Oracle® Communications

Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide

Release 23.1.0
F77649-05
July 2023

ORACLE

Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide, Release 23.1.0

F77649-05

Contents

My Oracle Support (MOS)

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 and Terminology

Acronym	Definition
CLI	Command Line Interface
CNC Console	Cloud Native Configuration Console
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment (CNE)
CSP	Communication Service Provider
KPI	Key Performance Indicator
MPS	Messages Per Second
NDB	Network Data Broker
NF	Network Function
NRF	Oracle Communications Cloud Native Core, Network Repository Function (NRF)
OHC	Oracle Help Center
OSDC	Oracle Service Delivery Cloud
SCP	Oracle Communications Cloud Native Core, Service Communication Proxy (SCP)
SEPP	Oracle Communications Cloud Native Core, Security Edge Protection Proxy (SEPP)
SVC	Services
TLS	Transport Layer Security
URI	Uniform Resource Identifier

What's New in This Guide

This section lists the documentation updates for Release *23.1.x* in *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.

Release 23.1.0 - F77649-05, July 2023

- Updated the Installing OCNADD Package section.
- Updated the Environment Setup Requirements section.

Release 23.1.0 - F77649-04, May 2023

Updated the section Pushing the Images to Customer Docker Registry.

Release 23.1.0 - F77649-03, May 2023

Updated the procedure Customizing OCNADD, step number 6 added.

Release 23.1.0 - F77649-02, April 2023

- Updated the information about creating admin user in the section Configuring OCNADD Database .
- Updated the procedure Configuring SSL or TLS Certificates.

Release 23.1.0 - F77649-01, March 2023

- Updated the Pushing the Images to Customer Docker Registry section to describe the compatible docker image versions for various microservices.
- Updated the Installing OCNADD to describe the procedure for installing 23.1.0 package.
- Updated the document to describe the support for `ocnadd-custom-values.yaml` file.
- Updated the Installing OCNADD section with Kafka deployment steps.
- Updated the Creating Service Account, Role, and Role Binding section to modify the role binding information for `ocnadd-sample-rolebinding-template.yaml`.
- Updated the Installing OCNADD GUI section with the single package for UI and Backend OCNADD services installation.
- Updated the Preupgrade Tasks section with preupgrade procedure.
- Added a new chapter, Fault Recovery, to describe the various fault recovery scenarios and their solutions. The *Oracle Communications Network Analytics Data Director (OCNADD) Disaster Recovery Guide* is deprecated from this release and all content from the document is moved to this chapter.

1 Introduction

This document provides information on how to install Oracle Communications Network Analytics Data Director (OCNADD) and its microservices.

1.1 Overview

Oracle Communications Network Analytics Data Director (OCNADD) is a specialized Network Data Broker (NDB) in 5G Network Architecture.

OCNADD receives the network traffic data from various sources, 5G network functions (NFs), Non-5G NFs, or third-party producers. It performs various rules on the received data and then securely sends it to the subscribed third-party consumers (third-party consumer applications or platforms).

OCNADD ensures data security, low latency, and redundancy while collecting and processing the data. It enables Communication Service Providers (CSPs) to correlate and transform the acquired data as per their data feed configuration to create comprehensive dashboards and Key Performance Indicators (KPIs). Hence, it achieves meaningful insights about all functions in 5G Network Architecture. This information is used for providing good quality of service, reducing downtime, easing network scalability, and minimising losses. The OCNADD generated data is beneficial for monitoring and troubleshooting issues during a network failure. The OCNADD also provides GUI that enables users to create, edit and delete the datafeeds. For more information about OCNADD architecture and features, see *Oracle Communications Network Analytics Data Director User Guide*.

Installation Overview

The OCNADD installation comprises of various tasks including prerequisites, pre-installation, and installation. Perform the installation tasks in the same sequence as outlined in the following table:

Table 1-1 Installation Overview

Task	Sub task	Reference Link
Prerequisites	Software Requirements	Software Requirements
	Environment Setup Requirements	Environment Setup Requirements
	Resource Requirements	Resource Requirements
Pre-installation	Creating OCNADD Namespace	Creating OCNADD Namespace
	Creating Service Account, Role, and Role Binding	Creating Service Account, Role, and Role Binding
	Configuring OCNADD Database	Configuring OCNADD Database
	Configuring Secrets for Accessing Database	Configuring Secrets for Accessing OCNADD Database
	Configuring TLS or SSL Certificates	Configuring SSL or TLS Certificates
Installation	Downloading OCNADD Package	Downloading OCNADD Package
	Installing OCNADD	Installing OCNADD

Table 1-1 (Cont.) Installation Overview

Task	Sub task	Reference Link
	Verifying OCNADD Installation	Verifying OCNADD Installation
	OCNADD GUI Installation	Installing OCNADD GUI

1.2 References

For more information on OCNADD, refer to the following documents:

- *Oracle Communications Network Analytics Data Director User Guide*
- *Oracle Communications Network Analytics Data Director Troubleshooting Guide*
- *Oracle Communications Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native DBTier Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Network Analytics Data Director Security Guide*
- *Oracle Communications Network Analytics Data Director Benchmarking Guide*

2
Installing OCNADD

This chapter describes how to install Oracle Communications Network Analytics Data Director (OCNADD) on the supported paltforms. The OCNADD installation is supported over the following platforms:

- **Oracle Communications Cloud Native Environment (OCCNE)**
  This document descibes the OCNADD installation on OCCNE. To perform the installation on OCCNE, see, Prerequisites.
- **VMware Tanzu Application Platform (TANZU)**
  The procedure for OCNADD installation on TANZU is similar to the OCNADD installation on OCCNE. However, any steps specific to TANZU platform are mentioned explicitely in the document.

2.1 Prerequisites

Before you begin with the procedure for installing Oracle Communications Network Analytics Data Director (OCNADD), make sure 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

The following software must be installed before installing Oracle Communications Network Analytics Data Director (OCNADD):

Table 2-1 Mandatory Software

Software	Version
Kubernetes	1.22.x and 1.21.x
Helm	3.8.x
Docker/Podman	19.03.x/4.1.x

> **Note**
>
> OCNADD 23.1.0 supports OCCNE 22.4.x.

To check the Oracle Communications Cloud Native Environment (OCCNE) version, run the following command:

```
echo $OCCNE_VERSION
```

To check the current Helm and Kubernetes versions installed in OCCNE, run the following commands:

```
kubectl version
```

```
helm version
```

> **Note**
>
> Starting with OCCNE 1.8.0, podman is the preferred container platform instead of docker. For more information on installing and configuring podman, see the *Oracle Communications Cloud Native Environment Installation Guide*.

If you are installing OCNADD on TANZU, the following software must be installed:

Table 2-2 Mandatory Software

Software	Version
Tanzu	1.4.1

To check the current TANZU version, run the following command:

```
tanzu version
```

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

Table 2-3 Additional Software

Software	Version	Required For
Prometheus-Operator	2.36.1	Metrics
Metallb	0.12.1	LoadBalancer
CNDBTier	22.4.x	MYSQL Database

> **Note**
>
> The softwares are available by default, if OCNADD is deployed in Oracle Communications Cloud Native Environment (OCCNE). If you are deploying OCNADD in any other environment, for instance, TANZU, the above-mentioned software must be installed before installing OCNADD.

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

```
helm ls -A
```

2.1.2 Environment Setup Requirements

This section provides information on environment setup requirements for installing Oracle Communications Network Analytics Data Director (OCNADD).

Network Access

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

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

  ```
  helm repo update
  ```

- Service FQDN or IP Addresses of the required OCNADD services, for instance, Kafka Brokers must be discoverable from outside of the cluster, which is publicly exposed so that Ingress messages to OCNADD can come from outside of Kubernetes.

Client Machine Requirements

> **Note**
>
> Run all the `kubectl` and `helm` commands in this guide on a system depending on the infrastructure and deployment. It could be a client machine, such as a virtual machine, server, local desktop, and so on.

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.
- configured helm repository
- network access to the Kubernetes cluster.
- required environment settings to run the `kubectl`, `podman`, and `docker` commands. The environment should have privileges to create namespace in the Kubernetes cluster.
- The helm client installed with the **push** plugin. Configure the environment in such a manner that the `helm install` command deploys the software in the Kubernetes cluster.

Server or Space Requirements

For information on the server or space requirements for installing OCNADD on OCCNE, see the following documents:

- *Oracle Communications Cloud Native Environment Installation Guide*
- *Oracle Communications Network Analytics Data Director Planning Guide*
- *Oracle Communications Cloud Native cnDBTier Installation Guide*

OCNADD GUI Requirements

For the OCNADD GUI to load successfully, the following URLs should be accessible from where the user is accessing OCNADD GUI:

- https://static.oracle.com
- https://static-stage.oracle.com

Secret File Requirements

> ⚠ **Caution**
>
> Users should provide their own `CAcert.pem` and `CAkey.pem` for generating certificates for the OCNADD SSL or TLS support.

For HTTPs, the certificates must be created before creating secret files for Keys and MySQL database credentials.

For more information about creating certificates, see Configuring SSL or TLS Certificates.

ServiceAccount Requirement

ServiceAccount is mandatory and it is created by default along with helm chart installation for OCNADD services. You can also create a service account as described in Creating Service Account, Role, and Role Binding.

cnDBTier Requirement

OCNADD supports cnDBTier 22.4.x in a CNE environment. cnDBTier must be up and running in case of containerized Cloud Native Environment. For more information about the installation procedure, see *Oracle Communications Cloud Native Core cnDBTier Installation Guide*.

OCNADD Images

The following table lists Data Director microservices and their corresponding images:

Table 2-4 OCNADD images

Microservices	Image	Tag
OCNADD-Configuration	ocnaddconfiguration	23.1.0
OCNADD-ConsumerAdapter	ocnaddconsumeradapter	23.1.0
OCNADD-AGG	ocnaddnrfaggregation ocnaddscpaggregation ocnaddseppaggregation	23.1.0
OCNADD-Alarm	ocnaddalarm	23.1.0
OCNADD-HealthMonitoring	ocnaddhealthmonitoring	23.1.0
OCNADD-Kafka	kafka-broker-x	23.1.0
OCNADD-Admin	ocnaddadminservice	23.1.0
OCNADD-Backendrouter	ocnaddbackendrouter	23.1.0
OCNADD-GUI	ocnaddgui	23.1.0

Note

The service images are prefixed with the OCNADD release name.

2.1.3 Resource Requirements

This section describes the resource requirements to install and run Oracle Communications Network Analytics Data Director (OCNADD).

OCNADD Resource Requirements

Table 2-5 OCNADD Resource Requirements

OCNADD Service	vCPU Req	vCPU Limit	Memory Req (Gi)	Memory Limit (Gi)	Min Replica	Max Replica	Partitions	Topic Name
ocnaddconfiguration	1	1	1	1	1	1		
ocnaddalarm	1	1	1	1	1	1		
ocnaddadmin	1	1	1	1	1	1		
ocnaddhealthmonitoring	1	1	1	1	1	1		
ocnaddbackendrouter	1	1	1	1	1	1		
ocnaddscpaggregation	1	3	1	2	1	2	12	SCP
ocnaddnrfaggregation	1	3	1	2	1	1	6	NRF
ocnaddseppaggregation	1	3	1	2	1	1	6	SEPP
ocnaddadapter	2	3	3	4	2	8	72	MAIN
ocnaddkafka	2	5	4	10	3	3		
zookeeper	1	1	1	1	3	3		

Table 2-5 (Cont.) OCNADD Resource Requirements

OCNADD Service	vCPU Req	vCPU Limit	Memory Req (Gi)	Memory Limit (Gi)	Min Replica	Max Replica	Partitions	Topic Name
ocnaddgui	1	2	1	1	1	2		

Ephemeral Storage Requirements

Table 2-6 Ephemeral Storage

Service Name	Ephemeral Storage (min) in Mi	Ephemeral Storage (max) in Mi
<app-name>-adapter	200	800
ocnaddadminservice	100	200
ocnaddalarm	100	500
ocnaddhealthmonitoring	100	500
ocnaddscpaggregation	100	500
ocnaddseppaggregation	100	500
ocnaddnrfaggregation	100	500
ocnaddconfiguration	100	500

2.2 Installation Sequence

This section provides information on how to install Oracle Communications Network Analytics Data Director (OCNADD). The steps are divided into two categories:

- Pre-Installation Tasks
- Installation Tasks

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

2.2.1 Pre-Installation Tasks

To install OCNADD, perform the preinstallation steps described in this section.

> **Note**
>
> The `kubectl` commands may vary based on the platform used for deploying OCNADD. 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 OCCNE's version of kube-api server.

2.2.1.1 Creating OCNADD Namespace

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

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

```
kubectl get namespaces
```

If the namespace exists, you may continue with the next steps of installation.

If the required namespace is not available, create a namespace using the following command:

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

Example

```
kubectl create namespace ocnadd_namespace
```

Naming Convention for Namespaces

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

- starts and ends with an alphanumeric character
- contains 63 characters or less
- contains only alphanumeric characters or '-'

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

2.2.1.2 Creating Service Account, Role, and Role Binding

This section is optional and it describes how to manually create a service account, role, and rolebinding. It is required only when customer needs to create a role, rolebinding, and service account manually before installing OCNADD.

> **Important**
>
> The secret(s) should exist in the same namespace where OCNADD is getting deployed. This helps to bind the Kubernetes role with the given service account.

Creating Service Account, Role, and RoleBinding

To create the service account, role, and rolebinding:

1. Create an OCNADD resource file:

   ```
   vi <ocnadd-resource-file>
   ```

   Where,

   `<ocnadd-resource-file>` is the name of the resource file.

Example:

```
vi ocnadd-resource-template.yaml
```

2. Update the `ocnadd-resource-template.yaml` with release specific information:

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

A sample template to update the `ocnadd-resource-template.yaml` file with is given below:

```
## Sample template start#
apiVersion: v1
kind: ServiceAccount
metadata:
    name: <custom name>-sa-ocnadd
    namespace: <namespace>
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <custom name>-cr
rules:
- apiGroups: [""]
  resources:
  - pods
  - services
  - configmaps
  verbs: ["get", "list", "watch"]
- apiGroups: ["policy"]
  resources: ["podsecuritypolicies"]
  verbs: ["use"]
  resourceNames: ["privileged"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
   name: <namespace>-crb
   namespace: <namespace>
roleRef:
   apiGroup: rbac.authorization.k8s.io
   kind: ClusterRole
   name: <custom-name>-cr
subjects:
- kind: ServiceAccount
  name: <custom-name>-sa-ocnadd
  namespace: <namespace>
---
## Sample template end#
```

3. Run the following command to create service account, role, and rolebinding:

   ```
   $ kubectl -n <namespace> create -f ocnadd-resource-template.yaml
   ```

   Where,

   <namespace> is the namespace where OCNADD is deployed.

   Example:

   ```
   $ kubectl -n ocnadd create -f ocnadd-resource-template.yaml
   ```

> **Note**
>
> Once the global service account is added, users must update the following parameters to **false** in the ocnadd-custom-values.yaml file; otherwise, installation may fail as a result of creating and deleting custom resource definitions (CRD):
>
> ```
> serviceAccount:
>     create: false
> clusterRole:
>     create: false
> clusterRoleBinding:
>     create: false
> ```

2.2.1.3 Configuring OCNADD Database

OCNADD microservices use MySQL database to store the configuration and run time data.

The database is managed by the helm pre-install hook. However, OCNADD requires the database administrator to create an admin user in MySQL database and provide the necessary permissions to access the databases. Before installing OCNADD it is required to create the MySQL user and databases.

> **Note**
>
> - If the admin user is already available, then update the credentials, such as username and password (base64 encoded) in ocnadd/templates/ocnadd-secret-hook.yaml.
> - If the admin user is not available, then create it using the following procedure. Once the user is created, update the credentials for the user in ocnadd/templates/ocnadd-secret-hook.yaml.

Creating Database

To create database:

1. Run the following command to log in to MySQL pod.

> **Note**
>
> Use the namespace in which the cnDBTier is deployed. For example, `occne-cndbtier` namespace is used. The default container name is `mysqlndbcluster`

```
$ kubectl -n occne-cndbtier exec -it ndbmysqld-0 -- bash
```

To verify all the available containers in the pod, run:

```
Use 'kubectl describe pod/ndbmysqld-0 -n occne-cndbtier'
```

2. Run the following command to login to MySQL server using MySQL client:

```
$ mysql -h 127.0.0.1 -uroot -p
$ Enter password:
```

3. To create a admin user, run the following command:

```
CREATE USER IF NOT EXISTS'<ocnadd admin username>'@'%' IDENTIFIED BY
'<ocnadd admin user password>';
```

Example:

```
CREATE USER IF NOT EXISTS 'ocdd'@'%' IDENTIFIED BY 'ocdd';
```

where:

<ocdd> is the admin username and <ocdd> is the password for MySQL admin user

4. Run the following command to grant the necessary permissions to the admin user and run the FLUSH command to reload the grant table:

```
GRANT ALL PRIVILEGES ON *.* TO 'ocdd'@'%' WITH GRANT OPTION;


FLUSH PRIVILEGES;
```

5. Access the `ocnadd-secret-hook.yaml` from the OCNADD helm files using the following path:

```
ocnadd/templates/ocnadd-secret-hook.yaml
```

.

6. Update the following parameters in the `ocnadd-secret-hook.yaml` with the admin user credentials:

```
data:
MYSQL_USER: b2NkZA==
MYSQL_PASSWORD: b2NkZA==
```

To generate the base64 encoded user and password from the terminal, run the following command:

```
echo -n <string> | base64 -w 0
```

Where, `<string>` is the admin username or password created in step3.

For example:

```
echo -n ocdd | base64 -w 0
b2NkZA==
```

Update Database Name

> **Note**
>
> - By default, the database names are configuration_schema, alarm_schema, and healthdb_schema for the respective services.
> - Skip this step if you plan to use the default database names during database creation. If not, change the database names as required.

To update the database names in the Configuration Service, Alarm Service, and Health Monitoring services:

1. Access the `ocdd-db-resource.sql` file from the helm chart using the following path:

   ```
   ocnadd/ocdd-db-resource.sql
   ```

2. Update all occurrences of the database name in `ocdd-db-resource.sql`.

> **Note**
>
> During the OCNADD re-installation, all three application databases must be removed manually by running the `drop database <dbname>;` command.

2.2.1.4 Configuring Secrets for Accessing OCNADD Database

The secret configuration for OCNADD database is automatically managed during the database creation the helm pre-install procedure.

2.2.1.5 Configuring SSL or TLS Certificates

> **Note**
>
> Before configuring the SSL/TLS certificates see, the "Customizing CSR and Certificate Extensions" section in the *Oracle Communications Network Analytics Suite Security Guide*.

> **Note**
>
> This is a mandatory procedure, perform this procedure before you proceed with installation.

Before generating certificates using CACert and CAKey, the Kafka access mode needs to be finalized, and the ssl_certs/default_values/values to be updated accordingly.

The following access modes are available:

- When the NF producers and OCNADD are in same cluster
  - with external access disabled
  - with external access enabled
- The NF producers and OCNADD are in different cluster
  - with LoadBalancer
  - with NodePort

> **Note**
>
> - If the NF Producers and OCNADD are deployed in same cluster, all three ports can be used that is, 9092 for PLAIN_TEXT, 9093 for SSL, and 9094 for SASL_SSL. However, the 9092 port is non-secure and hence not recommended to use.
> - If the NF Producers and OCNADD are deployed in different cluster, only the 9094 (SASL_SSL) port is exposed
> - It is recommended to use the individual server IPs in the Kafka bootstrap server list instead of single service IP like "kafka-broker:9094".

The NF producers and OCNADD are in the same cluster

- **With external access disabled**

In this mode, the Kafka cluster is not exposed externally. By default, the parameters externalAccess.enabled and externalAccess.autoDiscovery are set to false, therefore no change is needed. The parameters externalAccess.enabled and externalAccess.autoDiscovery are present in the ocnadd-custom-values.yaml file.

The default values of bootstrap-server are given below:

```
kafka-broker-0.kafka-broker:9093
kafka-broker-1.kafka-broker:9093
kafka-broker-2.kafka-broker:9093
```

- **With external access enabled**

This mode can be chosen when the customer wants to have different service names for each Kafka broker. This requires an update of the following parameters of Kafka section in `ocnadd-custom-values.yaml` file:

1. *externalAccess.type* to ClusterIP

2. *externalAccess.enabled* to true
3. *externalAccess.autoDiscovery* to true

The AdvertiseListeners in Kafka will be updated with *<kafka-broker-0-external>:<Port>,<kafka-broker-1-external>:<Port>* for each respective brokers.

Based on the AdvertisedListeners, the client bootstrap server list should be updated. Examples are given below:

```
kafka-broker-0-external:9094
kafka-broker-1-external:9094
kafka-broker-2-external:9094
```

The NF producers and OCNADD are in different cluster

If the NF producers and OCNADD are in different Cluster, then either the LoadBalancer or NodePort Service Type can be used. In both the cases, the IP addresses are required to be updated manually in the ssl_certs/default_values/values of kafka-broker section by using the following steps:

With LoadBalancer

1. Update the following parameters in Kafka section of the `ocnadd-custom-values.yaml` file:
   a. *externalAccess.type* to LoadBalancer
   b. *externalAccess.enabled* to true
   c. *externalAccess.autoDiscovery* to true
2. Update based on LoadBalance IP types as follows
   a. **When Static LoadBalancer IPs are used**
      i. Update the following parameters in the Kafka section of the `ocnadd-custom-values.yaml` file:
         - *externalAccess.setstaticLoadBalancerIps* to 'true'. Default is false.
         - Static IP list in "externalAccess.LoadBalancerIPList" separated with comma.

         For example:

         ```
         externalAccess:
                     setstaticLoadBalancerIps: true
                     LoadBalancerIPList:
         [10.20.30.40,10.20.30.41,10.20.30.42]
         ```

      ii. Add all the static IP's in ssl_certs/default_values/values under kafka-broker section.
         Example: for the static IP list "10.20.30.40,10.20.30.41,10.20.30.42"

         ```
         [kafka-broker]
         client.commonName=kafka-broker-zk
         server.commonName=kafka-broker
         DNS.1=*.kafka-broker.<nameSpace>.svc.<Cluster-Domain>
         DNS.2=kafka-broker
         DNS.3=*.kafka-broker
         ```

```
IP.1=10.20.30.40
IP.2=10.20.30.41
IP.3=10.20.30.42
```

b. **When LoadBalancer IP CIDR block is used**

i. Get the available LoadBalancer IP CIDR block using the following command `kubectl describe cm -n occne-infra occne-metallb`

For example:

```
address-pools:
- addresses:   -
    10.20.30.0/26
```

ii. Add all the available IP's in ssl_certs/default_values/values under kafka-broker section.
Example: for the "10.x.x.0/26" IP range

```
[kafka-broker]
client.commonName=kafka-broker-zk
server.commonName=kafka-broker
DNS.1=*.kafka-broker.<nameSpace>.svc.<Cluster-Domain>
DNS.2=kafka-broker
DNS.3=*.kafka-broker
IP.1=10.x.x.1
IP.2=10.x.x.2
.
.
IP.4=10.x.x.63
```

With NodePort

> **Note**
>
> **Not Supported for CNE 22.4.0 and above releases.**
>
> From CNE 22.4.0 onwards, the worker node External IP is not exposed. If user has them explicitly exposed, use the Node Port option else, use the LoadBalancer for external access.

1. Update the following parameters in the Kafka section of the `ocnadd-custom-values.yaml` file:
   a. *externalAccess.type* to NodePort
   b. *externalAccess.enabled* to true
   c. *externalAccess.autoDiscovery* to true
2. Get the available list of EXTERNAL-IP Node IPs, run the following command : `kubectl get no -o wide`
3. Add all the Node IP's in ssl_certs/default_values/values under kafka-broker section.

Example: For Node IPs 10.20.30.40, 10.20.30.50, and so on.

```
[kafka-broker]
client.commonName=kafka-broker-zk
server.commonName=kafka-broker
DNS.1=*.kafka-broker.<nameSpace>.svc.<Cluster-Domain>
DNS.2=kafka-broker
DNS.3=*.kafka-broker
IP.1=10.20.30.40
IP.2=10.20.30.50
```

2.2.1.5.1 Generate Certificates using CACert and CAKey

OCNADD allows the users to provide the CACert and CAKey and generate certificates for all the services by running a predefined script.

To generate certificates using CACert and CAKey:

1. Navigate to the `ssl_certs/default_values/values` file.
2. In the `values` file, edit the global parameters, CN, and SAN for each service based on the requirement as follows:

> **Note**
>
> Edit only values for global parameters and RootCA common name, and add service blocks for all the services for which certificate needs to be generated. The `values` will be available once the OCNADD helm files are extracted.

```
Global Params:
[global]
countryName=<country>
stateOrProvinceName=<state>
localityName=<city>
organizationName=<org_name>
organizationalUnitName=<org_bu_name>
defaultDays=<days to expiry>


Root CA common name (e.g., *.namespace.svc.domainName)
##root_ca
commonName=<rootca_common_name>


Service common name for client and server and SAN. (Make sure to follow
exact same format and provide an empty line at the end of each service
block)

[service-name-1]
client.commonName=client.cn.name.svc1
server.commonName=server.cn.name.svc1
IP=127.0.0.1
DNS.1=localhost
```

```
[service-name-2]
client.commonName=client.cn.name.svc2
server.commonName=server.cn.name.svc2
IP = 10.20.30.40
DNS.1 = *.svc2.namespace.svc.domainName

[service-name-3]
client.commonName=client.cn.name.svc3

[service-name-4]
server.commonName=server.cn.name.svc4
IP.1 = 10.20.30.41
IP.2 = 127.0.0.1
DNS.1 = *.svc4.namespace.svc.domainName
DNS.2 = *.svc44.namespace.svc.domainName

##end
```

3. Run the `generate_certs.sh` script with the following command:

   ```
   ./generate_certs.sh -cacert <path to>/CAcert.pem -cakey <path to>/CAkey.pem
   ```

4. Select “n” when prompted for create Certificate Authority (CA).

   ```
   Do you want to create Certificate Authority (CA)? n
   ```

5. Copy CA Certificate pem file (as `cacert.pem`) to “demoCA” folder and CA certificate key file (as `cakey.pem`) to “demoCA/private” if the paths to cacert and cakey are not provided through flags.
   (The demoCA folder is created by script in the same path where script exist.)

   ```
   cp /path/to/CAcert.pem /path/to/generate_certs_script/demoCA/cacert.pem
   cp /path/to/CAkey.pem /path/to/generate_certs_script/demoCA/private/
   cakey.pem
   ```

   > **Note**
   >
   > Perform this step only if you have not provided the paths to cacert and cakey.

6. Select “y” when prompted to use the existing CA to sign CSR for each service.

   ```
   Would you like to use existing CA to sign CSR for services? Y
   ```

7. Enter the password for your CA key.

   ```
   password: <enter your ca key password>
   ```

8. Select “y” when prompted to create CSR for each service.

   ```
   Create Certificate Signing Request (CSR) for each service? Y
   ```

9. Select “y” when prompted for signing CSR for each service with CA Key.

   ```
   Would you like to sign CSR for each service with CA key? Y
   ```

10. Select “y” if you would like to create secrets for each service in existing namespace or “n” if you want to create secrets in a new namespace.

    ```
    If “n”
    a.  Would you like to choose any above namespace for creating secrets
    (y/n) n
    b.  Enter new Kubernetes Namespace to create: <name of new ns to create>
    If “y”
    c.  Would you like to choose any above namespace for creating secrets
    (y/n) y
    d.  Enter new Kubernetes Namespace to create: <name of existing ns>
    ```

    The certificates are generated for each service and are available in the `demoCA/services` folder. The secret is created in the namespace, which is specified during the secret creation process.

11. Run the following command to check if the secrets are created in the specified namespace.

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

12. Run the following command to describe any secret created by script.

    ```
    kubectl describe secret <secret-name> -n <namespace>
    ```

2.2.1.5.2 Generate Certificate Signing Request (CSR)

Users can generate the certificate signing request for each of the services using the OCNADD script, and then can use the generated CSRs to generate the certificates using its own certificate signing mechanism (External CA server, Hashicorp Vault, and Venafi).

Perform the following procedure to generate the CSR:

1. Navigate to the `ssl_certs/default_values/values` file.
2. Edit global parameters, CN, and SAN for each service based on the requirement.

> **Note**
>
> Edit only values for global parameters and RootCA common name, and add service blocks for all the services for which certificate needs to be generated.

```
a.  Global Params:
[global]
countryName=<country>
stateOrProvinceName=<state>
localityName=<city>
organizationName=<org_name>
organizationalUnitName=<org_bu_name>
defaultDays=<days to expiry>
```

```
b.  Root CA common name (e.g., *.namespace.svc.domainName)
##root_ca
commonName=<rootca_common_name>

c.  Service common name for client and server and SAN. (Make sure to
follow exact same format and provide an empty line at the end of each
service block)

[service-name-1]
client.commonName=client.cn.name.svc1
server.commonName=server.cn.name.svc1
IP=127.0.0.1
DNS.1=localhost

[service-name-2]
client.commonName=client.cn.name.svc2
server.commonName=server.cn.name.svc2
IP = 10.20.30.40
DNS.1 = *.svc2.namespace.svc.domainName

[service-name-3]
client.commonName=client.cn.name.svc3

[service-name-4]
server.commonName=server.cn.name.svc4
IP.1 = 10.20.30.41
IP.2 = 127.0.0.1
DNS.1 = *.svc4.namespace.svc.domainName
DNS.2 = *.svc44.namespace.svc.domainName

##end
```

3. Run the `generate_certs.sh` script with the `--gencsr` or `-gc` flag.

   ```
   ./generate_certs.sh --gencsr
   ```

4. Navigate to CSR and keys in the `demoCA/services` (separate for client and server). The CSR can be signed using your own certificate signing mechanism and certificates should be generated.
5. Make sure that the certificates and keys naming is in the following format if the service is acting as client or server, or both.

   ```
   For Client
   servicename-clientcert.pem and servicename-clientprivatekey.pem
   For Server
   servicename-servercert.pem and servicename-serverprivatekey.pem
   ```

6. Copy the certificates in the respective `demoCA/services` folder after certificates are generated for each service by signing CSR with your own CA key.
   The certificates should be separate for client and server as their CSR are generated.

7. Run `generate_certs.sh` with the `cacert` path and `--gensecret` or `-gs` to generate secrets.

   ```
   ./generate_certs.sh -cacert /path/to/cacert.pem --gensecret
   ```

8. Enter "y" to continue generating secrets.

   ```
   Would you like to continue to generate secrets? (y/n) y
   ```

9. Select “y” if you want to create secrets for each service in the existing namespace or “n” if you want to create secrets in a new namespace.

   ```
   If “n”
   >    Would you like to choose any above namespace for creating secrets
   (y/n) n
   >    Enter new Kubernetes Namespace to create: <name of new ns to create>
   If “y”
   >    Would you like to choose any above namespace for creating secrets
   (y/n) y
   >    Enter new Kubernetes Namespace to create: <name of existing ns>
   ```

   The secret is created in the namespace, which is specified during the secret creation process.

10. Run the following command to check if the secrets are created in the specified namespace:

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

11. Run the following command to describe any secret created by script:

    ```
    kubectl describe secret <secret-name> -n <namespace>
    ```

2.2.1.5.3 Generate Certificates and Private Keys

Users can generate the certificates and private keys for all the required services, and then create Kubernetes secrets without using the OCNADD script.

Perform the following procedure to generate the certificates and private keys:

1. Run the `openssl` command to generate CSR for each service (separate for client and server if required).
   a. Run the following command to generate private key:

      ```
      openssl req -x509 -nodes -sha256 -days 365 -newkey rsa:2048 -keyout
      rsa_private_key -out rsa_certificate.crt
      ```

   b. Run the following command to convert private key to pem:

      ```
      openssl rsa -in rsa_private_key -outform PEM -out
      rsa_private_key_pkcs1.pem
      ```

   c. Update CN, SAN, and global parameters for each service in the `openssl.cnf` file.

d. Run the following command to generate CSR for each service using private key:

```
openssl req -new -key rsa_private_key -out service_name.csr -config
ssl.conf
```

2. Sign each service CSR with Root CA private key to generate certificates.
3. Generate Secrets using each service certificates and keys.
   a. Run the following command to create truststore and keystore password files:

```
echo "<password>" >> trust.txt
echo "<password>" >> key.txt
```

   b. Run the following command to create secrets using client and server certificates and cacert:

```
kubectl create secret generic <service_name>-secret --from-file=path/to/
cert/<service_name>-clientprivatekey.pem --from-file=path/to/cert/
<service_name>-clientcert.pem --from-file=path/to/cacert/cacert.pem  --
from-file=path/to/cert/<service_name>-serverprivatekey.pem --from-
file=path/to/cert/<service_name>-servercert.pem  --from-file=trust.txt
--from-file=key.txt --from-literal=javakeystorepass=changeit -n
<namespace>
```

> **Note**
>
> Repeat Step 1 and 2 for all services (separate for client and server).

2.2.2 Installation Tasks

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

> **Note**
>
> Before starting the installation tasks, ensure that the Prerequisites and Pre-Installation Tasks arec completed.

2.2.2.1 Downloading OCNADD Package

To download the Oracle Communications Network Analytics Data Director (OCNADD) package from MOS, perform the following steps:

1. Log in to My Oracle Support with your credentials.
2. Select the **Patches and Updates** tab to locate the patch.
3. In the **Patch Search** window, click **Product or Family (Advanced)**.
4. Enter "Oracle Communications Cloud Native Core - 5G" in the **Product** field, select "Oracle Communications Network Analytics Data Director 23.1.0.0.0" from **Release** drop-down list.
5. Click **Search**. The **Patch Advanced Search Results** displays a list of releases.

6. Select the required patch from the search results. The Patch Details window opens.
7. Click **Download**. File Download window appears.
8. Click the **<p********_<release_number>_Tekelec>.zip** file to download the OCNADD package file.
9. Extract the zip file to download the network function patch to the system where the network function must be installed.

2.2.2.2 Pushing the Images to Customer Docker Registry

Important

kubectl commands might vary based on the platform deployment. Replace kubectl with Kubernetes environment-specific command line tool to configure Kubernetes resources through kube-api server. The instructions provided in this document are as per the Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) version of kube-api server.

Docker Images

Oracle Communications Network Analytics Data Director (OCNADD) deployment package includes ready-to-use docker images and helm charts to help orchestrate containers in Kubernetes. The communication between Pods of services of OCNADD are preconfigured in the helm charts.

Following table lists the Docker images of OCNADD:

Table 2-7 Docker Images for OCNADD

Service Name	Docker Image Name	Image Tag
OCNADD-Configuration	ocnaddconfiguration	2.2.3
OCNADD-ConsumerAdapter	<app-name>-adapter	2.2.6
OCNADD-AGG	ocnaddnrfaggregation ocnaddscpaggregation ocnaddseppaggregation	2.1.5
OCNADD-Alarm	ocnaddalarm	2.1.3
OCNADD-HealthMonitoring	ocnaddhealthmonitoring	2.1.4
OCNADD-Kafka	kafka-broker-x	3.4.0:2.0.5
OCNADD-Admin	ocnaddadminservice	2.2.3
OCNADD-UIRouter	ocnadduirouter	23.2.75
OCNADD-GUI	ocnaddgui	23.6.91
OCNADD-CACHE	ocnaddcache	1.1.4
OCNADD-BACKUP-RESTORE	ocnaddbackuprestore	1.0.9

Note

The service image names are prefixed with the OCNADD release name.

> **Note**
>
> The above table depicts the default OCNADD microservices and their respective images. However, a few more necessary images are delivered as a part of the OCNADD package, you must push these images along with the default images.

Pushing Docker Images

To push the images to the registry:

1. Untar the OCNADD package zip file to retrieve the OCNADD docker image tar file.

   ```
   tar -xvzf ocnadd-pkg-23.2.0.0.0.tgz
   ```

   The directory consists of the following:

   - **OCNADD Docker Images File:**

     `ocnadd-images-23.2.0.tar`

   - **Helm File:**

     `ocnadd-23.2.0.tgz`

   - **Readme txt File:**

     `Readme.txt`

2. Run one of the following commands to load the `ocnadd-images-23.2.0.tar` file:

   ```
   docker load --input /IMAGE_PATH/ocnadd-images-23.2.0.tar
   ```

   ```
   podman load --input /IMAGE_PATH/ocnadd-images-23.2.0.tar
   ```

3. Run one of the following commands to verify if the images are loaded:

   ```
   docker images
   ```

   ```
   podman images
   ```

   Verify the list of images shown in the output with the list of images shown in the table Table 2-7. If the list does not match, reload the image tar file.

4. Run one of the following commands to tag each imported image to the registry:

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

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

5. Run one of the following commands to push the image to the registry:

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

```
podman 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.

6. Push the helm charts to the helm repository. Run the following command:

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

2.2.2.3 Installing OCNADD Package

This section describes how to install the Oracle Communications Network Analytics Data Director (OCNADD) package.

To install the OCNADD package, perform the following steps:

Create OCNADD Namespace

Create the OCNADD namespace, if not already created, using the following command:

```
kubectl create ns <dd-namespace-name>
```

For more information, see Creating OCNADD Namespace.

Generate Certificates

1. Run the following commands to generate certificates:

```
Change directory to <chart_path>/ssl_certs, and updat the file permission
as below

$ chmod 775 generate_certs.sh

$ chmod 775 generate_secrets.sh

(optional) Clean up the EOF encoding if copied from windows.

sed -i -e 's/\r$//' default_values/values
sed -i -e 's/\r$//' template/ca_openssl.cnf
sed -i -e 's/\r$//' template/services_server_openssl.cnf
sed -i -e 's/\r$//' template/services_client_openssl.cnf
sed -i -e 's/\r$//' generate_certs.sh
sed -i -e 's/\r$//' generate_secrets.sh
```

> **Note**
>
> Make sure that changes made in `default_values` reflect the namespace and cluster as described in *Configuring SSL or TLS Certificates* section. For more information on the certificate generation process, see Configuring SSL or TLS Certificates.

2. Perform the steps defined in Configuring SSL or TLS Certificates section to complete the certificate generation.

Update Database Parameters

To update the database parameters, see Configuring OCNADD Database.

Update ocnadd-custom-values.yaml

Update the `ocnadd-custom-values.yaml` (depending on the type of deployment model) with the required parameters. For more information on how to access and update the `ocnadd-custom-values.yaml` files, see Customizing OCNADD.

Configure OCNADD Backup Cronjob

1. Configure the `mysqlNameSpace` and `storageClass` details in `ocnadd-custom-values.yaml`.

```
cluster:
        secret:
            name: db-secret
        mysqlNameSpace:
            name: occne-cndbtierone    #---> the namespace in which the
dbtier is deployed
        mysqlPod: ndbmysqld-0          #---> the pod can be ndbmysqld-0 or
ndbmysqld-1 based on the dbTier deployment
        storageClass: standard         #---> Update the "storageClassName"
with  the respective storage class name in the case if deployment on
Tanzu  platform. For example "zfs-storage-policy"
```

2. Configure the following parameters in the `ocnadd-custom-values.yaml` file.
   The values for BACKUP_DATABASES can be set to `ALL`, which includes healthdb_schema, configuration_schema, and alarm_schema, or to the individual database names. The values for BACKUP_ARG can be set to `ALL, DB or` KAFKA. By default, the value is as `ALL`. PURGE_DAYS sets the backup retention period. The default value is 7 days.

   Example:

```
ocnaddbackuprestore:
   ocnaddbackuprestore:
       name: ocnaddbackuprestore
       env:
           BACKUP_STORAGE: 20Gi
           BACKUP_CRONEXPRESSION: "0 8 * * *"
           BACKUP_DATABASES: ALL
           BACKUP_ARG: ALL
```

Once the deployment is successful, the cronjob is spawned based on the `CRONEXPRESSION` mentioned in the `ocnadd-custom-values.yaml` file.

For more information on backup and restore, see Fault Recovery.

Install Helm Chart

Run any of the following `helm install` commands:

- In the case of Helm 2:

```
helm install <helm-repo> --name <deployment_name> -f ocnadd-custom-values.yaml --
namespace
      <namespace_name> --version <helm_version>
```

- In the case of Helm 3 and helm repo is used:

```
helm3 install <release name> -f ocnadd-custom-values.yaml --namespace <namespace>
      <helm-repo>/chart_name --version <helm_version>
```

- In case charts are extracted and Helm is used:

```
helm install <release name> -f ocnadd-custom-values.yaml --namespace <namespace>
      <helm_chart>
```

where:

helm_chart is the location of the helm chart extracted from `ocnadd-23.1.0.tgz` file

release name is the release name used by helm command.

> **Note**
>
> The release_name should not exceed 63 character limit.

namespace is the deployment namespace used by helm command.

Example:

```
helm install ocnadd-23.1.0 -f ocnadd-custom-values.yaml --namespace ocnadd-
deploy ocnadd
```

> **Caution**
>
> Do not exit from `helm install` command manually. After running the `helm install` command, it takes some time to install all the services. In the meantime, you must not press **Ctrl+C** to come out from the command.. It leads to some anomalous behavior.

> **Note**
>
> You can verify the installation while running the install command by entering this command on a separate terminal:
>
> ```
> watch kubectl get jobs,pods -n release_namespace
> ```

2.2.2.4 Verifying OCNADD Installation

This section describes how to verify if Oracle Communications Network Analytics Data Director (OCNADD) is installed successfully.

To check the status of OCNADD deployment, perform the following task:

1. In the case of Helm, run one of the following commands:

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

   Example:

   ```
   helm list -n ocnadd
   ```

   The system displays the status as deployed if the deployment is successful.
2. Run the following command to check whether all the services are deployed and active:

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

   Run the following command to check whether all the pods are up and active:

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

   Example:

   ```
   kubectl -n ocnadd get pods
   ```

   ```
   kubectl -n ocnadd get services
   ```

> **Note**
>
> - All microservices status must be Running and Ready.
> - Take a backup of the following files that are required during fault recovery:
>   - Updated Helm charts
>   - Secrets, certificates, and keys that are used during the installation
> - If the installation is not successful or you do not see the status as **Running** for all the pods, perform the troubleshooting steps. For more information, refer to *Oracle Communications Network Analytics Data Director Troubleshooting Guide*.

2.2.2.5 Creating OCNADD Topics

To create OCNADD Kakfa topics, see the "Creating Kafka Topic for OCNADD" section of *Oracle Communications Network Analytics Data Director User Guide*

2.2.2.6 Installing OCNADD GUI

This section describes how to install Oracle Communications Network Analytics Data Director (OCNADD) GUI using the following steps:

- Install OCNADD GUI
- Configure OCNADD GUI in CNC Console
- Access OCNADD GUI

Install OCNADD GUI

The OCNADD GUI gets installed along with the OCNADD services.

Configure OCNADD GUI in CNCC

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

Before installing CNCC, ensure to update the instances parameters with the following details in the `occncc_custom_values.yaml` file:

```
instances:
  - id: Cluster1-dd-instance1
    type: DD-UI
    owner: Cluster1
    ip: 10.xx.xx.xx    #--> give the cluster/node IP
    port: 31456        #--> give the node port of ocnaddgui
    apiPrefix: /occne-12ipcluster/ocnadd
  - id: Cluster1-dd-instance1
    type: DD-API
    owner: Cluster1
    ip: 10.xx.xx.xx   #--> give the cluster/node IP
    port: 32406       #--> give the node port of ocnaddbackendrouter
    apiPrefix: /occne-12ipcluster/ocnaddapi
```

```
# Applicable only for Manager and Agent core. Used for Multi-Instance-Multi-
Cluster Configuration Validation
  validationHook:
    enabled: false   #--> add this enabled: false to validationHook

#--> do these changes under section :
cncc iam attributes
# If https is disabled, this Port would be HTTPS/1.0 Port (secured SSL)
    publicHttpSignalingPort: 30085  #--> CNC console nodeport

#--> add these lines under cncc-iam attributes
# 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  #--> CNC console nodeport
    staticHttpsNodePort: 30053

#--> do these changes under section : manager cncc core attributes
#--> add these lines under mcncc-core attributes

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

#--> do these changes under section : agent cncc core attributes
#--> add these lines under acncc-core attributes
# 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: 30076
    staticHttpsNodePort: 30044
```

If CNCC is already installed, ensure to upgrade it with the following parameters updated in the `occncc_custom_values.yaml` file:

```
instances:
  - id: Cluster1-dd-instance1
    type: DD-UI
    owner: Cluster1
    ip: 10.xx.xx.xx    #--> update the cluster/node IP
    port: 31456        #--> ocnaddgui port
    apiPrefix: /<clustername>/<namespace>/ocnadd   # the clustername and
namespace where the OCNADD GUI is deployed
  - id: Cluster1-dd-instance1
    type: DD-API
    owner: Cluster1
    ip: 10.xx.xx.xx   #--> update the cluster/node IP
    port: 32406       #--> ocnaddbackendrouter port
    apiPrefix: /<clustername>/<namespace>/ocnadd   # the clustername and
namespace where the OCNADD GUI is deployed
```

Example:

If OCNADD GUI is deployed in the **occne-ocdd** cluster and the **ocnadd-deploy** namespace, then the prefix in CNCC `occncc_custom_values.yaml` will be as follows:

```
DD-UI apiPrefix:
/occne-ocdd/ocnadd-deploy/ocnadd
DD-API apiPrefix:
/occne-ocdd/ocnadd-deploy/ocnaddapi
```

Access OCNADD GUI

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

3
Customizing OCNADD

This chapter describes how to customize the Oracle Communications Network Analytics Data Director (OCNADD) deployment, supported deployment models, and provides a list of configuration parameters in the Helm file that are used for customization. The OCNADD deployment is customized by overriding the default values of various configurable parameters.

3.1 Global Parameters

Table 3-1 Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ocnaddalarm.enabled	BOOLEAN	true/false	true	M	To enabled alarm charts
ocnaddconfiguration.enabled	BOOLEAN	true/false	true	M	To enabled configuration charts
ocnaddhealthmonitoring .enabled	BOOLEAN	true/false	true	M	To enabled healthmonitoring charts
ocnaddfilter.enabled	BOOLEAN	true/false	true	M	To enabled filter charts
ocnaddaggregation.enabled	BOOLEAN	true/false	true	M	To enabled aggregation charts
ocnaddbackuprestore.enabled	BOOLEAN	true/false	true	M	To enabled backuprestore charts
ocnaddkafka.enabled	BOOLEAN	true/false	true	M	To enabled kafka charts
ocnaddadminsvc.enabled	BOOLEAN	true/false	true	M	To enabled adminsvc charts
ocnaddbackendrouter.enabled	BOOLEAN	true/false	true	M	To enabled backendrouter charts
ocnaddgui.enabled	BOOLEAN	true/false	true	M	To enabled gui charts
env.ocwebclient.OCWEBCLIENT_TIMEOUT	INTEGER	-	30	O	Webclient timeout in seconds
env.ocwebclient.OCWEBCLIENT_KEEPALIVE_IDLE	INTEGER	-	90	O	Webclient keepalive idle time in seconds

Table 3-1 (Cont.) Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
scaleDownOnePodAtATime	BOOLEAN	true/false	false	M	Scale Down Pods One at a Time
stabilizationWindowSeconds	INTEGER	-	60	M	Stabilization period in seconds post which scale down starts
scaleDownPeriodSeconds	INTEGER	-	30	M	Period of each scale down opeartion in seconds
scaleDownValue	INTEGER	-	1	M	Number of pods which shall go down in every scaleDownPeriodSeconds
initContainers.name	STRING	-	ocnaddinitcontainer	M	Name of initContainer for SSL support
initContainers.image	STRING	-	jdk17-openssl:1.0.6	M	InitContainer Image
initContainers.REPO_PATH	STRING	-	utils.repo	M	Repo path where init image is stored
initContainers.volumeMounts.ts_ks_volumeName	STRING	-	truststore-keystore-volume	M	Volume name for truststore
initContainers.volumeMounts.ts_ks_volumePath	STRING	-	/var/securityfiles/keystore	M	Path where keystore files are stored
initContainers.volumeMounts.certificateName	STRING	-	client-server-certificate	M	Volume name for server certificates
initContainers.volumeMounts.certificatePath	STRING	-	/var/securityfiles/certs	M	Path where cert files are stored
initContainers.env.cert_file_params.SERVER_CERT_FILE	STRING	-	servercert.pem	M	Server cert filename
initContainers.env.cert_file_params.CLIENT_CERT_FILE	STRING	-	clientcert.pem	M	Client cert filename

Table 3-1 (Cont.) Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
initContainers.env.cert_file_params.SERVER_KEY_FILE	STRING	-	serverprivatekey.pem	M	Server Private Key filename
initContainers.env.cert_file_params.CLIENT_KEY_FILE	STRING	-	clientprivatekey.pem	M	Client Private Key filename
initContainers.env.ks_file_params.SERVER_KEY_STORE	STRING	-	serverKeyStore.p12	M	Server Keystore file
initContainers.env.ks_file_params.CLIENT_KEY_STORE	STRING	-	clientKeyStore.p12	M	Client Keystore file
initContainers.env.ks_file_params.TRUST_STORE	STRING	-	trustStore.p12	M	Truststore file
initContainers.cacert.key	STRING	-	CA_CERT_FILE	M	Cacert key file
initContainers.cacert.value	STRING	-	cacert.pem	M	Cacert file
ssl.intraTlsEnabled	BOOLEAN	true/false	false	M	Enable internal service TLS
ssl.mTLS	BOOLEAN	true/false	false	M	Enable MTLS support for internal OCNADD services
image.ocnaddadminsvc.name	STRING		ocnaddadminsvc:1.0.72	M	Admin Service Image
image.ocnaddconsumeradapter.name	STRING		ocnaddconsumeradapter:1.0.76	M	Consumer Adapter Image
image.ocnaddaggregation.name	STRING		ocnaddaggregation:1.0.51	M	Aggregation Service Image
image.ocnaddalarm.name	STRING		ocnaddalarm:1.0.45	M	Alarm Service Image
image.ocnaddconfiguration.name	ocnaddfilter:1.0.29-dev		ocnaddconfiguration:1.0.61	M	Configuration Service Image
image.ocnaddegressgateway.name	STRING		ocnaddegressgateway:1.0.41-dev	M	Egress gateway image
image.ocnaddfilter.name	STRING		ocnaddfilter:1.0.29-dev	M	Filter Service Image

Table 3-1 (Cont.) Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
image.ocnaddhealthmonitoring.name	STRING		ocnaddhealthmonitoring:1.0.52	M	Health Monitoring Image
image.ocnaddbackendrouter.name	STRING		ocnaddbackendrouter:1.0.48	M	Backend Router Image
image.ocnaddbackuprestore.name	STRING		ocnaddbackuprestore:1.0.7	M	Backup Restore Image
image.ocnadd_kafka_healthclient.name	STRING		ocnadd-kafka-healthclient-3.2.3:1.0.11	M	Kafka Image
image.ocnadd_jmx_exporter.name	STRING		jmx-exporter-jdk-17.0.6:0.17.2	M	JMX exporter image
image.ocnaddaggregation.name	STRING		ocnaddgui:23.6.9-feature-mergeTest	M	OCNADD GUI Image
cluster.clusterName	STRING		occne-ocdd	M	Default cluster name of setup
cluster.secret.name	STRING		db-secret	M	Database Secret name where DB credentials are stored
cluster.mysqlNameSpace.name	STRING		occne-cndbtierone	M	DBTier namespace
cluster.mysqlPod	STRING		ndbmysqld-0	M	DBTier Pod Name
cluster.database.db_ip	STRING		mysql-connectivity-service.occne-cndbtierone	M	Hostname or IP of DBTier
cluster.database.db_port	INTEGER		3306	M	DB Port
cluster.database.configuration_db	STRING		configuration_schema	M	Configuration Service Schema Name
cluster.database.alarm_db	STRING		alarm_schema	M	Alarm Service Schema Name
cluster.database.health_db	STRING		healthdb_schema	M	Health Service Schema Name
cluster.storageClass	STRING		standard	M	Storage Class Name
cluster.nameSpace.name	STRING	-	ocnadd-deploy	M	OCNADD Namespace
cluster.serviceAccount.create	BOOLEAN	true/false	true	M	To create a ServiceAccount (true/false)

Table 3-1 (Cont.) Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
cluster.clusterRole.create	BOOLEAN	true/false	true	M	To create clusterRole (true/false)
cluster.clusterRoleBinding.create	BOOLEAN	true/false	true	M	To create clusterRoleBinding (true/false)
cluster.terminationGracePeriodSeconds	INTEGER	-	5	O	Pod grace termination
cluster.imagePullSecret.enable	BOOLEAN	true/false	false	M	Image Pull secret creation
cluster.imagePullSecret.name	STRING	-	regcred	O	Set to regcred if cluster.imagePullSecret.enable is true
cluster.kafka.ocnadd_kafka_bootstrap_servers	STRING	-	kafka-broker:9092	M	Bootstrap server for PLAINTEXT
cluster.kafka.ocnadd_kafka_bootstrap_servers_ssl	STRING	-	kafka-broker:9093	M	Bootstrap server for SSL
cluster.kafka.ocnadd_kafka_bootstrap_servers_sasl	STRING	-	kafka-broker:9094	M	Bootstrap server for SASL
cluster.prometheusScrapePort	INTEGER	-	9000	O	Port to scape metrics required if metrics enabled
cluster.prometheusPortName	STRING	-	cnc-metrics	O	Role required to define in alert rules yaml
cluster.max_latency	FLOAT	-	0.05	M	Max latency range of 50ms
cluster.memory_threshold	INTEGER	[0-100]	90	M	Max Threshold limit for memory
cluster.cpu_threshold	INTEGER	[0-100]	70	M	CPU max threshold limit
cluster.mps	INTEGER	-	55000	M	Default MPS rate
cluster.egwGroupLatencyMessageCountMax	INTEGER	-	50	M	Max latency count for Egress Gateway

Table 3-1 (Cont.) Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
cluster.prometheus_url	STRING	-	http://localhost:9000/cluster-name/prometheus/api/v1/query_range	M	Prometheus URL to scrape metrics
network.policy.enable	BOOLEAN	true/false	false	M	Network Policy enable for intercommunication of OCNADD services
network.ingress.denyall	BOOLEAN	true/false	false	C	Deny all ingress traffic
network.ingress.kafka	BOOLEAN	true/false	true	C	Allow ingress traffic for kafka
network.ingress.aggregation	BOOLEAN	true/false	true	C	Allow ingress traffic for aggregation service
network.ingress.filter	BOOLEAN	true/false	true	C	Allow ingress traffic for filter service
network.ingress.adapter	BOOLEAN	true/false	true	C	Allow ingress traffic for adapter service
network.ingress.egw	BOOLEAN	true/false	true	C	Allow ingress traffic for egress service
network.ingress.config	BOOLEAN	true/false	true	C	Allow ingress traffic for configuration service
network.ingress.alarm	BOOLEAN	true/false	true	C	Allow ingress traffic for alarm service
network.ingress.health	BOOLEAN	true/false	true	C	Allow ingress traffic for health monitoring service
network.ingress.admin	BOOLEAN	true/false	true	C	Allow ingress traffic for admin service
network.ingress.namespaces	STRING	-	- occne-infra - occncc	C	Network communication between allowed namespaces
network.ingress.external.enable	BOOLEAN	true/false	false	C	Allow kafka LoadBalancer IP to be created

Table 3-1 (Cont.) Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
network.ingress.external.cidrs	STRING	-	- 10.0.0.0/8	C	Cidr for network communication
network.egress.denyall	BOOLEAN	true/false	false	C	Deny egress traffic

3.2 Helm Hook Parameters

Table 3-2 Helm Hook Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ocnaddhelmhook.config.name	STRING	-	helmhook-configmap	M	Name of ConfigMap
ocnaddhelmhook.config.upgrade_name	STRING	-	helmhook-upgrade-configmap	M	Name of Upgrade ConfigMap
ocnaddhelmhook.config.rollback_name	STRING	-	helmhook-rollback-configmap	M	Name of Rollback ConfigMap
ocnaddhelmhook.name	STRING	-	ocnaddhelmhook	M	Helm Hook Name
ocnaddhelmhook.container.name	STRING	-	ocnaddhelmhook	M	Container Name of Helm Hook Job
ocnaddhelmhook.container.image	STRING	-	pre-install-image:1.0.5	M	Image used for pre-install hooks
ocnaddhelmhook.container.imagePullPolicy	STRING	IfNotPresent/ Always/Never	IfNotPresent	M	Image Pull Policy
ocnaddpostinstallhelmhook.name	STRING	-	ocnaddpostinstallhelmhook	M	Post Install Hook Name
ocnaddpostupgradehelmhook.name	STRING	-	ocnaddpostupgradehelmhook	M	Post Upgrade Hook Name
ocnaddpostrollbackhelmhook.name	STRING	-	ocnaddpostrollbackhelmhook	M	Post Rollback hook name
ocnaddpreupgradehelmhook.name	STRING	-	ocnaddpreupgradehelmhook	M	Pre Upgrade Hook Name
ocnaddprerollbackhelmhook.name	STRING	-	ocnaddprerollbackhelmhook	M	Pre Rollback Hook Name

3.3 Aggregation Service Parameters

Table 3-3 Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ocnaddnrfaggregation.name	STRING	-	ocnaddnrfaggregation	M	Name of the application
ocnaddscpaggregation.name	STRING	-	ocnaddscpaggregation	M	Name of the application
ocnaddseppaggregation.name	STRING	-	ocnaddnrfaggregation	M	Name of the application
ocnaddnrfaggregation.resources.limit.cpu	INTEGER	-	3	M	Number of max CPU for NRF Aggregation
ocnaddnrfaggregation.resources.limit.memory	STRING	-	2Gi	M	Max Memory limit for NRF Aggregation
ocnaddnrfaggregation.resources.limit.ephemeralstorage	STRING	-	500Mi	M	Ephemeral Storage for NRF Aggregation
ocnaddscpaggregation.resources.limit.cpu	INTEGER	-	3	M	Number of max CPU for SCP Aggregation
ocnaddscpaggregation.resources.limit.memory	STRING	-	2Gi	M	Max Memory limit for SCP Aggregation
ocnaddscpaggregation.resources.limit.ephemeralstorage	STRING	-	500Mi	M	Ephemeral Storage for SCP Aggregation
ocnaddseppaggregation.resources.limit.cpu	INTEGER	-	3	M	Number of max CPU for SEPP Aggregation
ocnaddseppaggregation.resources.limit.memory	STRING	-	2Gi	M	Max Memory limit for SEPP Aggregation
ocnaddseppaggregation.resources.limit.ephemeralstorage	STRING	-	500Mi	M	Ephemeral Storage for SEPP Aggregation
OCNADD_AGGREGATION_LOG_ROOT	STRING	-	INFO	O	Default Log level set for the application.
OCNADD_AGGREGATION_LOG_NETTY	STRING	-	INFO	O	Default Netty Log level set for the application.

Table 3-3 (Cont.) Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_AGGREGATION_SERVICE_TOPIC_RETRIES_THRESHOLD	INTEGER	-	120000	O	Retry Threshold for TOPIC reachability
OCNADD_METRICS_CPU_THRESHOLD	FLOAT	[0, 1]	0.45	O	Parameter to compute the CPU related metrics of the application.
OCNADD_METRICS_MEM_THRESHOLD	FLOAT	[0, 1]	0.95	O	Parameter to compute the Memory related metrics of the application.
ALARM_SERVICE_URL	STRING	-	http://ocnaddalarm:9099	M	Alarm Service API Root to raise alarm
KAFKA_PRODUCER_SECURITY_PROTOCOL	STRING	-	PLAINTEXT	M	Kafka Producer Secutiry Protocol.
KAFKA_PRODUCER_SSL_PROTOCOL	STRING	-	TLSv1.3	M	SSL Protocol
KAFKA_PRODUCER_SASL_MECHANISM	STRING	-	PLAIN	M	Kafka Producer SSAL Mechanism.
KAFKA_PRODUCER_SSL_CLIENT_AUTH	BOOLEAN	true, false	false	M	Kafka SSL client authentication.
KAFKA_BOOTSTRAP_SERVER	STRING	-	kafka-broker:9092	M	Kafka Boot strap server address.
KAFKA_MAX_AGE_CONFIG	INTEGER	-	7500	M	The period of time in milliseconds after which we force a refresh of metadata.
OCNADD_TRUST_KEYSTORE	BOOLEAN	true, false	true	M	Enable to secure connection via OCWeb Client.
KAFKA_FETCH_MIN_BYTES	STRING	-	1	O	The minimum amount of data per-partition the server will return

Table 3-3 (Cont.) Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
KAFKA_FETCH_MAX_BYTES	INTEGER	-	576720	O	The maximum amount of data per-partition the server will return
KAFKA_MAX_PARTITIONS_FETCH_BYTES	INTEGER	-	104858	O	The maximum amount of data per-partition the server will return.
FETCH_MAX_WAIT_MS	INTEGER	-	100	O	The maximum amount of time the server will block before answering the fetch request if there isn't sufficient data to immediately satisfy the requirement given by fetch.min.bytes
SESSION_TIME_OUT	INTEGER	-	15000	O	The timeout used to detect client failures when using Kafka's group management facility.
HEARTBEAT_INTERVAL_MS	INTEGER	-	5000	O	The expected time between heartbeats to the consumer coordinator when using Kafka's group management facilities
MAX_POLL_INTERVAL_MS	INTEGER	-	30000	O	The maximum delay between invocations of poll() when using consumer group management
MAX_POLL_RECORDS	INTEGER	-	300	O	The maximum number of records returned in a single call to poll()

Table 3-3 (Cont.) Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
CONSUMER_POLL_MS	INTEGER	-	50	O	Polling time in ms for consumer
BATCH_SIZE	INTEGER	-	65536	O	The maximum amount of data to be collected before sending the batch.
LINGER_MS	INTEGER	-	1	O	The time to wait before sending messages out to Kafka
REQUEST_TIMEOUT_MS	INTEGER	-	1000	O	The configuration controls the maximum amount of time the client will wait for the response of a request
KAFKA_SOCKET_BYTES_BUFFER	INTEGER	-	104857	O	Kafka Socket Buffer setting for consumer

3.4 Configuration Service Parameters

Table 3-4 Configuration Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ocnaddconfiguration.name	STRING	-	ocnaddconfiguration	M	Name of configuration service
logging.type	STRING	STDOUT/ LOGJSON	STDOUT	O	Logging type Standard Output or JSON format
logging.level.reactor.netty	STRING	-	INFO	O	Netty logging level
OCNADD_TRUST_KEYSTORE	BOOLEAN	-	false	O	Truststore enable for Configuration Service
logging.level.com.oracle.cgbu.cne.ocnadd	STRING	-	INFO	O	Logging level for OCNADD services

Table 3-4 (Cont.) Configuration Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
logging.level.com.oracle.cgbu.cne.ocdd	STRING	-	INFO	O	Logging level for Common OCNADD services

3.5 Health Monitoring and Alarm Service Parameters

Table 3-5 Health Monitoring Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ocnaddhealthmonitoring.name	STRING	ocnaddhealthmonitoring	M	Healthmonitoring monitoring service name
HEALTH_MONITORING_TIMER	INTEGER	5000	O	Timer to check Health of integrated services
HEALTH_METRICS_SCHEDULED	BOOLEAN	true	O	Scheduler for metrics
HEALTH_METRICS_TIMER	INTEGER	120000	O	Timer for health metrics
HEALTH_PURGE_TIME_HR	INTEGER	24	O	Health profile purging timer in hour
HEALTH_MONITORING_CPUTHRESHOLD	INTEGER	75	M	CPU threshold to raise alarm
HEALTH_MONITORING_MEMORYTHRESHOLD	INTEGER	95	M	Memory threshold to raise alarm
Logging Properties				
HEALTH_LOG_HTTPCLIENT	STRING	INFO	O	Set Default Log level for Http Client
HEALTH_LOG_SPRING_WEB	STRING	INFO	O	Set Default Log level for Spring Web
logging.level.com.oracle.cgbu.cne.ocnadd	STRING	INFO	O	Logging level for Health Monitoring OCNADD Service
logging.level.com.oracle.cgbu.cne.ocdd	STRING	INFO	O	Logging level for Common OCNADD Service
HEALTH_APPLICATION_LOG_LEVEL	STRING	INFO	O	Set application logger level

Table 3-6 Alarm Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ocnaddalarm.name	STRING	-	ocnaddalarm	M	Application name for Alarm Service
logging.type	STRING	-	STDOUT	O	Logging Type Standard Outpuut or JSON Format
OCNADD_TRUST_KEYSTORE	BOOLEAN	true/false	true	O	Trust Keystore Enable
logging.level.com.oracle.cgbu.cne.ocnadd	STRING	-	INFO	O	Logging level for Alarm OCNADD Service
logging.level.com.oracle.cgbu.cne.ocdd	STRING	-	INFO	O	Logging level for Common OCNADD Service

3.6 Admin Service Parameters

Table 3-7 Admin Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ocnadd.admin.name	STRING	-	ocnaddadminservice	M	Application name admin service
Environment Variables					
OCNADD_ADAPTER_LOG_LEVEL_NETTY	STRING	-	INFO	O	Netty log level
OCNADD_ADAPTER_LOG_LEVEL	STRING	-	INFO	O	Adapter service log level
logging.level.com.oracle.cgbu.cne.ocnadd	STRING	-	INFO	O	Logging level for OCNADD services
logging.level.com.oracle.cgbu.cne.ocdd	STRING	-	INFO	O	Logging level for Common OCNADD services
OCNADD_ADAPTER_MIN_REPLICAS	INTEGER	-	2	M	Minimum Replicas for Adapter

Table 3-7 (Cont.) Admin Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_ADAPTER_MAX_REPLICAS	INTEGER	-	8	M	Max Replicas for Adapter
MAX_TCP_CONNECTION_PER_DEST	INTEGER	-	1	M	Max allowed TCP connection per destination
ADAPTER_KAFKA_FETCH_MAX_BYTES	INTEGER	-	576720	O	The maximum amount of data the server should return for a fetch request
ADAPTER_KAFKA_MAX_PARTITION_FETCH_BYTES	INTEGER	-	104857	O	The maximum amount of data per-partition the server will return
ADAPTER_KAFKA_FETCH_MAX_WAIT_MS	INTEGER	-	40	O	The maximum amount of time the server will block before answering the fetch request
ADAPTER_KAFKA_SESSION_TIME_OUT	INTEGER	-	15000	O	The timeout used to detect client failures when using Kafka's group management facility
ADAPTER_KAFKA_HEARTBEAT_INTERVAL_MS	INTEGER	-	5000	O	The expected time between heartbeats to the consumer coordinator when using Kafka's group management facilities
ADAPTER_KAFKA_MAX_POLL_INTERVAL_MS	INTEGER	-	30000	O	The maximum delay between invocations of poll() when using consumer group management
ADAPTER_KAFKA_MAX_POLL_RECORDS	INTEGER	-	1500	O	The maximum number of records returned in a single call to poll()

Table 3-7 (Cont.) Admin Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ADAPTER_KAFKA_COMMIT_INT_CONFIG	INTEGER	-	30	O	The frequency in milliseconds that the consumer offsets are committed to Kafka
ADAPTER_KAFKA_NUM_THREADS_CONFIG	INTEGER	-	9	O	The number of threads to execute stream processing
ADAPTER_KAFKA_CONSUMER_POLL_MS	INTEGER	-	30	O	The amount of time in milliseconds to block waiting for input
ADAPTER_KAFKA_RECEIVE_BUFFER_BYTES	INTEGER	-	104857	M	The size of the TCP receive buffer (SO_RCVBUF) to use when reading data.
ADAPTER_WEBCLIENT_TYPE	STRING	-	netty	M	Webclient Type for Adapter Service
OCNADD_ADAPTER_USE_TS	BOOLEAN	true/false	true	M	Parameter to enable trustore use for Adapter
OCNADD_ADAPTER_SERVER_SSL	BOOLEAN	true/false	false	M	Enable Server SSL for Adapter
EGRESS_MAX_CONNECTION_POOL_COUNT	INTEGER	-	100	O	Max number of connections per connection pool
EGRESS_SSL_HANDSHAKE_TIMEOUT	INTEGER	-	40	O	SSL handshake timeout
EGRESS_SSL_CLOSE_NOTIFY_FLUSH_TIMEOUT	INTEGER	-	20	O	HttpClient Notify Flush timeout
EGRESS_SSL_CLOSE_NOTIFY_READ_TIMEOUT	INTEGER	-	20	O	HttpClient Notify Read timeout
EGRESS_CLIENT_MAX_INMEMORY	INTEGER	-	12	O	Webclient Max In Memory in Mb

Table 3-7 (Cont.) Admin Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
EGRESS_JETTY_MAX_CONNECTION_PER_DESTINATION	INTEGER	-	12	O	Jetty Max Connection per Destination
EGRESS_JETTY_MAX_REQUEST_PER_DESTINATION	INTEGER	-	1000	O	Jetty Max Request per Destination
EGRESS_JETTY_CLIENT_IDLE_TIMEOUT	INTEGER	-	300000	O	Jetty client IDLE timeout
EGRESS_JETTY_CLIENT_CONNECT_BLOCKING	BOOLEAN	true/false	false	O	Jetty Client Connection Block Enable
EGRESS_JETTY_CLIENT_CONNECT_TIMEOUT	INTEGER	-	120000	O	Jetty Client Connection Timeout
EGRESS_JETTY_CLIENT_STREAM_IDLE_TIMEOUT	INTEGER	-	300000	O	Jetty Client Stream IDLE timeout
EGRESS_JETTY_CLIENT_MAX_CONCURR_PUSHED_STREAM	INTEGER	-	12	O	Jetty HTTP2 client max concurrent pushed stream
ADAPTER_TCP_CLIENT_MSG_RETRY_MAX_ATTEMPTS	INTEGER	-	10	O	Max TCP message retry
ADAPTER_TCP_CLIENT_MSG_RETRY_DELAY	INTEGER	-	50	O	TCP Message retry delay
ENABLE_ADAPTER_COUNTER_METRICS	BOOLEAN	true/false	true	O	Enable Adapter Counter Metric
ENABLE_ADAPTER_LATENCY_METRICS	BOOLEAN	true/false	true	O	Enable Adapter Latency Metric

3.7 Kafka Configuration Parameters

Table 3-8 Kafka Configuration Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
kafkaBroker.kafkaProperties.logdirs	String	-	/kafka/logdir/kafka-logs	M	The path to store the kafka logs
kafkaBroker.name	String		kafka-broker		Name of the kafka broker
kafkaBroker.replicas	Int		1		The number of replicas that you want to be available for the pod
kafkaBroker.pvcClaimSize	String		10Gi	M	Size of Block Volume to attach to kafka
target.averageCpuUtilPercentage	Int		50		The target average CPU utilization percentage
target.memoryUtilPercentage	Int		80		The target average memory utilization percentage
kafkaBroker.resource.limits.cpu	Int		5		The maximum limit for the number of CPUs used for the container
kafkaBroker.resource.limits.memory	String		24Gi		The maximum limit for the size of the memory used for the container
kafkaBroker.kafkaProperties.logRetentionMinutes	Int		5	M	Log Retention Time of Topic Data in Minutes
kafkaBroker.kafkaProperties.kafkaSslProtocol	String		TLSv1.2,TLSv1.3	M	TLS version supported
kafkaBroker.kafkaProperties.socketSendBufferBytes	Int		10485760	M	TCP socket buffer sizes for the producer
kafkaBroker.kafkaProperties.socketReceiveBufferBytes	Int		10485760	M	TCP socket buffer sizes for the consumer

Table 3-8 (Cont.) Kafka Configuration Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
kafkaBroker.kafkaProperties.socketRequestMaxBytes	Int		104857600	M	The maximum number of bytes in a socket request
kafkaBroker.kafkaProperties.queuedMaxRequests	Int		512	M	Number of concurrent connections
kafkaBroker.kafkaProperties.numIoThreads	Int		640	M	Number of threads that pick up requests from the request queue to process them
kafkaBroker.kafkaProperties.numNetworkThreads	Int		640	M	Network threads handle requests to the Kafka cluster, such as produce and fetch requests from client applications
kafkaBroker.kafkaProperties.numReplicaFetchers	Int		512	M	Number of fetcher threads used to replicate records from each source broker
kafkaBroker.kafkaProperties.backgroundThreads	Int		128	M	The number of threads to use for various background processing tasks
kafkaBroker.kafkaProperties.replicaFetchMinBytes	Int		619200	M	Minimum bytes expected for each fetch response
kafkaBroker.kafkaProperties.replicaFetchMaxBytes	Int		37152000	M	The maximum number of bytes we will return for a fetch request
kafkaBroker.kafkaProperties.replicaFetchWaitMaxMs	Int		50	M	The maximum wait time for each fetcher request issued by follower replicas

Table 3-8 (Cont.) Kafka Configuration Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
kafkaBroker.kafkaProperties.replicaSocketReceiveBufferBytes	Int		10485760	M	The socket receive buffer for network requests
kafkaBroker.externalAccess.enabled	Boolean		false	M	Flag to enable External access for Kafka
kafkaBroker.externalAccess.autoDiscovery	Boolean		false	M	Flag to enable auto-discovery of LoadBalancer IPs
kafkaBroker.externalAccess.type	String		LoadBalancer	M	Service Type of Kafka Broker
kafkaBroker.externalAccess.setstaticLoadBalancerIps	Boolean		false	M	Setting Static LoadBalancer IPs
kafkaBroker.externalAccess.LoadBalancerIPList	List		[ ]	C	List if LoadBalancer Static IP available for use

3.8 Backend Router Parameters

Table 3-9 Backend Router Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ocnaddbackendrouter.name	String	-	ocnaddbackendrouter	M	Application name for ocnaddbackendrouter
OCNADD_BKNROUTER_ROOT_LOG_LEVEL	String		INFO	O	Root log level configuration (INFO,DEBUG, ERROR)
OCNADD_BKNROUTER_COM_LOG_LEVEL	String		INFO	O	Log level configuration (INFO,DEBUG, ERROR)

Table 3-9 (Cont.) Backend Router Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_BKNROUTER_LOG_LEVEL	String		INFO	O	Log level configuration (INFO,DEBUG, ERROR)
OCNADD_BKNROUTER_CLIENT_SSL_USE_TS	Boolean		true	O	Client SSL Truststore enable
OCNADD_BKNROUTER_CLIENT_WIRETAP	Boolean		false	O	Client Wiretap
OCNADD_BKNROUTER_SERVER_WIRETAP	Boolean		false	O	Server Wiretap
WEBCLIENT_MAX_BUFFER_SIZE	String		17777216	O	Webclient Max buffer size for Backend router

4
Upgrading OCNADD

This section provides information on how to upgrade an existing OCNADD deployment.

> **Note**
>
> The OCNADD can be upgraded from a source release to a target release using CLI procedures as outlined in the following sections. These steps can also be followed for any hotfix upgrade.

4.1 Preupgrade Tasks

Before starting the procedure to upgrade OCNADD, perform the following tasks:

> **Note**
>
> - While performing an upgrade, you must align the `ocnadd-custom-values.yaml` file of the target release as per the `ocnadd/values.yaml` file of the source release or the older release.
> - Do not enable any new feature during the upgrade.
> - The parent or sub-charts `values.yaml` parameter must not be changed while performing the upgrade, unless it is explicitly specified in this document.
>
> For information about enabling any new feature through Helm parameters, see *Oracle Communications Network Analytics Data Director User Guide*.

1. Fetch the images and charts of the target release as described in Installing OCNADD.
2. Keep a backup of the current `ocnadd/values.yaml` file of the source release as a backup while upgrading to target release.
3. Update the following helm chart files of the target release with the parameter values of the source release files:
   - `ocnadd-custom-values.yaml`
   - `ocnadd/ocdd-db-resource.sql`
   - `ocnadd/templates/ocnadd-secret-hook.yaml`
   - Update the `pvcClaimSize` parameter of all the Kafka brokers `ocnadd-custom-values.yaml`.

4.2 OCNADD Upgrade

This section includes information about upgrading an existing OCNADD deployment.

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

Important

- (Optional) A timeout interval of 15 minutes can be set while performing an upgrade as only one POD of the Data Director services is upgraded at a time.
- Ensure that no OCNADD pod is in the failed state
- Ensure that the defined in the Preupgrade Tasks are complete
- There can be a downtime of Kafka brokers for about a minute while performing an upgrade that affects all of the brokers. You can avoid this downtime by upgrading the brokers one at a time, if applicable. Kafka upgrade along with PVC storage changes are not supported.
- The Consumer Adapter pods/services are created when Data feed is created from OCNADD GUI. By default, the upgrade of these pods is set to **false**. To upgrade them, follow the procedure described in "Upgrade Consumer Adapter" section in the *Oracle Communications Network Analytics Data Director User Guide* .

Execute the following command to upgrade an existing OCNADD deployment:

1. Upgrade the OCNADD microservices:
   - When using the local Helm chart:

     ```
     helm upgrade <release_name> <helm_chart> --namespace <namespace-name>
     --timeout=15m
     ```

     where,

     <release_name> is the release name used by the Helm command

     <helm_chart> is the location of the Helm chart extracted from the target ocnadd-<releaseNumber>.tgz file

     <namespace-name> is the OCNADD namespace in which the release is deployed
   - When using the chart from Helm repo:

     ```
     helm upgrade <release_name> <helm_repo/helm_chart> --version
     <chart_version> --namespace <namespace-name> --timeout=15m
     ```

     where,

     <helm_repo> is the OCNADD Helm repo.

     <chart_version> is the version of the Helm chart extracted from the ocnadd-<releaseNumber>.tgz file

     <namespace-name> is the OCNADD namespace in which the release is deployed
2. Check the status of the upgrade:

   ```
   helm history <release_name> --namespace <namespace-name>
   ```

3. Verify if the upgrade is successful using the following steps:
   a. All the pods that have been respawned after upgrade, have the latest age (in secs)
   b. The Adapter pods also gets respawned for any upgrade. The status can also be verified from GUI for respective Data Feeds.
      In case of any failure, follow the steps mentioned in the *Oracle Communications Network Analytics Data Director Troubleshooting Guide*.

4.2.1 Hotfix Upgrade

For a HotFix patch upgrade, follow the steps mentioned in the OCNADD Upgrade section.

5
Rollback OCNADD

This chapter describes the OCNADD rollback procedure from a target release to a previous supported version.

Rollback Steps

To roll back to a previous version, follow the steps as mentioned:

> **Note**
>
> - (Optional) A timeout interval of 15 minutes can be set while performing an upgrade as only one POD of the Data Director services is upgraded at a time.
> - Ensure the status for the target version is not in failed or error state.
> - Before Rollback from target release to source release, the configurations created with the new data feed and synthetic packets should be deleted manually using the GUI.

1. Run the following command to check the revision you must roll back to:

   ```
   $ helm history <release_name> -n <release_namespace>
   ```

   Where,

   <release_name> is the release name used by the Helm command.

   <release_namespace> is the OCNADD release name, for example, ocnadd.

   Example:
2. Run the command to rollback to the required revision:

   ```
   helm rollback <release_name> <REVISION_number> --namespace
   <release_namespace>
       --timeout=15m
   ```

   Where, <REVISION_number> is the release version to which Data Director needs to be rolled back is obtained in the previous step.

   Example:

   helm rollback ocnadd 1 --namespace ocnadd --timeout=15m
3. Verify if the rollback is successful
   a. All the pods that has been respawned after upgrade, have the latest age(in secs)
   b. The Adapter pods also gets respawned for any upgrade. The status can also be verified from GUI for respective Data Feeds.
      If the rollback is not successful, perform the troubleshooting steps mentioned in *Oracle Communications Network Analytics Data Director Troubleshooting Guide.*.

6
Uninstalling OCNADD

This chapter provides information on how to uninstall Oracle Communications Network Analytics Data Director (OCNADD).

When you uninstall a helm chart from the OCNADD deployment, it removes only the Kubernetes objects created during the installation.

> **Note**
>
> `kubectl` commands might vary based on the platform deployment. Replace `kubectl` with Kubernetes environment-specific command line tool to configure kubernetes resources through kube-api server. The instructions provided in this document are as per the Oracle Communications Cloud Native Environment (OCCNE) version of kube-api server.

> **Caution**
>
> - While deleting any OCNADD resources make sure to provide the corresponding namespace used in the deployment.
> - Based on requirement, make sure to retain the OCNADD backup before the uninstallation procedure. For more information, see Performing OCNADD Backup Procedures.
> - Ensure any configured datafeeds are deleted using the OCNADD GUI prior to performing the OCNADD uninstallation steps. For deletion of the datafeeds, refer to *Oracle Communications Network Analytics Data Director User Guide*.

To uninstall OCNADD, run the following command:

Helm 2:

```
helm delete --purge <release_name> --namespace <namespace>
```

Helm 3:

```
helm3 uninstall <release_name> --namespace <namespace>
```

where, *release_name* is a name provided to identify the helm deployment.

release-namespace is the name provided to identify the namespace of OCNADD deployment.

Clean Up Database

To clean up database, perform the following steps:

1. Log in to the MySQL client on SQL Node with the OCNADD user and password:

   ```
   mysql -h <IP_adress of SQL Node> -u <ocnadduser> -p (Give password in
   prompt)
   ```

2. To clean up the configuration, alarm, and health database, run the following command:

   ```
   mysql> drop database <dbname>;
   ```

3. To remove MySQL users while uninstalling OCNADD, run the following commands:

   ```
   SELECT user FROM mysql.user;
   DROP USER 'ocnaddappuser@'%';
   ```

Clean up Kafka Configuration

To clean up Kafka configuration, perform the following steps:

1. To list the secrets in the namespace, run the following command:

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

2. To delete all the secrets related to Kafka, run the following command:

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

3. To delete configmap used for Kafka, run the following command:

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

4. To delete PVCs used for Kafka,
   a. run the following command, and list the PVCs used in the namespace:

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

   b. run the following command, and delete the PVCs used by the brokers and zookeepers:

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

Delete Cluster Role and Cluster Rolebindings

To delete cluster role and cluster rolebindings, perform the following steps:

1. To list the cluster roles in the namespace, run the following command:

   ```
   kubectl get clusterrole | grep <namespace>
   ```

2. To delete the cluster roles, run the following command:

   ```
   kubectl delete clusterrole <name of clusterrole>
   ```

3. To list the cluster rolebindings, run the following command:

   ```
   kubectl get clusterrolebinding | grep <namespace>
   ```

4. To delete the cluster rolebindings, run the following command:

   ```
   kubectl delete clusterrolebinding <name of clusterrolebinding>
   ```

6.1 Verifying Uninstallation

To verify the Oracle Communications Network Analytics Data Director (OCNADD) uninstallation, run the following command:

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

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

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

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

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

     ⚠ Caution

     The command deletes all the Kubernetes objects of the specified namespace. In case, you have created the RBAC resources and service accounts before the helm installation in the same namespace, and these resources are required, then do not delete them.

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

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

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

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

   Example:

   ```
   kubectl delete namespace ocnadd
   ```

   ⚠ Caution

   The command removes all the resources or objects created in the namespace. Therefore, ensure that you run the command only when you want to delete the namespace completely.

7
Fault Recovery

This chapter provides information about fault recovery for OCNADD deployment.

7.1 Overview

This section describes procedures to perform the backup and restore for the Oracle Communications Network Analytics Data Director (OCNADD) deployment. The backup and restore procedures will be used in the fault recovery of the OCNADD. The OCNADD operators can take only the OCNADD instance specific database and required OCNADD Kafka metadata backup and restore them either on the same or a different Kubernetes cluster.

The backup and restore procedures are helpful in the following scenarios:

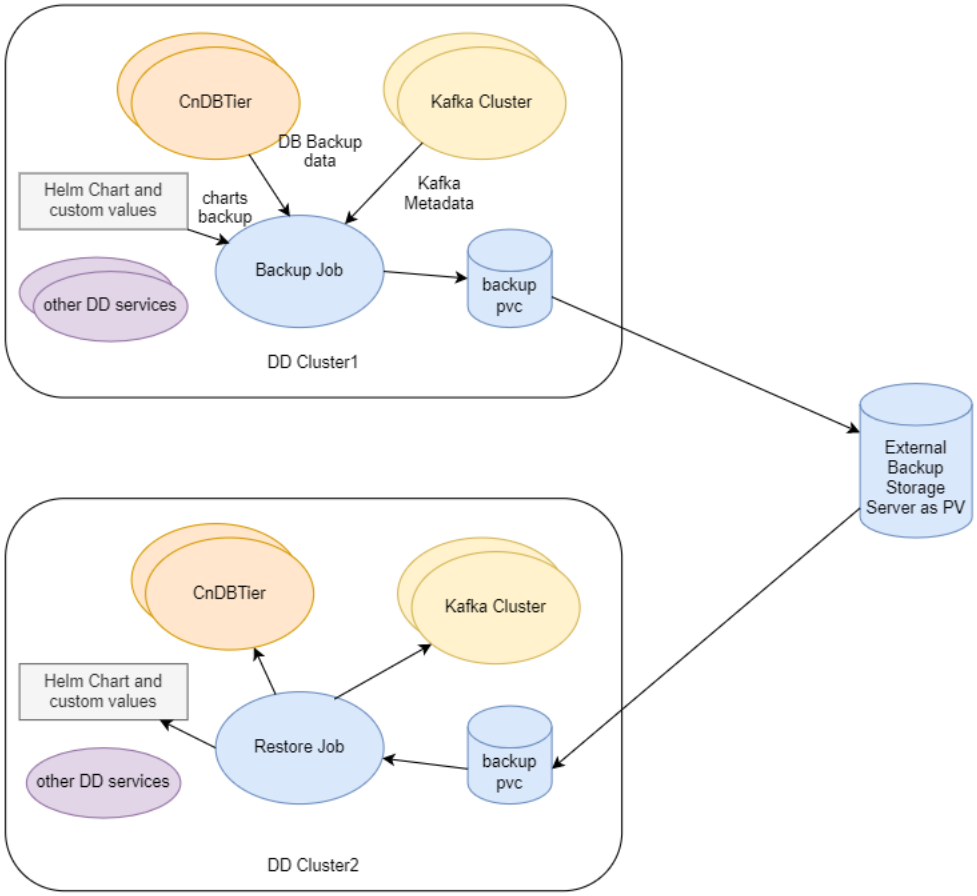
- OCNADD fault recovery
- OCNADD cluster migration
- OCNADD setup replication from production to development or staging
- OCNADD cluster upgrade to new CNE version or K8s version

The OCNADD backup contains the following data:

- OCNADD database(s) backup
- OCNADD Kafka metadata backup including the topics and partitions information

> **Note**
>
> If the deployed helm charts and the customized `ocnadd-custom-values.yaml` for the current deployment is stored in the customer helm or artifact repository, then the helm chart and `ocnadd-custom-values.yaml` backup is not required.

Figure 7-1 OCNADD Backup and Restore



OCNADD Database(s) Backup

The OCNADD database consists of the following:

- **Configuration data**: This data is exclusive for the given OCNADD instance. Therefore, an exclusive logical database is created and used by an OCNADD instance to store its configuration data and operator driven configuration. Operators can configure the OCNADD instance specific configurations using the Configuration UI service through the Cloud Native Configuration Console.
- **Alarm configuration data**: This data is also exclusive to the given OCNADD instance. Therefore, an exclusive logical database is created and used by an OCNADD Alarm service instance to store its alarm configuration and alarms.
- **Health monitoring data**: This data is also exclusive to the given OCNADD instance. Therefore, an exclusive logical database is created and used by an OCNADD Health monitoring service instance to store the health profile of various other services.

The database backup job uses the mysqldump utility.

The Scheduled regular backups helps in:

- Restoring the stable version of the data directory databases
- Minimize significant loss of data due to upgrade or rollback failure
- Minimize loss of data due to system failure

- Minimize loss of data due to data corruption or deletion due to external input
- Migration of the database information from one site to another site

OCNADD Kafka Metadata Backup

The OCNADD Kafka metadata backup contains the following information:

- Created topics information
- Created partitions per topic information

7.1.1 Fault Recovery Impact Areas

The following table shares information about impact of OCNADD fault recovery scenarios:

Table 7-1 OCNADD Fault Recovery Scenarios Impact Information

Scenario	Requires Fault Recovery or Reinstallation of CNE?	Requires Fault Recovery or Reinstallation of cnDBTier?	Requires Fault Recovery or Reinstallation of Data Director?
Scenario 1: Deployment Failure Recovering OCNADD when its deployment is corrupted	No	No	Yes
Scenario 2: cnDBTier Corruption	No	Yes	No However, it requires to restore the databases from backup and Helm upgrade of the same OCNADD version to update the OCNADD configuration. For example, change in cnDBTier service information, such as cnDB endpoints, DB credentials, and so on.
Scenario 3: Database Corruption Recovering from corrupted OCNADD configuration database	No	No	No However, it requires to restore the databases from old backup.
Scenario 4: Site Failure Complete site failure due to infrastructure failure, for example, hardware, CNE, and so on.	Yes	Yes	Yes
Scenario 5: Backup Restore in a Different Cluster Obtaining the OCNADD backup from one deployment site and restore it to another site	No	No	No However, it requires to restore the database.

7.1.2 Prerequisites

Before you run any fault recovery procedure, ensure that the following prerequisites are met:

- cnDBTier must be in a healthy state and available on a new or newly installed site where the restore needs to be performed
- Automatic backup should be enabled for OCNADD.
- Docker images used during the last installation or upgrade must be retained in the external data storage or repository
- The `ocnadd-custom-values.yaml` used at the time of OCNADD deployment must be retained. If the `ocnadd-custom-values.yaml` file is not retained, it is required to be recreated manually. This task increases the overall fault recovery time.

Important

Do not change DB Secret or CnDBTier MySQL FQDN or IP or PORT configurations during backup and restore.

7.2 Backup and Restore Flow

Important

1. It is recommended to keep the backup storage in the external storage that can be shared between different clusters. This is required, so that in an event of a fault, the backup is accessible on the other clusters. The backup job should create a PV/PVC from the external storage provided for the backup.
2. In case the external storage is not made available for the backup storage, the customer should take care to copy the backups from the associated backup PV in the cluster to the external storage. The security and connectivity to the external storage should be managed by the customer. To copy the backup from the backup PV to the external server, follow Verifying OCNADD Backup.
3. The restore job should have access to the external storage so that the backup from the external storage can be used for the restoration of the OCNADD services. In case the external storage is not available, the backup should be copied from the external storage to the backup PV in the new cluster. For information on the procedure, see Verifying OCNADD Backup.

> **Note**
>
> At a time, only one among the three backup jobs (ocnaddmanualbackup, ocnaddverify or ocnaddrestore) can be running. If any existing backup job is running, that job needs to be deleted to spawn the new job.
>
> ```
> kubectl delete job.batch/<ocnadd*> -n <namespace>
>
> where namespace = Namespace of OCNADD deployment
>       ocnadd* = Running jobs in the namespace (ocnaddmanualbackup,
> ocnaddverify or ocnaddrestore)
>
> Example:
>       kubectl delete job.batch/ocnaddverify -n ocnadd-deploy
> ```

Backup

1. The OCNADD backup is managed using the backup job created at the time of installation. The backup job runs as a cron job and takes the daily backup of the following:
   - OCNADD databases for configuration, alarms, and health monitoring
   - OCNADD Kafka metadata including topics and partitions, which are already created
2. The automated backup job spawns as a container and takes the backup at the scheduled time. The backup folder `OCNADD_Backup_DD-MM-YYYY_hh-mm-ss` is created and stored in the PV mounted on the path `/work-dir/backup` by the backup container.
3. On-demand backup can also be created by creating the backup container. For more information, see Performing OCNADD Manual Backup.
4. The backup can be stored on external storage.

Restore

1. The OCNADD restore job must have access to the backups from the backup PV/PVC.
2. The restore uses the latest backup file available in the backup storage if the BACKUP_FILE argument is not given.
3. The restore job performs the restore in the following order:
   a. Restore the OCNADD database(s) on the CnDBTier.
   b. Restore the Kafka metadata.

7.3 OCNADD Backup

The OCNADD Backup is of two types:

- Automated Backup
- Manual Backup

Automated Backup

- This is managed by the automated K8s job configured during the installation of the OCNADD. For more information, see "Create Data Director Backup Job" in *Oracle Communications Network Analytics Data Director Installation Guide*.
- It is a scheduled job and runs daily at the configured time to collect the OCNADD backup and creates the backup file `OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss.tgz`.

Manual Backup

- This is managed by an on-demand job.
- A new K8s job will be created on executing the Performing OCNADD Manual Backup procedure.
- The job completes after taking the backup. Follow Verifying OCNADD Backup procedure to verify the generated backup.

7.4 Performing OCNADD Backup Procedures

7.4.1 Performing OCNADD Manual Backup

Perform the following steps to take the manual backup:

1. Go to `custom-templates` folder inside the extracted ocnadd-release package and update the `ocnadd_manualBackup.yaml` file with the following information:
   a. Values for BACKUP_DATABASES can be set to ALL (i.e. healthdb_schema, configuration_schema, and alarm_schema) or the individual DB names can also be passed. By default, the value is 'ALL'.
   b. Values of BACKUP_ARG can be set to ALL, DB, or KAFKA. By default, the value is as ALL.
   c. Update other values as follows:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: ocnaddmanualbackup
  namespace: ocnadd-deploy     #---> update the namespace
spec:
  template:
    metadata:
      name: ocnaddmanualbackup
    spec:
      volumes:
      - name: backup-vol
        persistentVolumeClaim:
            claimName: backup-mysql-pvc
      - name: config-vol
        configMap:
          name: config-backuprestore-scripts
      serviceAccountName: ocnadd-deploy-gitlab-admin      #---> update
the service account name. Format:<namespace>-gitlab-admin
      containers:
```

```
      - name: ocnaddmanualbackup
        image: <repo-path>/ocnaddbackuprestore:1.0.7      #---> update
repository path
        volumeMounts:
          - mountPath: "work-dir"
            name: backup-vol
          - mountPath: "config-backuprestore-scripts"
            name: config-vol
        env:
          - name: HOME
            value: /tmp
          - name: DB_USER
            valueFrom:
              secretKeyRef:
                name: db-secret
                key: MYSQL_USER
          - name: DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: db-secret
                key: MYSQL_PASSWORD
          - name: BACKUP_DATABASES
            value: ALL
          - name: BACKUP_ARG
            value: ALL
        command:
        - /bin/sh
        - -c
        - |
          cp /config-backuprestore-scripts/*.sh /tmp
          chmod +x /tmp/*.sh
          mkdir /work-dir/backup
          echo "Executing manual backup script"
          bash /tmp/backup.sh $BACKUP_DATABASES $BACKUP_ARG
          ls -lh /work-dir/backup
      restartPolicy: Never
status: {}
```

2. Execute the below command to run the job:

   ```
   kubectl create -f ocnadd_manualBackup.yaml
   ```

7.4.2 Verifying OCNADD Backup

> **⚠ Caution**
>
> The connectivity between the external storage via either PV/PVC or network connectivity must be ensured.

To verify the backup, perform the following steps:

1. Go to the `custom-templates` folder inside the extracted ocnadd-release package and update the `ocnadd_verify_backup.yaml` file with the following information:

a. Sleep time is configurable, update it if required. (default sleep is 10m)

b. Update other values as follows:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: ocnaddverify
  namespace: ocnadd-deploy          #---> update the namespace
spec:
  template:
    metadata:
      name: ocnaddverify
    spec:
      volumes:
      - name: backup-vol
        persistentVolumeClaim:
            claimName: backup-mysql-pvc
      - name: config-vol
        configMap:
          name: config-backuprestore-scripts
      serviceAccountName: ocnadd-deploy-gitlab-admin      #---> update
the service account name. Format:<namespace>-gitlab-admin
      containers:
      - name: ocnaddverify
        image: <repo-path>/ocnaddbackuprestore:1.0.7      #---> update
repository path
        volumeMounts:
          - mountPath: "work-dir"
            name: backup-vol
          - mountPath: "config-mysql-scripts"
            name: config-vol
        env:
          - name: HOME
            value: /tmp
          - name: DB_USER
            valueFrom:
              secretKeyRef:
                name: db-secret
                key: MYSQL_USER
          - name: DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: db-secret
                key: MYSQL_PASSWORD
        command:
        - /bin/sh
        - -c
        - |
          cp /config-backuprestore-scripts/*.sh /tmp
          chmod +x /tmp/*.sh
          echo "Checking backup path"
          ls -lh /work-dir/backup
          sleep 10m
      restartPolicy: Never
status: {}
```

2. Run the following command to run the verify job for verifying the backup generated at the mounted PV by running inside the running container:

   ```
   kubectl create -f ocnadd_verify_backup.yaml
   ```

3. If the external storage is used as PV/PVC, get inside the ocnaddverify container using the following commands:
   a. `kubectl exec -it <verify_pod> -n <ocnadd namespace> -- bash`
   b. Change the directory to `/work-dir/backup` and inside the latest backup folder OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss.
   c. Verify the DB backup and Kafka metadata backup files

7.4.3 Obtain the OCNADD Backup files

1. Run the Verifying OCNADD Backup procedure to spawn the `verify_pod`.
2. Get into the running `ocnaddverify` pod to identify and retrieve the desire backup folder with the following commands:
   a. Access the pod

      ```
      kubectl exec -it <ocnaddverify-*> -n <namespace> -- bash
      ```

      where namespace = namespace of ocnadd

      ocnaddverify-* = is the verify pod in the namespace
   b. Change the directory to /work-dir/backup and identify the backup folder "OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss"
   c. Exit from the ocnaddverify pod
3. Copy the backup from the pod to the local bastion server by copying the OCNADD Database and kafka_metadata files from the ocnaddverify pod with the following commands:
   a. Onbastion host create a folder with the name similar to backup folder "OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss"
   b. Change to the folder created in step a.
   c. Use below command to copy backup files from the ocnaddverify pod To copy ocnadd Database backup:

      ```
      kubectl exec -n <namespace>  <verify_pod> -- cat/work-dir/backup/
      OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss/DB_BACKUP_DD-MM-YYYY_hh-mm-ss.sql  >
      DB_BACKUP_DD-MM-YYYY_hh-mm-ss.sql
      ```

      Example:

      ```
      kubectl exec -n ocnadd ocnaddverify-ncczz -- cat/work-dir/backup/
      OCNADD_BACKUP_11-11-2022_16-54-05/DB_BACKUP_11-11-2022_16-54-05.sql  >
      DB_BACKUP_11-11-2022_16-54-05.sql
      ```

To copy ocnadd Kafka metadata backup:

```
kubectl exec -n <namespace>  <verify_pod> -- cat/work-dir/backup/
OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss/kafka_metadata_DD-MM-YYYY_hh-mm-
ss.txt  > kafka_metadata_DD-MM-YYYY_hh-mm-ss.txt
```

Example:

```
kubectl exec -n ocnadd ocnaddverify-ncczz -- cat/work-dir/backup/
OCNADD_BACKUP_11-11-2022_16-54-05/
kafka_metadata_11-11-2022_16-54-05.txt  >
kafka_metadata_11-11-2022_16-54-05.txt
```

where, namespace = namespace of ocnadd

verify_pod = is the verify pod in the namespace

7.4.4 Copy and Restore the OCNADD backup

1. Get the ocnadd backup folder.
2. Execute the Verifying OCNADD Backup procedure to spawn the verify_pod.
3. Create a folder inside the verify_pod under directory path:/work-dir/backup/ with the name similar to the backup folder "OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss" .
   a. Access the pod with "kubectl exec -it <ocnaddverify-*> -n <namespace> -- bash".
   b. Create backup folder using command "mkdir -p /work-dir/backup/ OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss".
4. Copy the backup files from the local bastion server to the running ocnaddverify pod.
   a. Go to the ocnadd backup directory
   b. Use the following command to copy backup files inside the ocnaddverify pod:

      ```
      cat DB_BACKUP_DD-MM-YYYY_hh-mm-ss.sql | kubectl exec -i -n <namespace>
      <verify_pod> -- bash -c "cat > /work-dir/backup/OCNADD_BACKUP_DD-MM-
      YYYY_hh-mm-ss/DB_BACKUP_DD-MM-YYYY_hh-mm-ss.sql"
      ```

      Example:

      ```
      cat DB_BACKUP_11-11-2022_16-54-05.sql | kubectl exec -i -n ocnadd
      ocnaddverify-ncczz -- bash
            -c "cat >
            /work-dir/backup/OCNADD_BACKUP_11-11-2022_16-54-05/
      DB_BACKUP_11-11-2022_16-54-05.sql"
      ```

   c. To copy ocnadd Kafka metadata backup:

      ```
      cat kafka_metadata_DD-MM-YYYY_hh-mm-ss.txt | kubectl exec -i -n
      <namespace> <verify_pod> -- bash -c "cat  > /work-dir/backup/
      OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss/kafka_metadata_DD-MM-YYYY_hh-mm-
      ss.txt"
      ```

Example:

```
cat kafka_metadata_11-11-2022_16-54-05.txt | kubectl exec -i -n ocnadd
ocnaddverify-ncczz -- bash -c "cat > /work-dir/backup/
OCNADD_BACKUP_11-11-2022_16-54-05/
kafka_metadata_11-11-2022_16-54-05.txt"
```

5. Verify the backup has been copied by getting into the ocnaddverify pod and path: /work-dir/backup/OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss
6. Restore OCNADD using the procedure defined in Create OCNADD Restore Job.
7. Restart the ocnaddconfiguration, ocnaddalarm, and ocnaddhealthmonitoring pods.

7.5 Fault Recovery Scenarios

This chapter describes the fault recovery procedures for different recovery scenarios.

7.5.1 Scenario 1: Deployment Failure

This section describes how to recover OCNADD when the OCNADD deployment corrupts.

For more information, see Restoring OCNADD.

7.5.2 Scenario 2: cnDBTier Corruption

This section describes how to recover the cnDBTier corruption. For more information, see *Oracle Communication Cloud Native Core DBTier Fault Recovery Guide*. After the cnDBTier recovery, restore the OCNADD database from the previous backup.

To restore the OCNADD database, execute the procedure Create OCNADD Restore Job by setting BACKUP_ARG to DB.

7.5.3 Scenario 3: Database Corruption

This section describes how to recover from the corrupted OCNADD database.

Perform the following steps to recover the OCNADD configuration database (DB) from the corrupted database:

1. Retain the working ocnadd backup by following Obtain the OCNADD Backup files procedure.
2. Drop the existing Databases by accessing the MySql DB.
3. Perform the Copy and Restore the OCNADD backup procedure to restore the backup.

7.5.4 Scenario 4: Site Failure

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

Perform the following steps in case of a complete site failure:

1. Run the Cloud Native Environment (CNE) installation procedure to install a new Kubernetes cluster. For more information, see *Oracle Communications Cloud Native Environment Installation Guide*.

2. Run the cnDBTier installation procedure. For more information, see *Oracle Communications Cloud Native Core DBTier Installation and Upgrade Guide*.
3. For cnDBTier fault recovery, take a data backup from an older site and restore it to a new site. For more information about cnDBTier backup, see "Create On-demand Database Backup" and to restore the database to a new site, see "Restore DB with Backup" in *Oracle Communications Cloud Native Core DBTier Fault Recovery Guide*.
4. Restore OCNADD. For more information, see Restoring OCNADD.

7.5.5 Scenario 5: Backup Restore in a Different Cluster

This section describes how to obtain the OCNADD backup from one deployment site and restore it to another site.

Perform the following steps:

1. Obtain the OCNADD Backup files
2. Copy and Restore the OCNADD backup

7.6 Restoring OCNADD

Perform this procedure to restore OCNADD when a fault event has occurred or deployment is corrupted.

> **Note**
>
> This procedure expects the OCNADD backup folder is retained.

1. Get the retained backup folder "OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss".
2. Get the helm charts that was used in the earlier deployment.
3. Run the following command to uninstall the corrupted OCNADD deployment:
   a. helm uninstall <release_name> --namespace <namespace>
   b. Where,
      - <release_name> is a name used to track this installation instance.
      - <namespace> is the namespace of OCNADD deployment.

      Example:

      helm uninstall ocnadd --namespace ocnadd-ns
4. Install OCNADD using the helm charts that was used in the earlier deployment. For information about installing OCNADD using Helm, refer to *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide*.
5. To verify whether OCNADD installation is complete, perform the "Verify Data Director Installation" procedure as described in the *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide.*
6. Follow procedure Copy and Restore the OCNADD backup

7.7 Create OCNADD Restore Job

1. Restore the OCNADD database by executing the following steps:
   a. Go to the custom-templates folder inside the extracted ocnadd-release package and update the ocnadd_restore.yaml file based on the restore requirements:
      i. The value of BACKUP_ARG can be set to DB, KAFKA, and ALL. By default, the value is 'ALL'.
      ii. The value of BACKUP_FILE can be set to folder name which needs to be restored, if not mentioned the latest backup will be used.
      iii. Update other values as below:

```
apiVersion:batch/v1
kind:Job
metadata:
   name:ocnaddrestore
   namespace:ocnadd-deploy    #---> update the namespace
spec:
   template:
       metadata:
          name:ocnaddrestore
       spec:
         volumes:
          -name:backup-vol
           persistentVolumeClaim:
             claimName:backup-mysql-pvc
          -name:config-vol
            configMap:
              name:config-backuprestore-scripts
              serviceAccountName:ocnadd-deploy-gitlab-admin
#---> update the service account name. Format:<namespace>-gitlab-
admin
              containers:
              -name:ocnaddrestore
              image:<repo-path>/ocnaddbackuprestore:1.0.7      #---
>update repository path
              volumeMounts:
                    -mountPath:"work-dir"
                     name:backup-vol
                    -mountPath:"config-backuprestore-
scripts"
                     name:config-vol

                 env:
                  -name:HOME
                   value:/tmp
                  -name:DB_USER
                   valueFrom:
                      secretKeyRef:
                         name:db-secret
                          key:MYSQL_USER
```

```
                        -name:DB_PASSWORD
                           valueFrom:
                             secretKeyRef:
                                name:db-secret
                                key:MYSQL_PASSWORD

                         -name:BACKUP_ARG
                             value:ALL
                         -name:BACKUP_FILE
                            value:
                    command:
                    - /bin/sh
                    - -c
                    - |
                    cp /config-backuprestore-scripts/*.sh /tmp
                    chmod +x /tmp/*.sh
                    echo "Executing restore script"
                    ls -lh /work-dir/backup
                    bash /tmp/restore.sh $BACKUP_ARG $BACKUP_FILE
                    sleep 15m

                restartPolicy:Never
        status:{}
```

2. Execute the below command to run the restore job:

   ```
   kubectl create -f ocnadd_restore.yaml
   ```

3. Restart the ocnaddconfiguration, ocnaddalarm, and ocnaddhealthmonitoring pods once the restore job gets completed.

> **Note**
>
> If the backup is not available for the mentioned date, the pod will be in an error state, notifying the Backup is not available for the given date: $DATE, in such case provide the correct backup dates and repeat the procedure.

7.8 Configuring Backup and Restore Parameters

To configure backup and restore parameters, configure the parameters listed in the following table:

Table 7-2 Backup and Restore Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
BACKUP_STORAGE	STRING	-	20Gi	M	Persistent Volume storage to keep the OCDD backups

Table 7-2 (Cont.) Backup and Restore Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
MYSQLDB_NAMESPACE	STRING	-	occne-cndbtierone	M	Mysql Cluster Namespace
BACKUP_CRONEXPRESSION	STRING	-	0 8 * * *	M	Cron expression to schedule backup cronjob
BACKUP_ARG	STRING	-	ALL	M	KAFKA , DB or ALL backup
BACKUP_FILE	STRING	-	-	O	Backup folder name which needs to be restored
BACKUP_DATABASES	STRING	-	ALL	M	Individual databases or all databases backup that need to be taken