cf4645-12hvp	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
zookeeper-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 Function 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.

5.5 Backup and Restore

This chapter describes the Backup and Restore procedures.

5.5.1 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

OCNWDAB 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 ndbmcmd-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 ~]$
```

OCNWDAB DB Backup

If the OCNWDAB 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 nwdafe bash
mysqldump --quick -h127.0.0.1 -u dbuser -p nwdafe | 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. 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 < nwdafeBackup.sql.gz | mysql -h127.0.0.1 -u dbuser -p newnwdafe
```

4. Enter the password when prompted.

5.5.2 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 OCNWDAF database and restore the database as described in [Restoring OCNWDAF and cnDBTier](#) Perform this task once every evening.

If OCNWDAF deployment is corrupted, do the following:

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

```
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 OCNWDAF using the backed up copy of the `custom_values.yaml` file. For information about installing OCNWDAF using Helm, see [Installation Tasks](#).

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

1. Install `cnDBTier` as described in *Oracle Communication Cloud Native Core cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
2. Restore OCNWDAFDB as described in [Restoring OCNWDAF and cnDBTier](#).
3. Install OCNWDAF using the backed up copy of the `custom_values.yaml` file. For information about installing OCNWDAF, see [Installation Tasks](#).

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: "nwdafrprivilegeduser-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
```

5.5.3 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.