Oracle® Communications

Network Analytics Data Director Installation and Upgrade Guide

Release 22.0.0
F71104-03
February 2023

Oracle Communications Network Analytics Data Director Installation and Upgrade Guide, Release 22.0.0

F71104-03

Contents

1 Introduction

Overview 1-1
References 1-2

2 Installing OCNADD

Prerequisites 2-1
  Software Requirements 2-1
  Environment Setup Requirements 2-3
  Resource Requirements 2-5
Installation Sequence 2-6
  Pre-Installation Tasks 2-7
    Creating OCNADD Namespace 2-7
    Creating Service Account, Role, and RoleBinding 2-8
    Configuring OCNADD Database 2-10
    Configuring Secrets for Accessing OCNADD Database 2-12
    Configuring SSL or TLS Certificates 2-13
  Installation Tasks 2-18
    Downloading OCNADD Package 2-18
    Pushing the Images to Customer Docker Registry 2-19
    Installing OCNADD Package 2-21
    Verifying OCNADD Installation 2-26
    Creating OCNADD Topics 2-27
    Installing OCNADD GUI 2-27

3 Customizing OCNADD

Global Parameters 3-2
Aggregation Service Parameters 3-6
Configuration Service Parameters 3-12
Consumer Adapter Service Parameters 3-13
Health Monitoring and Alarm Service Parameters 3-18
Admin Service Parameters 3-21

Kafka Configuration Parameters 3-27
Backend Router Parameters 3-32

4 Upgrading OCNADD

Supported Upgrade Path 4-1
Preupgrade Tasks 4-1
OCNADD Upgrade 4-2
    Hotfix Upgrade 4-3

5 Rollback OCNADD

6 Uninstalling OCNADD

Verifying Uninstallation 6-3

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.

What's New in This Guide

This section lists the documentation updates for Release *22.0.0* in *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide*.

Release 22.0.0 - F71104-03, February 2023

Updated the document with editorial changes.

Release 22.0.0 - F71104-02, February 2023

Updated the document with editorial changes.

Release 22.0.0 - F71104-01, December 2022

This is the first release of the document.

Updated the document with editorial changes.

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
CNCC	Cloud Native Core Console
CNE	Cloud Native Environment
CSP	Communication Service Provider
KPI	Key Performance Indicator
MPS	Messages Per Second
NDB	Network Data Broker
NF	Network Function
NRF	Network Repository Function
OHC	Oracle Help Center
OSDC	Oracle Service Delivery Cloud
SCP	Service Communication Proxy
SVC	Services
TLS	Transport Layer Security
URI	Uniform Resource Identifier

1 Introduction

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

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 RoleBinding
	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

Table 1-1 (Cont.) Installation Overview

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

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 and Installation Guide*
- *Oracle Communications Cloud Native Core Disaster Recovery Guide*
- *Oracle Communications Cloud Native DBTier Installation Guide*
- *Oracle Communications Cloud Native Core Console Installation and Upgrade 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.

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.

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 22.0.0 supports OCCNE 22.3.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	22.3.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
```

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 *Oracle Communications Cloud Native Environment Installation Guide*.

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 can be specified using the `ocnaddServiceAccountName` parameter. If it is not specified in the `ocnadd/values.yaml` file, OCNADD creates a default service account at the time of installation. You can also create a service account as described in Creating Service Account, Role, and RoleBinding, and then update the `ocnaddServiceAccountName` accordingly in the `ocnadd/values.yaml` file.

cnDBTier Requirement

OCNADD supports cnDBTier 22.3.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	22.0.0
OCNADD-ConsumerAdapter	ocnaddconsumeradapter	22.0.0
OCNADD-EgressGW	ocnaddegressgateway	22.0.0
OCNADD-AGG	ocnaddnrfaggregation ocnaddscpaggregation	22.0.0
OCNADD-Alarm	ocnaddalarm	22.0.0
OCNADD-HealthMonitoring	ocnaddhealthmonitoring	22.0.0
OCNADD-Kafka	kafka-broker-x	22.0.0
OCNADD-Admin	ocnaddadminservice	22.0.0
OCNADD-Backendrouter	ocnaddbackendrouter	22.0.0
OCNADD-GUI	ocnaddgui	22.0.0

Note:

The service images are prefixed with the OCNADD release name.

Resource Requirements

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

Table 2-5 OCNADD Resource Requirements

Service	vCPU Req	vCPU Limit	Memory Req(Gi)	Memory Limit (Gi)	Min Replica	Max Replica	Partitions	Topic Name
ocnaddconfiguration	1	2	1	2	1	1		
ocnaddalarm	1	2	1	2	1	1		
ocnaddadmin	1	1	1	1	1	1		
ocnaddhealthmonitoring	1	2	1	2	1	1		
ocnaddbackendrouter	1	2	1	2	1	1		
ocnaddscpaggregation	3	3	4	4	1	2	3	SCP
ocnaddnrfaggregation	3	3	4	4	1	2	3	NRF

Table 2-5 (Cont.) OCNADD Resource Requirements

Service	vCPU Req	vCPU Limit	Memory Req(Gi)	Memory Limit (Gi)	Min Replica	Max Replica	Partitions	Topic Name
ocnaddadapter	2.5	2.5	8	8	3	5	6	MAIN
ocnaddegressgateway	4	4	8	8	2	3		
ocnaddkafka	4	4	24	24	3	3		
zookeeper	1	2	2	2	3	3		

Note:

To deploy beyond 50000 Messages Per Second (MPS), Standard Profile (KAFKA) is recommended.

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
<app-name>-gw	400	800
ocnaddadminservice	100	200
ocnaddalarm	100	500
ocnaddhealthmonitoring	100	500
ocnaddscpaggregation	100	500
ocnaddnrfaggregation	100	500
ocnaddconfiguration	100	500

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.

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.

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.

Creating Service Account, Role, and RoleBinding

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

Important:

The steps described in this section are optional and you can skip it in any of the following scenarios:

- If service accounts are created automatically at the time of OCNADD deployment.
- If the global service account with the associated role and role-bindings is already configured or if you are using any internal procedure to create service accounts.
  If a service account with necessary rolebindings is already available, then update the `ocnadd/values.yaml` with the account details before initiating the installation procedure. In case of incorrect service account details, the installation fails.

Create Service Account

To create the global service account:

1. Create an OCNADD resource file:

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

   Example:

   ```
   vi ocnadd-sampleserviceaccount-template.yaml
   ```

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

   Note:

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

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

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

Define Permissions using Role

To define permissions using roles:

1. Create an OCNADD resource file:

   ```
   vi <ocnadd sample role file>
   ```

   Example:

   ```
   vi ocnadd-samplerole-template.yaml
   ```

2. Update the `ocnadd-samplerole-template.yaml` with the role specific information:

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

Create RoleBindings

To bind the roles with the service account:

1. Create an OCNADD rolebinding resource file:

   ```
   vi <ocnadd sample rolebinding file>
   ```

   Example:

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

2. Update the `ocnadd-sample-rolebinding-template.yaml` with the role binding specific information:

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

```
    name: <helm-release>-serviceaccount
    namespace: <namespace>
```

Create resources

Run the following commands to create resources:

```
kubectl -n <namespace> create -f ocnadd-sample-serviceaccount-
template.yaml;



kubectl -n <namespace> create -f ocnadd-sample-role-template.yaml;


kubectl -n <namespace> create -f ocnadd-sample-rolebinding-
template.yaml
```

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

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 a privileged 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 privileged user is already available, then update the credentials, such as username and password (base64 encoded) in `ocnadd/values.yaml`.
> - If the privileged user is not available, then create it using the following procedure. Once the user is created, update the credentials for the user in `ocnadd/values.yaml`.

Creating Database

To create database:

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

> **Note:**
>
> Use the namespace in which the DBTier 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 privileged user, run the following command:

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

Example:

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

where:

<ocnadd> is the privileged username and <ocnadd> is the password for MySQL privileged user

4. Run the following command to grant the necessary permissions to the privileged user:

```
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:

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

.

6. Update the following parameters in the `ocnadd-secret-hook.yaml` file to change the 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
   ```

Update Database Name

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:

   ```
   helm-charts/ocdd-db-resource.sql
   ```

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

   Note:

   By default, the database names are configuration_schema, alarm_schema, and healthdb_schema for the respective services.

3. Update the database IP and database name in `ocnadd/values.yaml`.

   ```
   database:
     db_ip: 10.20.30.40 (Change DB IP)
     db_port: 3306 (If using a different port for DB, change it. By
   default, DB port is 3306)
     configuration_db: configuration_schema (Change to new DB name)
     alarm_db: alarm_schema (Change to new DB name)
     health_db: healthdb_schema (Change to new DB name)
   ```

Note:

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

Configuring Secrets for Accessing OCNADD Database

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

Configuring SSL or TLS Certificates

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.yaml` 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.yaml` 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>
    ```

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>
    ```

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).

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.

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 Cloud Native Core Network Analytics Data Director 22.0.0.0.0" 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.

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 Environment (OCCNE) 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.

Table 2-7 Docker Images for OCNADD

Service Name	Docker Image Name	Image Tag
OCNADD-Configuration	ocnaddconfiguration	22.0.0
OCNADD-ConsumerAdapter	<app-name>-adapter	22.0.0
OCNADD-EgressGW	<app-name>-gw	22.0.0
OCNADD-AGG	ocnaddnrfaggregation ocnaddscpaggregation	22.0.0
OCNADD-Alarm	ocnaddalarm	22.0.0
OCNADD-HealthMonitoring	ocnaddhealthmonitoring	22.0.0
OCNADD-Kafka	kafka-broker-x	22.0.0
OCNADD-Admin	ocnaddadminservice	22.0.0
OCNADD-UIRouter	ocnaddbackendrouter	22.0.0
OCNADD-GUI	ocnaddgui	22.0.0

Note:

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

Pushing Docker Images

To Push the images to customer docker resgistry, perform the following steps:

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

   ```
   tar -xvzf ocnadd-pkg-22.0.0.0.0.tgz
   ```

   The directory consists of the following:
   - **OCNADD Docker Images File:**

     ```
     ocnadd-images-22.0.0.tar
     ```

   - **Helm File:**

     ```
     ocnadd-22.0.0.tgz
     ```

   - **Readme txt File:**

     ```
     Readme.txt
     ```

2. Load the `ocnadd-images-22.0.0.tar` file into the docker system

   ```
   docker load --input /IMAGE_PATH/ocnadd-images-22.0.0.tar
   ```

   For CNE 1.8.0 and later, use the following command:

   ```
   podman load --input /IMAGE_PATH/ocnadd-images-22.0.0.tar
   ```

3. To verify if the image is loaded correctly, run the following command:

   ```
   docker images
   ```

4. Create a new tag for each imported image and push the image to the customer docker registry by entering the following command:

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

   Note:

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

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

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

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 values.yaml

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

Perform Kafka Pre-Install Configuration

1. Clean the script file.
   Script files might get encoded in the Windows format if they are pushed from the Windows Git client or opened with editors that have EOF encoding in the Windows format. Run the following commands:

   ```
   chmod 755 <chartpath>/charts/ocnaddkafka/scripts/start-service.sh
   dos2unix <chartpath>/charts/ocnaddkafka/scripts/start-service.sh
   ```

   If dos2unix is not available in your system, run the following command:

   ```
   sed -i 's/\r$//' <chartpath>/charts/ocnaddkafka/scripts/start-
   service.sh
   ```

2. Select the required brokers for deployment.
   By default, the deployment comes with three brokers. If the addition or removal of brokers is required, perform the following steps:
   a. To add a broker, run the following command:

      ```
      cp <chartpath>/charts/ocnaddkafka/default/
      ocnaddkafkaBrokerX.yaml <chartpath>/charts/ocnaddkafka/templates
      ```

      Example:

      ```
      cp <chartpath>/charts/ocnaddkafka/default/
      ocnaddkafkaBroker4.yaml <chartpath>/charts/ocnaddkafka/templates
      ```

   b. To remove a broker, run the following command:

      ```
      rm <chartpath>/charts/ocnaddkafka/default/ocnaddkafkaBrokerX.yaml
      ```

      Example:

      ```
      rm <chartpath>/charts/ocnaddkafka/default/ocnaddkafkaBroker3.yaml
      ```

3. To change the profiles of the brokers, edit the respective values (CPU, memory, storage) in `values.yaml`.

   ```
   location : <chartpath>/charts/ocnaddkafka/values.yaml
   ```

   If any formatting or indentation issues occur while editing, refer to the files in `<chartpath>/charts/ocnaddkafka/default`.

4. To create a secret for the Brokers-Zookeeper connection, run the following command:

   ```
   kubectl create secret generic jaas-secret --from-
   literal=jaas_password=ocnadd -n $nameSpace
   ```

   Example:

   ```
   kubectl create secret generic jaas-secret --from-
   literal=jaas_password=ocnadd -n ocnadd-deploy
   ```

5. To create the configmap, run the following command:

   ```
   kubectl create configmap allfiles-configmap  --from-file=<filepath> -n
   <namespace>
   ```

   The name used for the configmap is `allfiles-configmap`. The scripts file is placed under `<chartpath>/charts/ocnaddkafka/scripts/`.

   Example:

   ```
   kubectl create configmap allfiles-configmap --from-file=<chartpath>/
   charts/ocnaddkafka/scripts/ -n <namespace>
   ```

6. To configure `storageClass`, update the `storageClass` in the following files with the respective storage class name of the TANZU platform, for example, `zfs-storage-policy`.

   Note:

   This step is specific to the TANZU platform. Skip this step if you are installing OCNADD on OCCNE. For OCCNE, the default `storageClass` is `standard`.

   ```
    <chartpath>/charts/ocnaddkafka/templates/ocnadd-zookeeper.yaml
            <chartpath>/charts/ocnaddkafka/templates/
   ocnaddkafkaBrokerX.yaml  ----> where X stands for Broker1, Broker2,
   Broker3 ..etc
            <chartpath>/charts/ocnaddkafka/default/ocnaddkafkaBrokerx.yaml
   ----> where X stands for Broker1, Broker2, Broker3 ..etc
   ```

Configure OCNADD Backup Cronjob

1. Configure the `mysqlNameSpace` and `storageClass` details in `<chartpath>/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 `<chartpath/charts/ocnaddbackuprestore/values.yaml`.
   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. By default, the value is as `ALL`. `PURGE_DAYS` sets the backup retention period. The default value is 7 days.

   Example:

```
  env:
           BACKUP_STORAGE: 20Gi
           BACKUP_CRONEXPRESSION: "0 8 * * *"
           BACKUP_DATABASES: ALL
           PURGE_DAYS: 7
           STORAGE_CLASS: standard       #---> this should be same
as cluster.storageClass
```

   Once the deployment is successful, the cronjob is spawned based on the `CRONEXPRESSION` mentioned in the `<chartpath>/charts/ocnaddbackuprestore/values.yaml`.

   For more information on backup and restore, refer to *Oracle Communications Network Analytics Data Director Backup and Disaster Recovery Guide*.

Install Helm Chart

Run any of the following `helm install` commands:

- In the case of Helm 2:

```
helm install <helm-repo> --name <deployment_name> --namespace
<namespace_name> --version <helm_version>
```

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

```
helm3 install <release name> --namespace <namespace> <helm-repo>/chart_name
--version <helm_version>
```

- In case charts are extracted and Helm is used:

```
helm install <release name> --namespace <namespace> <chartpath>
```

where:

helm_chart is the location of the helm chart extracted from `ocnadd-22.0.0.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-22.0.0 --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
```

Perform Kafka Post-Install Configuration

1. Update the EXTERNAL-IP IPs in `<chartpath>/charts/ocnaddkafka/values.yaml`:
   a. Run the following command to get the EXTERNAL-IP details for the Kafka brokers:

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

      Sample output:

      ```
      kafka-broker1   LoadBalancer   10.20.30.40   10.xx.xx.xx
              9092:30946/TCP,9093:31912/TCP,9094:30663/TCP
      ```

   b. update the following `kafkaBrokerX.advertiseListeners1` parameter of each broker with respective EXTERNAL-IP details captured in the previous step:

      ```
      advertiseListeners1: PLAINTEXT://kafka-broker1:9092,SSL://kafka-
      broker1:9093,SASL_SSL://kafka-broker1:9094
      ```

      Example

      For EXTERNAL-IP=10.xx.xx.xx of Kafka broker1

      ```
      advertiseListeners1: PLAINTEXT://10.xx.xx.xx:9092,SSL://
      10.xx.xx.xx:9093,SASL_SSL://10.xx.xx.xx:9094
      ```

> **Note:**
>
> Other ports accessibility will be blocked if only EXTERNAL-IP of SASL_SSL port(9094) is updated.

2. Upgrade the helm chart with the following command:

   ```
   helm upgrade <release-name> -n <namespace> <chartpath>
   ```

   > **Note:**
   >
   > Once the charts are upgraded, the brokers restart is expected. In case the required brokers images are not available locally the Kafka brokers might take a few minutes until the zookeepers pull the image.

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 disaster 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*.

Creating OCNADD Topics

Create topics (MAIN, SCP, and NRF) using admin service, before starting data ingestion. For more details on topic and partitions see, "Kafka PVC Storage Requirements" section of *Oracle Communications Network Analytics Data Director Benchmarking Guide*.

To create a topic connect to any worker node and send a **POST** curl request to the **API Endpoint** described below.

API Endpoint : **<ClusterIP:Admin Port>/ocnadd-admin-svc/v1/topic**

```
{
    "topicName":"<topicname>",
    "partitions":"3",
    "replicationFactor":"2",
    "retentionMs":"120000"
}
```

Note:

- In case worker node access is not available then the adminservice Service-Type can be changed to LoadBalancer or NodePort in the admin service `values.yaml` (helm upgrade is required for any such changes)
- For Loadbalancer service ensure that the admin port is not blocked in the cluster.

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

Perform the following steps to install OCNADD GUI:

1. Extract the helm charts from `ocnaddgui-pkg-Releasenumber.tgz` provided inside the `ocnadd-pkg-22.0.0.0.0.tgz` package.
2. Update `helm-charts/values.yaml` with the namespace and clusterName in which OCNADD is installed.
3. Update the `ocnaddgui` image repo path REPO_HOST_PORT in `helm-charts/values.yaml`.
4. Run the following command to install the OCNADD GUI:

   ```
   helm install <chart-name> helm-charts/ -n <namespace>
   ```

   Example:

   ```
   helm install ocnadd-gui helm-charts/ -n ocnadd-deploy
   ```

5. Run the following command to verify the OCNADD GUI installation:

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

   Example:

   ```
   kubectl get all -n ocnadd
   ```

Note:

At present, the OCNADD GUI service supports single cluster deployments only.

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 Core 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 Core Console Installation and Upgrade Guide*.

3

Customizing OCNADD

This chapter describes how to customize the Oracle Communications Network Analytics Data Director (OCNADD) deployment 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.

Perform the following procedure to customize the values.yaml files as per requirements for both parent and sub-charts.

1. Ensure that you have the Data Director charts tgz file, which is available in the extracted release package. For information about how to download the release package from MOS, see Downloading OCNADD Package.
2. Extract the Data Director charts tgz file if not already extracted.
    a. Change the directory to ocnadd to access the parent `values.yaml`. This file is used to customize the deployment parameters during installation.

       Change the following parameters in the file and save the file.:

       ```
       1. Update the repository path in

           global.env.repo.REPO_HOST_PORT: <customer repository path>

       2. Update the CLUSTER-INFO parameters
           a) cluster.clusterName: <customer cluster name>
           b) cluster.nameSpace.name: <created namespace of DD>

       3. Change the Prometheus Monitoring Details, bases on the desired MPS
       profile, default threshould MPS is 40K
          cluster.mps: 40000
       ```

3. Customize the rules file `<chartpath>/templates/ocnadd-alerting-rules.yaml`:
    - If OCNADD is to be installed in OCCNE Setup, then all the services will be monitored by Prometheus By default. So There will not be any Modifications in the Helm Chart. All the Prometheus Alert Rules Present in Helm Chart will be Updated in Prometheus Server. (Here the Label Used to Update the Prometheus Server is **"role: cnc-alerting-rules"**, which is added By Default in Helm Charts)
    - If OCNADD is to be installed in Tanzu Setup, then modify the "metadata.labels" value in <chartpath>/templates/ocnadd-alerting-rules.yaml file as below, Example **"release: prom-operator" instead of "role: cnc-alerting-rules"**,

      To obtain the labels details use the below command:

      ```
      kubectl get prometheus <Prometheus_Configuration_NAME> -n
      <Prometheus_Namespace> -o=jsonpath='{.spec.ruleSelector.matchLabels}'
      ```

Example:

```
$ kubectl get prometheus prom-operator-kube-prometh-prometheus -
n occne-infra -
o=jsonpath='{.spec.ruleSelector.matchLabels}'{"release: prom-
operator"}
```

Sample Alert File:

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  labels:
    release: prom-operator
  name: ocnadd-alerting-rules
  namespace: {{ .Values.global.cluster.nameSpace.name }}
```

4. Update the following default parameters of helm charts before installation:

Service Name	Parameter Name	Parameter Value	FilePath
Admin Service	OCNADD_ADAPTER_THIRD_PARTY_INTERVAL	2000	ocnadd/charts/ocnaddadminsvc/values.yaml
Admin Service	OCNADD_ADAPTER_MAX_PARTITION	20	ocnadd/charts/ocnaddadminsvc/values.yaml
Admin Service	OCNADD_ADAPTER_MAX_POLL_RECORDS	25 (is the default value) [50-600] (when Message size > 3000 Bytes and Replication factor >1 for MAIN topic)	ocnadd/charts/ocnaddadminsvc/values.yaml
Admin Service	OCNADD_DEPLOYMENT_TERMINATION_GRACE	5	ocnadd/charts/ocnaddadminsvc/values.yaml

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

Table 3-1 (Cont.) Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
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
nodeName	STRING	-	occne-ocdd-k8s-node-x	O	Defines the worker NodeName of the k8 cluster.
OCNADD_SSL_KEY_TYPE	STRING	-	PKCS12	M	Type of ssl key
OCNADD_SSL_TRUST_TYPE	STRING	-	PKCS12	M	Type of ssl store
OCNADD_TRUST_KEYSTORE	STRING	true/false	true	M	Enable trust keystore
OCWEBCLIENT_TIMEOUT	INTEGER	-	30	M	Webclient timeout in seconds
OCWEBCLIENT_KEEPALIVE_IDLE	INTEGER	-	90	M	Webclient keepalive idle time in seconds
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

Table 3-1 (Cont.) Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
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
global.cluster.nameSpace.name	STRING	-	ocnadd-deploy	M	Namespace name
global.cluster.mysqlNameSpace	STRING	-	occne-cndbtierone	O	DB Tier Namespace
global.cluster.secret.name	STRING	-	*****	M	DB secret name
global.cluster.secret.data.dbUsername	STRING	-	*****	M	DB User name (value shoud be converted into base64)
global.cluster.secret.data.dbPassword	STRING	-	*****	M	DB Password (value shoud be converted into base64)
global.cluster.serviceAccount.create	BOOLEAN	true/false	true	M	To create a ServiceAccount (true/false)
global.cluster.serviceAccount.name	STRING	-	ocnadd-deploy-sa	M	Service Account Name used during RBAC authorization creation
global.cluster.clusterRole.create	BOOLEAN	true/false	true	M	To create clusterRole (true/false)
global.cluster.clusterRole.name	STRING	-	ocnadd-deploy-cr	M	ClusterRole name used during RBAC authorization creation
global.cluster.clusterRoleBinding.create	BOOLEAN	true/false	true	M	To create clusterRoleBinding (true/false)

Table 3-1 (Cont.) Global Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
global.cluster.clusterRoleBinding.name	STRING	-	ocnadd-deploy-crb	M	ClusterRoleBinding name used during authorization creation
global.cluster.prometheusScrapePort	INTEGER	-	9000	O	Port to scape metrics required if metrics enabled
global.cluster.prometheusPortName	STRING	-	cnc-metrics	O	Role required to define in alert rules yaml
global.cluster.max_latency	FLOAT	-	0.05	M	Max latency range of 50ms
global.cluster.memory_threshold	INTEGER	[0-100]	70	M	Max Threshold limit for memory
global.cluster.cpu_threshold	INTEGER	[0-100]	70	M	CPU max threshold limit
global.cluster.mps	INTEGER	-	5000	M	Default MPS rate
global.cluster.egwGroupLatencyMessageCountMax	INTEGER	-	50	M	Max latency count for Egress Gateway
serviceAccount.create	BOOLEAN	true/false	true	M	Disable if Service Account already created and not required to create again
clusterRole.create	BOOLEAN	true/false	true	M	ClusterRole creation parameter make it false if already created
clusterRoleBinding.create	BOOLEAN	true/false	true	M	ClusterRoleBinding creation parameter make it false if already created

Table 3-1 (Cont.) Global 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.cluster.secret.name	STRING	-	db-secret	M	Name of database secret object
ocnaddhelmhook.name	STRING	-	`ocnaddhelmhook`	M	Helm Hook Name
ocnaddhelmhook.container.name	STRING	-	`ocnaddhelmhook`	M	Container Name of Helm Hook Job
ocnaddhelmhook.container.image	STRING	-	*****	M	Image will be depending on setup and image repo
ocnaddhelmhook.container.imagePullPolicy	STRING	IfNotPresent/ Always/Never	IfNotPresent	M	Image Pull Policy

Aggregation Service Parameters

Table 3-2 Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_AGGREGATION_NAME	STRING	-	ocnaddnrfaggregation	M	Name of the application
OCNADD_AGGREGATION_ACTIVE_PROFILE	STRING	dev, prod	prod	M	Active Profile to be used by the application.
OCNADD_AGGREGATION_SERVICE_SOURCE_TOPIC	STRING	SCP/NRF	SCP	M	Source topic details for Aggregation Service (i.e., To Read)

Table 3-2 (Cont.) Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_AGGREGATION_SERVICE_SOURCE_NAME	STRING	-	AGGREGATION_SOURCE	M	Name of the Kafka Stream Source Node.
OCNADD_AGGREGATION_SERVICE_PROCESSROR_NAME	STRING	-	AGGREGATION_PROCESSOR	M	Name of the Kafka Stream Processor Node.
OCNADD_AGGREGATION_SERVICE_SINK_NAME	STRING	-	AGGREGATION_SINK	M	Name of the Kafka Stream Sink Node.
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.
OCNADD_AGGREGATION_LOG_FILENAME	STRING	-	logs/ aggregation-service.log	O	Location where the application specific logs are stored.
OCNADD_AGGREGATION_APPLICATION_NAME	STRING	-	OCNADD	O	Aggregation Service Pod name to capture metrics related information.
OCNADD_AGGREGATION_MICROSERVICE_NAME	STRING	ocnadd-aggregation-service	O	Aggregation Service Name to communicate & Capture Metrics related information.	

Table 3-2 (Cont.) Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_AGGREGATION_ENABLED_SCHEDULING_JOBS	BOOLEAN	true, false	false	O	Parameters to enable scheduling Jobs (i.e., Calculating the Metrics) for CPU, Memory, Kafka Lag and Kafka Streams.
OCNADD_METRICS_CPU_THRESHOLD	FLOAT	[0, 1]	0.75	O	1. Parameter to compute the CPU related metrics of the application. 2. Work only when OCNADD_AGGREGATION_ENABLED_SCHEDULING_JOBS is enabled.

Table 3-2 (Cont.) Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_METRICS_MEM_THRESHOLD	FLOAT	[0, 1]	0.95	O	1. Parameter to compute the Memory related metrics of the application. 2. Work only when OCNADD_AGGREGATION_ENABLED_SCHEDULING_JOBS is enabled.
ALARM_SERVICE_URL	STRING	-	http://ocnaddalarm:9099	M	Alarm Service API Root to raise alarm
OCNADD_PRODUCER_SERVER_SSL_ENABLED	BOOLEAN	true, false	false	M	Parameter to enable SSL support for the application.
KAFKA_PRODUCER_SECURITY_PROTOCOL	STRING	-	PLAINTEXT	M	Kafka Producer Secutiry Protocol.
OCNADD_PRODUCER_SSL_KEYSTORE_LOCATION	STRING	-	<location in pod>	M	SSL Key store file location.
OCNADD_PRODUCER_SSL_TRUSTSTORE_LOCATION	STRING	-	<location in pod>	M	SSL Trust store file location.
KAFKA_PRODUCER_SSL_PROTOCOL	STRING	-	TLSv1.3	M	SSL Protocol
KAFKA_PRODUCER_SASL_MECHANISM	STRING	-	PLAIN	M	Kafka Producer SSAL Mechanism.

Table 3-2 (Cont.) Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
KAFKA_PRODUCER_SSL_CLIENT_AUTH	BOOLEAN	true, false	false	M	Kafka SSL client authentication.
KAFKA_BOOTSTRAP_SERVER	STRING	-	kafka-broker1:9092	M	Kafka Boot strap server address.
AGGREGATION_SERVICE_KAFKA_APPLICATION_ID	STRING	-	ocnadd-aggregation-service	M	Aggregation Service Kafka Stream application Id (By using this name Kafka Consumer Group will be created).
AGGREGATION_SERVICE_KAFKA_GROUP_NAME	STRING	-	ocnadd-aggregation-service-group	M	Aggregation Service Kafka Consumer Group Id.
KAFKA_STREAM_STATE	STRING	-	/tmp/ocnadd/kafka/state	M	Location to store any stateful information by the application.
KAFKA_REPLICATION_FACTOR	INTEGER	>= 1	1	M	The replication factor for changelog topics and repartition topics created by the application.
KAFKA_OFFSET_CONFIG	STRING	latest, earliest	latest	M	A flag to tell Kafka application from where to start reading offsets in case you do not have any 'commit' yet.

Table 3-2 (Cont.) Aggregation Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
KAFKA_ENABLE_AUTO_COMMIT	STRING	true, false	false	M	A flag for consumer auto commit, to guarantee at-least-once processing semantics.
KAFKA_AUTOCOMMIT_INT_CONFIG	INTEGER	-	10000	M	A hard auto commit by the Kafka set in the application.
KAFKA_COMMIT_INT_CONFIG	INTEGER	-	10000	M	The frequency with which to save the position (offsets in source topics) of tasks.
KAFKA_NUMBER_THREADS_CONFIG	INTEGER	>= 1	2	M	The number of threads to execute stream processing.
KAFKA_MAX_AGE_CONFIG	INTEGER	-	10000	M	The period of time in milliseconds after which we force a refresh of metadata.
OCNADD_AGGREGATION_HEALTH_SVC_TYPE	STRING	-	AGGREGATION	M	Service Type
OCNADD_TRUST_KEYSTORE	BOOLEAN	true, false	false	M	Enable to secure connection via OCWeb Client.
OCNADD_TRUST_CLIENT_TRUST_STORE	STRING	-	<location of truststore>	M	OCWeb Client Trust Store location.

Configuration Service Parameters

Table 3-3 Configuration Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_CONFIGURATION_PROFILE	STRING	dev,prod	prod	O	Application profile to use
ALARM_SERVICE_URL	STRING	-	http://ocnaddalarm:9099	M	Alarm Application URL
CONFIGURATION_SSL_ENABLED	BOOLEAN	-	FALSE	M	whether to enable ssl
SPRING_MVC_LOG_DETAILS	BOOLEAN	-	true	M	-
CONFIGURATION_ROOT_LOG_LEVEL	STRING	-	INFO	O	Set Default Log Level for Spring Application
CONFIGURATION_WEB_LOG_LEVEL	STRING	-	INFO	O	Set Default Log level for Spring Web
CONFIGURATION_DB_SHOW_SQL	BOOLEAN	-	false	O	Whether to enable logging of SQL statements.
CONFIGURATION_DB_SQL_COMMENTS	BOOLEAN	-	false	O	If turned on, Hibernate will generate comments inside the SQL, for easier debugging, defaults to `false` .
CONFIGURATION_DB_QUERY_TIMEOUT	INTEGER	-	30000	M	sets the length of time to wait for an SQL request to complete.

Table 3-3 (Cont.) Configuration Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
CONFIGURATION_DB_LOCK_TIMEOUT	INTEGER	-	30000	M	sets the length of time to wait on a blocked resource.
DB_DATASOURCE_URL	STRING	-	jdbc:mysql://<DB_IP>:3306// configuration_schema	M	Ip address of Database service
DB_DATASOURCE_USER	STRING	-	******	M	Database User name from secret
DB_DATASOURCE_PASSWORD	STRING	-	******	M	Database User Password from secret

Consumer Adapter Service Parameters

Table 3-4 Consumer Adapter Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)
OCNADD_CONSUMER_ADAPTER_PORT	Integer	9182	M
OCNADD_CONSUMER_ADAPTER_PROTOCOL	String	http	M
OCNADD_CONSUMER_ADAPTER_SERVICE_NAME	String	ocnadd-consumeradapter	M
CONSUMER_ADAPTER_SELF_URL	String	${protocol}://${spring.application.name}:${server.port}/ ocnadd-consumeradapter/v1/ notifications	O
EGRESS_GATEWAY_ENDPOINT	String	http:// 10.75.245.109:32122/	M
THIRD_PARTY_CONSUMER-ENDPOINT	String	http:// 10.75.245.109:30513/ ocdd-consumer/v1/ messages	M

Table 3-4 (Cont.) Consumer Adapter Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)
BOOTSTRAP_SERVER	String	10.75.245.109:30511	M
OCNADD_CONSUMER_ADAPTER_TOPIC	String	FILTER	M
OCNADD_CONSUMER_ADAPTER_CONFIGURATION_NAME	String	first1	M
OCNADD_CONSUMER_ADAPTER_FETCH_MIN_BYTES	String	1024	M
OCNADD_CONSUMER_ADAPTER_FETCH_MAX_WAIT_MS	String	50	M
OCNADD_CONSUMER_ADAPTER_FETCH_MAX_BYTES	String	52428800	M
OCNADD_CONSUMER_ADAPTER_MAX_PARTITION_FETCH_BYTES	String	250000000	M
OCNADD_CONSUMER_ADAPTER_AUTO_OFFSET_RESET	String	earliest	M
OCNADD_CONSUMER_ADAPTER_MAX_POLL_RECORDS	String	10000	M
OCNADD_CONSUMER_ADAPTER_AUTO_COMMIT	String	false	M
OCNADD_CONFIGURATION_SERVICE_API_ROOT	String	http://localhost:9192/ocnadd-configuration/v1/subscription	M
OCNADD_CONSUMER_ADAPTER_ENABLE_FIRE_AND_IGNORE	String	true	M
OCNADD_CONSUMER_ADAPTER_PAGE_NUMBER	String	10	M
OCNADD_CONSUMER_ADAPTER_PAGE_SIZE	String	100	M
EGRESS_RETRY_INTERVAL	Integer	10000	M
INITIAL_DATA_DELIVERY_DELAY	Integer	0	-
SERVICEID_RETRY	Integer	20	-

Table 3-4 (Cont.) Consumer Adapter Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)
OCNADD_CONSUMER_ADAPTER_CONFIG_RETRY_COUNT	Integer	0	-
OCNADD_CONSUMER_ADAPTER_CONFIG_RETRY_DELAY	Integer	10	-
CONNECTION_THRESHOLD	Integer	5	-
PUBLISHER_THRESHOLD	Integer	1000	-
THIRD_PARTY_RETRY_INTERVAL_MS	Integer	10000	-
OCNADD_CONSUMER_ADAPTER_MAX_PARTITION	Integer	15	-
KAFKA_SESSION_TIMEOUT_MS	Integer	5000	-
CONFIGURATION_HTTP2_ENABLED	String	true	-
CONFIGURATION_SSL_ENABLED	String	false	-
OCNADD_SSL_KEY_TYPE	String	PKCS12	-
OCNADD_SSL_KEY_STORE	-	-	-
OCNADD_SSL_KEY_STORE_PASSWORD	String	secret	-
OCNADD_SSL_TRUST_TYPE	String	PKCS12	-
OCNADD_SSL_TRUST_STORE	-	-	-
OCNADD_SSL_KEY_TRUST_PASSWORD	String	secret	M
OCNADD_TRUST_CLIENT_KEY_STORE	-	-	M
OCNADD_TRUST_CLIENT_TRUST_STORE	String	src/main/resources/keystore/clientTrustStore.p12	M
OCNADD_TRUST_CLIENT_PASSWORD	String	secret	M
OCNADD_TRUST_CLIENT_KEY_TYPE	String	PKCS12	M
OCNADD_TRUST_KEYSTORE	String	true	M
OCNADD_TRUST_PASSWORD	String	secret	M

Table 3-4 (Cont.) Consumer Adapter Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)
OCNADD_CONSUMER_ADAPTER_TARGET_CONSUMER_NAME	Integer	100	M
ADAPTER_TIMEOUT	Integer	10	M
ADAPTER_CHANNEL_TIMEOUT	Integer	60	M
ADAPTER_CONN_POOL_MAX_CNT	Integer	400	M
ADAPTER_CONN_POOL_MAX_IDLE	Integer	90	M
ADAPTER_CONN_POOL_MAX_LIFE	Integer	3600	M
ADAPTER_CONN_POOL_MAX_PENDING	Integer	50	M
ADAPTER_PENDING_CONN	Integer	50	M
ADAPTER_KEEPALIVE_IDLE	Integer	60	M
ADAPTER_KEEPALIVE_INT	Integer	60	M
ADAPTER_KEEPALIVE_CNT:	Integer	10	M
ADAPTER_PUBLISHER_TIMEOUT	Integer	12	M
OCWEBCLIENT_TIMEOUT	Integer	60	O
OCWBCLIENT_CHANNEL_TIMEOUT	Integer	60	O
OCWEBCLIENT_SSL_HANDSHAKE_TIMEOUT	Integer	30	O
OCWEBCLIENT_SSL_FLUSH_TIMEOUT	Integer	10	O
OCWEBCLIENT_SSL_READ_TIMEOUT	Integer	10	O
OCWEBCLIENT_CONN_POOL_MAX_CNT	Integer	50	O
OCWEBCLIENT_CONN_POOL_MAX_IDLE	Integer	60	O
OCWEBCLIENT_CONN_POOL_MAX_LIFE	Integer	120	O
OCWEBCLIENT_CONN_POOL_MAX_PENDING	Integer	120	O
OCWEBCLIENT_KEEPALIVE_IDLE	Integer	60	O

Table 3-4 (Cont.) Consumer Adapter Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)
OCWEBCLIENT_KEEPALIVE_INT	Integer	60	O
OCWEBCLIENT_KEEPALIVE_CNT	Integer	10	O
OCNADD_CONSUMER_ADAPTER_MAX_REPLICA	Integer	1	M
OCNADD_ALARM_SVC_URL	String	-	M
SPRING_MVC_LOG_DETAILS	String	true	M
ADAPTER_LOG_LEVEL	String	INFO	M
OCNADD_CONSUMER_ADAPTER_LOG_LEVEL	String	INFO	M
OCNADD_CONFIGURATION_SERVICE_LOG_FILENAME	String	configuration-service.log	M
OCNADD_CONSUMER_ADAPTER_HEALTH_RETRY_COUNT	Integer	3	M
OCNADD_CONSUMER_ADAPTER_HEALTH_RETRY_DELAY	Integer	10	M
OCNADD_CONSUMER_ADAPTER_HEALTH_ENDPOINT	Integer	-	M
OCNADD_CONSUMER_ADAPTER_HEALTH_HB_TIMER	Integer	120000	M
OCNADD_CONSUMER_ADAPTER_HEALTH_SVC_TYPE	String	CONSUMER_ADAPTER	M
OCNADD_CONSUMER_ADAPTER_HEALTH_SELF_ENDPOINT	String	https://ocnadd-consumeradapter:9182/healthmonitoring	M

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
DD_HEALTH_PROFILE_ACTIVE	STRING	Prod	M	Service profile
ALARM_SERVICE_URL	STRING	http://ocnaddalarm:9099/ocnadd-alarm/v1	M	Alarm Application URL
ALARM_FOR_DEREGISTEREDSERVICE	BOOLEAN	True	M	Alarm on deregistered service
HEALTH_MONITORING_TIMER	INTEGER	5000	M	Timer to check Health of integrated services
HEALTH_METRICS_SCHEDULED	BOOLEAN	FALSE	M	Scheduler for metrics
HEALTH_METRICS_TIMER	INTEGER	120000	M	Timer for health metrics
HEALTH_PURGE_TIME_HR	INTEGER	24	M	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
HEALTH_LOG_REQUEST_DETAILS	BOOLEAN	TRUE	O	Set detailed log for spring mvc
HEALTH_APPLICATION_LOG_LEVEL	STRING	INFO	O	Set application logger level
DATA Source Properties				

Table 3-5 (Cont.) Health Monitoring Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
DB_URL	STRING	jdbc:mysql://<DB_HOST>:<DB_PORT>/<HEALTH SCHEMA>	M	Ip address of Database service
MYSQL_USER	STRING	<Secret>	M	Database User name
MYSQL_PASSWORD	STRING	<Secret>	M	Database User Password
SHOW_SQL	BOOLEAN	FALSE	O	Whether to enable logging of SQL statements.
LOGGING_LEVEL_SQL	STRING	INFO	O	SQL Logs
HTTP2 Properties				
HEALTH_SSL_ENABLED	BOOLEAN	FALSE	M	whether to enable ssl

Table 3-6 Alarm Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_ALARM_PORT	INTEGER	-	9099	M	Server HTTP port
OCNADD_ALARM_PROFILE	STRING	dev,prod	prod	M	Alarm Service Profile (choose Based on Environment)
Alarm Purging Properties					
ALARM_PURGE_DAYS	INTEGER	>1	30	M	Delete the Alarms which are terminated more than this no of days
ALARM_SCHEDULED_CRON	STRING	-	0 0 0 * * 0	M	cron expression for scheduled deletion of alarms. Default: once a week or can also mention macros i.e(@yearly,@monthly,@weekly,@daily,@hourly,@midnight)

Table 3-6 (Cont.) Alarm Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
Logging Properties					
ALARM_WEB_LOG_LEVEL	STRING	-	INFO	O	Set Default Log level for Spring Web
ALARM_ROOT_LOG_LEVEL	STRING	-	INFO	O	Set Default Log Level for Spring Application
ALARM_LOGGING_TYPE	STRING	STDOUT/ LOGJSON	STDOUT	O	Logging Type Standard Out / JSON
JPA Properties					
ALARM_DB_SHOW_SQL	BOOLEAN	-	false	O	Whether to enable logging of SQL statements.
ALARM_DB_SQL_COMMENTS	BOOLEAN	-	true	O	If turned on, Hibernate will generate comments inside the SQL, for easier debugging, defaults to `false` .
ALARM_DB_QUERY_TIMEOUT	INTEGER	-	30000	M	sets the length of time to wait for an SQL request to complete.
ALARM_DB_LOCK_TIMEOUT	INTEGER	-	30000	M	sets the length of time to wait on a blocked resource.
DATA Source Properties					
MYSQL_URL	STRING	-	localhost	M	Ip address of Database service
MYSQL_PORT	INTEGER	-	3306	M	Port for Database service
ALARM_SCHEMA_NAME	STRING	-	alarm_schema	M	Name of Alarm Schema
DB_DATASOURCE_USER	STRING	-	ocdd	M	Database User name

Table 3-6 (Cont.) Alarm Service Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
DB_DATASOURCE_PASSWORD	STRING	-	ocdd	M	Database User Password
HTTP2 Properties					
ALARM_SSL_ENABLED	BOOLEAN	-	false	M	whether to enable ssl

Admin Service Parameters

Table 3-7 Admin Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
ADMINSVC_HTTP2_ENABLED	BOOLEAN	true	M	Whether to enable HTTP2
ADMINSVC_SSL_ENABLED	BOOLEAN	false	M	Whether to enable ssl
OCNADD_SSL_KEY_TYPE	STRING	PKCS12	M	-
OCNADD_SSL_KEY_STORE	STRING	-	M	-
OCNADD_SSL_KEY_STORE_PASSWORD	STRING	-	M	-
OCNADD_SSL_TRUST_TYPE	STRING	PKCS12	M	-
OCNADD_SSL_TRUST_STORE	STRING	-	M	-
OCNADD_SSL_KEY_TRUST_PASSWORD	STRING	-	M	-
OCNADD_TRUST_CLIENT_KEY_STORE	STRING	-	M	-
OCNADD_TRUST_CLIENT_TRUST_STORE	STRING	-	M	-
OCNADD_TRUST_CLIENT_PASSWORD	STRING	-	M	-
OCNADD_TRUST_CLIENT_KEY_TYPE	STRING	PKCS12	M	-
OCNADD_TRUST_KEYSTORE	STRING	-	M	-

Table 3-7 (Cont.) Admin Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_TRUST_PASSWORD	STRING	-	M	-
OCWebClient				
OCWEBCLIENT_TIMEOUT	INTEGER	60	M	-
OCWBCLIENT_CHANNEL_TIMEOUT	INTEGER	60	M	-
OCWEBCLIENT_SSL_HANDSHAKE_TIMEOUT	INTEGER	30	M	-
OCWEBCLIENT_SSL_FLUSH_TIMEOUT	INTEGER	10	M	-
OCWEBCLIENT_SSL_READ_TIMEOUT	INTEGER	10	M	-
OCWEBCLIENT_CONN_POOL_MAX_CNT	INTEGER	50	M	-
OCWEBCLIENT_CONN_POOL_MAX_IDLE	INTEGER	60	M	-
OCWEBCLIENT_CONN_POOL_MAX_LIFE	INTEGER	120	M	-
OCWEBCLIENT_CONN_POOL_MAX_PENDING	INTEGER	120	M	-
OCWEBCLIENT_KEEPALIVE_IDLE	INTEGER	60	M	-
OCWEBCLIENT_KEEPALIVE_INT	INTEGER	60	M	-
OCWEBCLIENT_KEEPALIVE_CNT	INTEGER	10	M	-
Configurable Kafka parameters and their default values				
OCNADD_KAFKA_BOOTSTRAP_SERVER	URL	-	M	-
Configurable Kubernetes Parameters				
OCNADD_NAMESPACE	STRING	-	M	-
OCNADD_NODE_NAME	STRING	-	M	-
OCNADD_NODE_KEY	STRING	-	M	-

Table 3-7 (Cont.) Admin Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_NODE_OPERATOR	STRING	In	M	-
OCNADD_STORAGE_CLASS	STRING	standard	M	-
Consumer Adapter Properties				
OCNADD_CONSUMER_GROUP_NAME	STRING	consumer	M	-
OCNADD_EGRESS_GATEWAY_ENDPOINT	URL	-	M	-
OCNADD_THIRD_PARTY_CONSUMER_ENDPOINT	URL	-	M	-
OCNADD_CONSUMER_ADAPTER_IMAGE_NAME	STRING	-	M	-
OCNADD_CONSUMER_GROUP_MAX_REPLICAS	INTEGER	1	M	-
OCNADD_SERVICE_ACCOUNT	STRING	-	M	-
OCNADD_CONSUMER_TOPIC_NAME	STRING	FILTER	M	-
CONFIGURATION_SERVICE_API_ROOT	URL	-	M	-
OCNADD_CONSUMER_PORT	INTEGER	-	M	-
OCNADD_CONSUMER_PROTOCOL	STRING	-	M	-
OCNADD_CONSUMER_ADAPTER_SERVICE_NAME	STRING	-	M	-
CONSUMER_ADAPTER_SELF_URL	URL	ocnaddconsumeradapter	M	-
OCNADD_ADMINSVC_HEALTH_RETRY_COUNT	INTEGER	3	M	-
OCNADD_ADMINSVC_HEALTH_RETRY_DELAY	INTEGER	10	M	-
OCNADD_ADMINSVC_HEALTH_ENDPOINT	URL	-	M	-

Table 3-7 (Cont.) Admin Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_ADMINSVC_HEALTH_HB_TIMER	INTEGER	120000	M	-
OCNADD_ADMINSVC_HEALTH_SVC_TYPE	STRING	-	M	-
OCNADD_ADMINSVC_HEALTH_SELF_ENDPOINT	URL	-	M	-
ALARM_SERVICE_URL	URL	-	M	-
Egress Gateway Parameter				
OCNADD_NAMESPACE	STRING	<namespace>	M	Namespace to spawn egw
OCNADD_CONSUMER_EGW_TLS_CN	STRING	egw	M	Suffix for egw configuration
OCNADD_CONSUMER_EGW_IMAGE_PULLPOLICY	STRING	IfNotPresent	M	Image Pull Policy for deployment
OCNADD_CONSUMER_EGW_IMAGE_PULLSECRET	STRING	<secret>	M	Image pull secret to pull image from Repo
OCNADD_CONSUMER_EGW_INIT_IMAGE	STRING	<Init Image>	O	Init Image
OCNADD_CONSUMER_EGW_IMAGE_NAME	STRING	<EGW Image>	M	Image to spawn Egw
OCNADD_CONSUMER_EGW_CPU_LIMIT	STRING	4	M	CPU Limit for EGW
OCNADD_CONSUMER_EGW_CPU_REQUEST	STRING	4	M	CPU Min requirement for EGW
OCNADD_CONSUMER_EGW_MEMORY_LIMIT	STRING	6Gi	M	Memory limit for EGW
OCNADD_CONSUMER_EGW_MEMORY_REQUEST	STRING	6Gi	M	Memory Min Requirement for EGW
OCNADD_CONSUMER_EGW_MAX_REPLICA	INTEGER	16	M	Max Replicas for EGW
OCNADD_CONSUMER_EGW_MIN_REPLICA	INTEGER	1	M	Min Replicas for EGW

Table 3-7 (Cont.) Admin Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_CONSUMER_EGW_MAX_SURGE	INTEGER	1	M	Amount of pods more than the desired number of Pods
OCNADD_CONSUMER_EGW_MAX_UNAVAILABLE	INTEGER	0	M	Amount of pods that can be unavailable during the update process
OCNADD_CONSUMER_EGW_BOOTUP_MINREADYSEC	INTEGER	0	M	The bootup time of your application, Kubernetes waits specific time till the next pod creation to serve traffic
OCNADD_CONSUMER_EGW_AVG_CPU_UTIL	INTEGER	70	M	Average CPU Utilization
OCNADD_CONSUMER_EGW_POD_SCALE_DOWN	INTEGER	1	M	No. of pod scale down
OCNADD_CONSUMER_EGW_POD_SCALE_DOWN_PERIOD	INTEGER	30	M	Scale down in seconds
OCNADD_CONSUMER_EGW_POD_SCALE_DOWN_STABLE	INTEGER	60	M	Stabilization period in seconds post which scale down starts
OCNADD_DEPLOYMENT_TERMINATION_GRACE	INTEGER	30	M	Grace Termination of Pods in seconds
OCNADD_CONSUMER_EGW_PROFILE_ACTIVE	STRING	prod	M	Active profile
OCNADD_CONSUMER_EGW_HTTP_CLIENT_POOL_MAX_ACQUIRE	INTEGER	60000	O	Max time to acquire connection pool in ms
OCNADD_CONSUMER_EGW_HTTP_CLIENT_POOL_MAX_CONN	INTEGER	300	O	Max number of connections per pool
OCNADD_CONSUMER_EGW_HTTP_CLIENT_POOL_MAX_IDLE	INTEGER	40	O	Max number of connection in a pool to be idle

Table 3-7 (Cont.) Admin Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_CONSUMER_EGW_HTTP_CLIENT_CONNECT_TIMEOUT	INTEGER	30000	O	Connection timeout parameter
OCNADD_CONSUMER_EGW_HTTP_CLIENT_RESPONSE_TIMEOUT	STRING	8s	O	Response timeout parameter
OCNADD_EGW_EMPHEMERAL_STORAGE_LIMIT	STRING	400Mi	M	Ephemeral storage Max
OCNADD_EGW_EMPHEMERAL_STORAGE_REQUEST	STRING	200Mi	M	Ephemeral storage Min
ENABLE_EGW_COUNTER_METRICS	BOOLEAN	true	M	Counter Metrics Enable
ENABLE_EGW_LATENCY_METRICS	BOOLEAN	true	M	Latency Metrics Enable
OCNADD_EGW_LIVENESS_DELAY	INTEGER	60	O	Pod liveliness delay
OCNADD_EGW_LIVENESS_PERIOD	INTEGER	15	O	Pod liveliness Timeperiod
OCNADD_EGW_LIVENESS_FAILURE	INTEGER	5	O	Liveliness failure in seconds
OCNADD_EGW_LIVENESS_TIMEOUT	INTEGER	20	O	Pod Liveliness Timeout
EGW_MIN_EXPECTED_VALUE_LATENCY_MS	INTEGER	50	M	Min expected latency for metrics
EGW_MAX_EXPECTED_VALUE_LATENCY_MS	INTEGER	50	M	Max expected latency for metrics
EGW_MIN_EXPECTED_VALUE_ETE_LATENCY_MS	INTEGER	50	M	Min expected latency for metrics
EGW_MAX_EXPECTED_VALUE_ETE_LATENCY_MS	INTEGER	50	M	Max expected latency for metrics
Logging				
SPRING_MVC_LOG_LEVEL	STRING	INFO	M	Mvc logs

Table 3-7 (Cont.) Admin Service Parameters

Parameter Name	Data Type	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
SPRING_MVC_LOG_TYPE	STRING	STDOUT	M	STDOUT output in pod logs

Kafka Configuration Parameters

Table 3-8 Kafka Configuration Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
chart.name	String		ocdd-kafka	M	Defines the name of the kafka chart
broker.id	Int		-1	M	Gives the broker an int as an identifier. -1 will let the cluster choose an unqiue identifier automatically
delete.topic.enable	boolean		true	C	Enables the feature of deleting a topic
group.initial.rebalance.delay.ms	Int		20		The amount of time the group coordinator will wait for more consumers to join a new group before performing the first rebalance.
log.dir	String		/tmp/kafka-logs	M	The path to store the kafka logs
zookeeper.ssl.client.enable	boolean		true	M	To enable or disable ssl in the connection between the kafka broker and the zookeeper
zookeeper.ssl.protocol	String		TLSv1.2	M	SSL protocol for connection between kafka broker and zookeeper

Table 3-8 (Cont.) Kafka Configuration Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
zookeeper.ssl.truststore.location	String			M	Path to where the zookeeper truststore is mounted.
zookeeper.ssl.truststore.password	String			M	Password for the zookeeper truststore
zookeeper.ssl.keystore.location	String			M	Path to where the zookeeper keystore is mounted.
zookeeper.ssl.keystore.password	String			M	Password for the zookeeper keystore
security.inter.broker.protocol	String		PLAINTEXT	M	Protocol for inter broker communication
auto.create.topics.enable	boolean		false		When set to true, when applications attempt to produce, consume, or fetch metadata for a non-existent topic, Kafka will automatically create the topic with the default replication factor and number of partitions.
super.users	String				Super users are allowed to perform any operation on any resource in a Kafka cluster
kafkaBroker.name	String				Name of the kafka broker
minReplicas	Int		1		The minumum number of replicas that must be available at all times

Table 3-8 (Cont.) Kafka Configuration Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
maxReplicas	Int		2		The maximum number of replicas allowed for the pod
replicas	Int		1		The number of replicas that you want to be available for the pod
target.averageCpuUtilPercentage	Int		50		The target average CPU utilization percentage
target.memoryUtilPercentage	Int		80		The target average memory utilization percentage
container.port			9098		The port which at which the kafka container is exposed for connection
container.saslPort			9099		The port which at which the kafka container is exposed for SASL connection
kafkaBroker.container.image	String				The url of the kafka image that is to be pulled
kafkaBroker.container.pullPolicy	String		IfNotPresent		The image pull policy for the container.image
sharedVolumeName	String				The name of the volume that is mounted
mountPath	String				The path within the container where the volume will be mounted
resource.requests.cpu	Int				The requested number of CPU cores

Table 3-8 (Cont.) Kafka Configuration Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
resource.requests.memory	String				The requested size of memory
resource.limits.cpu					The maximum limit for the number of CPUs used for the container
resource.limits.memory	Int				The maximum limit for the size of the memory used for the container
partitions	Int				The number of partitions in the broker
replicationFactor	Int				The number of copies of the partitions that are to be maintained
kafkaSvcName	String				The name of the kafka service name
service.type	String				The type of the kafka service
serivce.targetPort					The port at which the service will send requests to
service.saslSslTargetPort					The port at which the service will send SASL requests to
delete.topic.enable	boolean		true	C	Enables the feature of deleting a topic
group.initial.rebalance.delay.ms	Int		20		The amount of time the group coordinator will wait for more consumers to join a new group before performing the first rebalance.
log.dir	String	M	/tmp/kafka-logs		The path to store the kafka logs

Table 3-8 (Cont.) Kafka Configuration Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
zookeeper.ssl.client.enable	boolean	M	true		To enable or disable ssl in the connection between the kafka broker and the zookeeper
zookeeper.ssl.protocol	String	M	TLSv1.2		SSL protocol for connection between kafka broker and zookeeper
zookeeper.ssl.truststore.location	String	M			Path to where the zookeeper truststore is mounted.
zookeeper.ssl.truststore.password	String	M			Password for the zookeeper truststore
zookeeper.ssl.keystore.location	String	M			Path to where the zookeeper keystore is mounted.
zookeeper.ssl.keystore.password	String	M			Password for the zookeeper keystore
ssl.truststore.location					Path to where the truststore for connection between broker and clients is mounted.
ssl.truststore.password					Password of the truststore for conection between broker and clients.
ssl.keystore.location					Path to where the keystore for coneection between broker and clients is mounted.
ssl.keystore.password					Password of the keystore for conection between broker and clients.

Table 3-8 (Cont.) Kafka Configuration Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
num.io.threads			8		Number of threads that pick up requests from the request queue to process them
num.network.threads					Network threads handle requests to the Kafka cluster, such as produce and fetch requests from client applications
socket.send.buffer.bytes					Denotes the value of the buffer size for sending messages
socket.receive.buffer.bytes					Denotes the value of the buffer size for receiving messages
num.recovery.threads.per.data.dir					This parameter is used to specify the number of threads used for log loading at startup and flushing at shutdown.

Backend Router Parameters

Table 3-9 Backend Router Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_BKNROUTER_APPLICATION_NAME	String		ocnaddbackendrouter	M	Application Name

Table 3-9 (Cont.) Backend Router Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
OCNADD_BKNROUTER_PORT	Int		8988	M	Server port
OCNADD_BKNROUTER_ROOT_LOG_LEVEL	String		INFO		Root log level configuration (INFO,DEBUG,ERROR)
OCNADD_BKNROUTER_COM_LOG_LEVEL	String		INFO		Log level configuration (INFO,DEBUG,ERROR)
OCNADD_BKNROUTER_LOG_LEVEL	String		INFO		Log level configuration (INFO,DEBUG,ERROR)
OCNADD_BKNROUTER_LOG_FILENAME	String		var/occne/ocnaddbackendrouter.dir		Log file directory update.
OCNADD_BKNROUTER_CONFIG_REDIRECTURI	String		http://ocnaddconfiguration:12590	M	Configuration service URI
OCNADD_BKNROUTER_HEALTHSVC_REDIRECTURI	String		http://ocnaddhealthmonitoring:12591	M	Health service URI
OCNADD_BKNROUTER_AlARM_REDIRECTURI	String		http://ocnaddalarm:9099	M	Alarm Service URI
OCNADD_BKNROUTER_CONFIG_PATH_PATERN	String		/ocnadd-configuration/**		
OCNADD_BKNROUTER_HEALTH_PATH_PATERN	String		/ocnaddapi/ocnadd-health/**		
WEBCLIENT_MAX_BUFFER_SIZE	String		17777216		

4

Upgrading OCNADD

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

Supported Upgrade Path

The following table lists the supported upgrade paths for OCNADD.

Table 4-1 Supported Upgrade Paths

Source OCNADD release	Target OCNADD release
22.0.0	22.0.0

Preupgrade Tasks

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

Note:

- While performing an upgrade, you must align the `ocnadd/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/values.yaml`
   - `ocnadd/ocdd-db-resource.sql`
   - `ocnadd/templates/ocnadd-secret-hook.yaml`
   - Update the `pvcClaimSize` parameter of all the Kafka brokers `ocnadd/charts/ocnaddkafka/values.yaml`.

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 and Egress gateway 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 the Consumer Adapter and Egress gateway services, see the "Parameter Updates for OCNADD Microservices" section of 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>
```

   In case of any failure, follow the steps mentioned in the *Oracle Communications Network Analytics Data Director Troubleshooting Guide*.

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.

Supported Rollback Path

The following table lists the supported rollback paths for OCNADD:

Table 5-1 Supported Rollback Path

Source Release	Traget Release
22.0.0	22.0.0

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.

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

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:

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 -n <namespace> <secretname> <secretname>
   ```

   Note:

   To delete multiple secrets simultaneously, run the following command:

   ```
   kubetl delete secret kafka-broker1-secret kafka-broker2-
   secret kafka-broker3-secret kafka-broker3-secret jaas-
   secret ocnaddadminservice-secret -n ocnadd
   ```

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

   ```
   kubectl delete configmap allfiles-configmap -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 broker1-pvc-kafka-broker1-0 broker2-pvc-kafka-
      broker2-0 broker3-pvc-kafka-broker3-0 kafka-broker-security-
      zookeeper-0 kafka-broker-security-zookeeper-1 kafka-broker-
      security-zookeeper-2 -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>
   ```

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 Kubernete objects:

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

   - To delete all the configmaps:

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

     Caution:

     The commands delete 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.