

# Oracle® Communications

## Network Analytics Suite Security Guide



Release 25.2.100  
G41332-01  
October 2025

ORACLE®

Copyright © 2022, 2025, Oracle and/or its affiliates.

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

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

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

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

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

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

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

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

# Contents

1	Introduction	
1.1	References	1
2	Overview	
2.1	OCNADD Overview	1
3	Secure Development Practices	
3.1	Vulnerability Handling	1
3.2	Trust Model for OCNADD	1
3.2.1	Context Diagram	1
3.2.2	Key Trust Boundaries	4
3.2.3	External Data Flows	5
4	Implementing Security Recommendations and Guidelines	
4.1	Implementing Security Recommendations and Guidelines for OCNADD	1
4.1.1	TLS Configuration	1
4.1.2	Network Policies	22
4.1.3	Kafka Security Configuration	25
4.1.4	MySQL Configuration	26
4.1.5	Certificate Creation for NFs and Third Party Consumers	26
4.1.6	Renewing OCNADD Certificates	27
4.1.7	Signing OCI Container Registry Images for Security	30
4.1.8	Cleaning Up Sensitive Information After Installation	31
4.1.9	Druid User Management	31
A	Network Port Flows	

# Preface

- [Documentation Accessibility](#)
- [Diversity and Inclusion](#)
- [Conventions](#)

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

### Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

## Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

## Conventions

The following text conventions are used in this document:

Convention	Meaning
<b>boldface</b>	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

# My Oracle Support

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

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

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

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

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

# Acronyms

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

**Table    Acronyms**

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

**Table (Cont.) Acronyms**

Acronym	Description
SEPP	Security Edge Protection Proxy
SMF	Session Management Function
SNMP	Simple Network Management Protocol
SVC	Services
SUPI	Subscription Permanent Identifier
UDM	Unified Data Management
UE	User Equipment
URI	Uniform Resource Identifier

# What's New in This Guide

This section lists the documentation updates for Network Analytics Suite release 25.2.1xx.

## Release 25.2.100 - G41332-01, October 2025

### OCNADD

- Updated procedures for "Enabling/Disabling intraTLS in Data Director" and "Certificates and Secret Generation" in the [TLS Configuration](#) section
  - Removed the steps and instances of Admin service as it not supported in this release.
  - Added a section "Certificates for Apache Druid".
- Added a note about allowing connection from Druid IPs to Kafka when Druid is configured to ingest from Kafka in the [Network Policies](#) section.
- Added the procedures for [Druid User Management](#).



# 1

## Introduction

The Security Guide provides an overview of the security information applicable to the Oracle Communications Network Analytics Suite of products.

This document contains recommendations and step-by-step instructions to assist the customer in hardening the Network Analytics Suite system, it also provides a simplified trust model for the system.

### 1.1 References

The following references provide additional background on product operations and support:

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

# 2

## Overview

### 2.1 OCNADD Overview

#### Deployment Environment

The OCNADD can be deployed in a variety of possible configurations and environments:

**Table 2-1 Deployment Environment**

Type	Host	CNE	Description
Cloud	Customer Cloud	OCCNE	The OCCNE is deployed on the cloud provided by the customer.  The OCNADD is deployed on the Kubernetes cluster managed by OCCNE.

#### OCNADD Services

**Table 2-2 OCNADD Services**

Services	Description
Admin Service	The Admin Service administers the Kafka Cluster and it provides an interface to create and delete a Consumer Adapter deployment.
Aggregation Service	The Aggregation Service collects and aggregates network traffic from multiple NFs (For example: SCP 1 and 2 and/or SEPP, NRF, 3rd party NF, and so on).
Alarm Service	The OCNADD Alarm service is used to store the alarms generated by other OCNADD services.
UI Router Service	The OCNADD UI Router service acts as an interface between the CNC Console and the OCNADD services.
Configuration Service	The Configuration Service is used by the OCNADD UI service to configure 3rd party message feed.
Consumer Adapter Service	The Consumer Adapter Service manages the third-party specific connection parameters and provide secure message transport towards the third-party consumer application.
Health Monitoring Service	The Health Monitoring Service monitors the health of each OCNADD service and provides alerts related to the overall health of the OCNADD.

Table 2-2 (Cont.) OCNADD Services

Services	Descriptio
Kafka Broker Service	The Kafka Broker Service stores the incoming network traffic from different NFs. It also stores the intermediate processed data from different microservices.
GUI service	The GUI service is used as an interface between the CNC Console and OCNADD that provides the user interface to configure the Data Feeds.
Filter service	The filter service filters the feed data as per configuration. The filtered feed is then used for preparing xDR reports or consumed by External Kafka Feeds.
Correlation service	The correlation service creates xDR reports for the feed data as per configuration. The xDR reports are then consumed by External Kafka Feeds.
Redundancy Agent	The Redundancy agent is responsible for syncing configurations and maintaining two-site redudancy between mated worker group pair.
Export Service	The export service is responsible for exporting the XDRs to the third party server using the secure file transfer service (SFTP) based on the export configuration.
Ingress Adapter	The ingress adapter service is responsible for providing an interface to ingest data into the OCNADD using REST from the non Oracle 5G NFs.
Storage Adapter	The service is responsible for storing the generated xDRs by the correlation service in the extended storage database.

# 3

## Secure Development Practices

Given below are the practices followed for a secure development environment:

### 3.1 Vulnerability Handling

For details about the vulnerability handling, refer [Oracle Critical Patch Update Program](#). The primary mechanism to backport fixes for security vulnerabilities in Oracle products is quarterly Critical Patch Update (CPU) program.

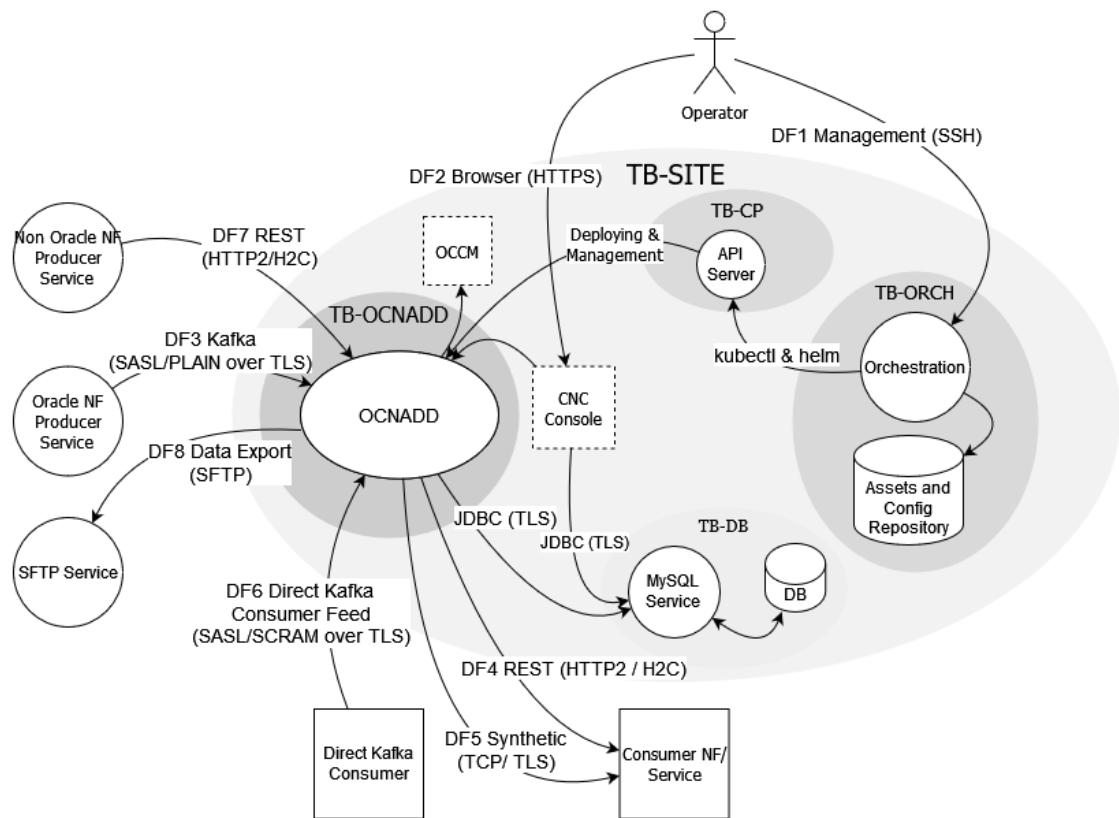
In general, OCNADD Software is on a quarterly release cycle, with each release providing feature updates and fixes and updates to relevant third party software. These quarterly releases provide cumulative patch updates.

### 3.2 Trust Model for OCNADD

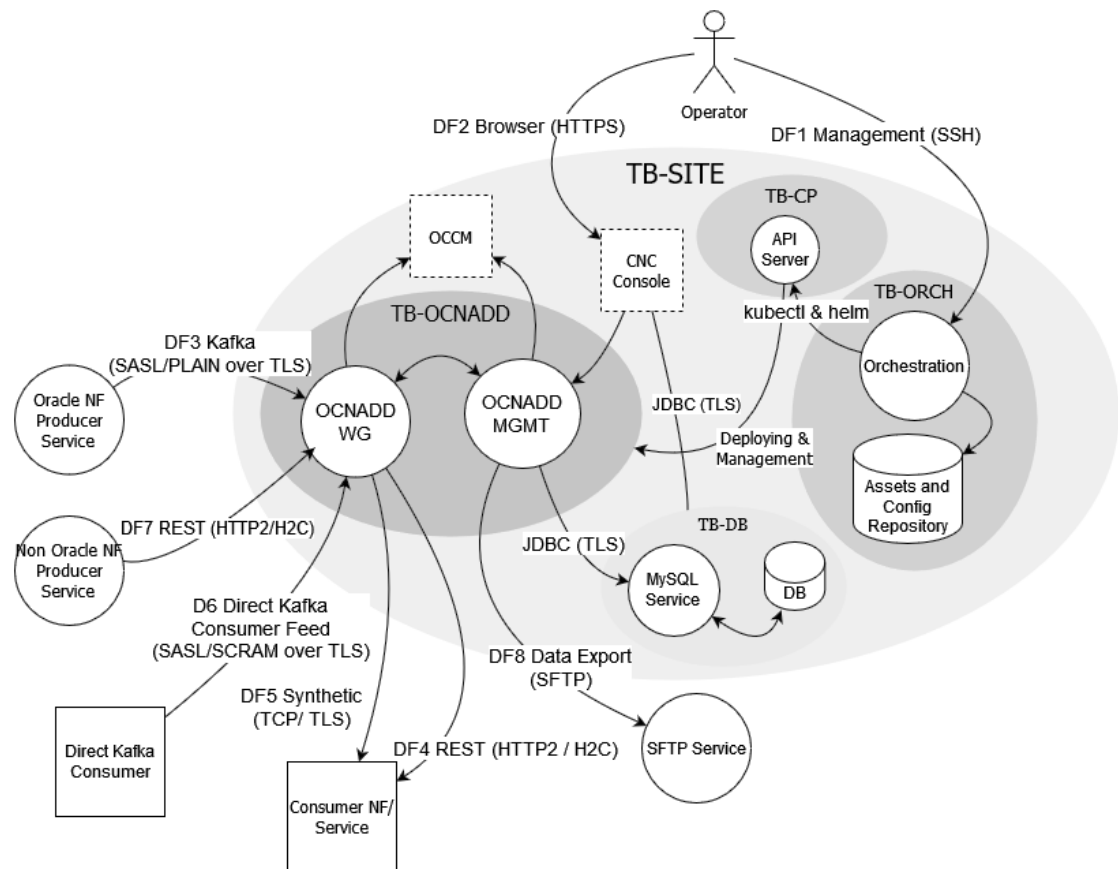
The following Trust Model depicts the reference trust model (regardless of the target environment). The model describes the critical access points and controls site deployment.

#### 3.2.1 Context Diagram

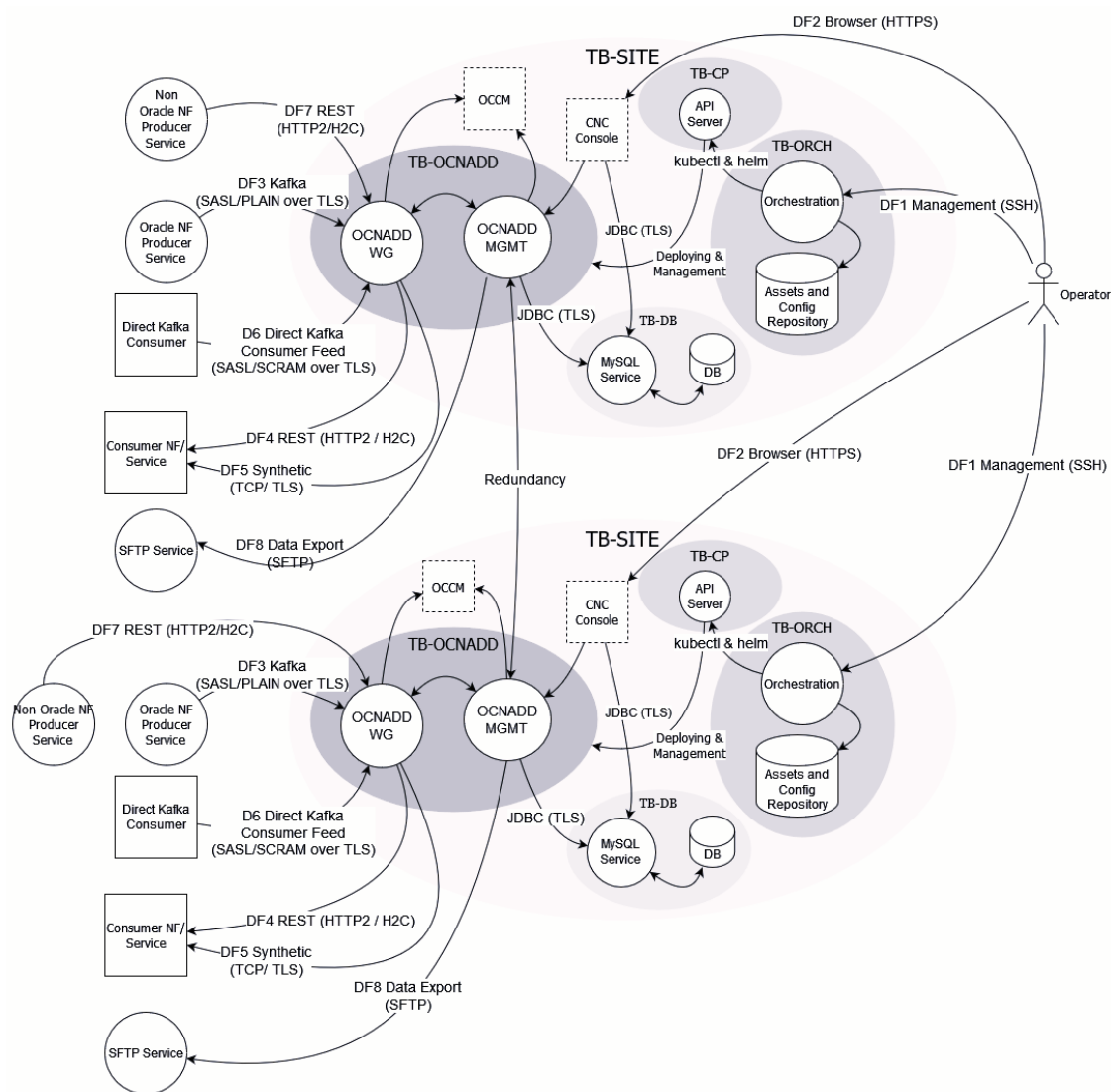
**Non-Centralized Deployment**



### Centralized Deployment



## Two-Site Redundancy



### 3.2.2 Key Trust Boundaries

Following are the key trust boundaries:

**Table 3-1 Key Trust Boundaries**

Trust Boundary	Access Control
OCNADD	Kubernetes Namespace for OCNADD where its internal micro-services are deployed.
Site	Where the Kubernetes cluster is deployed.
Control Plane	This trust boundary delineates the control plane elements of the clusters, that is, API Server, kubelet, containerd and etcd. The configuration database (ETCD service) is isolated so that only control plane services can access it.
Database	MySQL service deployed in a separate Kubernetes namespace.

**Table 3-1 (Cont.) Key Trust Boundaries**

Trust Boundary	Access Control
CNE Infra	Namespace containing all the infrastructure related services (like Prometheus). Provided by CNE.
Orchestration	Includes the orchestration server and the Code and Image Repository.

### 3.2.3 External Data Flows

The following table describes external data flows of OCNADD:

**Table 3-2 External Data Flows**

Data Flow	Protocol	Description
DF1: Management	SSH	Operator will login to the orchestration server through SSH for deploying OCNADD and/or managing the OCNADD Kubernetes deployment using helm.
DF2: Browser	HTTPS	Operator uses CNC Console to create, manage feed configuration and monitor OCNADD.  CNC Console is accessed through the browser and the Operator is authenticated with username and password before access is granted.
DF3: Kafka	SASL_SSL	Oracle NFs write the 5G SBI data or messages into the Kafka exposed by OCNADD in the respective topics. OCNADD will then process according to the feed configurations.  The communications are encrypted using TLS and Oracle NF will authenticate themselves to Kafka through SASL/PLAIN (username and password).
DF4: Consumer NF	HTTP2(w/TLS)	OCNADD forwards the message feed to respective consumer NF/s as HTTP2 (over TLS) or H2C (HTTP2 clear text) messages according to the feed configurations.  The traffic segregation is provided using CNLB egress NAD support. The CNLB support has to be enabled in the helm charts for the egress traffic separation in the consumer adapter and corresponding feed configuration should be done. The CNLB route the packets using the layer3/layer4 information from the defined network attachment definition(NAD).



Table 3-2 (Cont.) External Data Flows

Data Flow	Protocol	Description
DF5: Consumer NF (Synthetic Packet)	TCP	OCNADD forwards the Synthetic Packet to respective consumer NF/s as TCP or TCP_SECURED messages based on the feed configuration.
DF6: Direct Kafka Consumer Feed	Kafka (SASL/SCRAM over TLS)	External Kafka consumer can read data directly from authorized Kafka topic. Consumers are authenticated using SASL/SCRAM (SCRAM-256 and/or SCRAM-512). All communications will be encrypted with TLS.
DF7: REST	HTTP2	Non Oracle NF forwards the messages to OCNAD ingress adapter service using HTTP2 (over TLS) or H2C (HTTP2 clear text) according to the ingress feed configuration. The traffic segregation is provided using CNLB ingress NAD support. The CNLB support has to be enabled in the helm charts for the ingress traffic separation in the ingress adapter and corresponding feed configuration should be done. The CNLB route the packets using the layer3/layer4 information from the defined network attachment definition (NAD).
DF8: Data Export	SFTP	OCNADD send the exported records to the external third party server using secure file transport service based on the data export configuration.

# 4

## Implementing Security Recommendations and Guidelines

### 4.1 Implementing Security Recommendations and Guidelines for OCNADD

This section provides specific recommendations and guidelines for OCNADD security.

#### 4.1.1 TLS Configuration

##### External TLS Communication

**DF2: Browser** and **DF3: Kafka communication** is over TLS only. No additional steps apart from certificate creation need to be performed.

**DF4: Consumer NF** communication can be configured to use clear text or TLS through the OCNADD GUI. For more information, see "Configuring OCNADD Using CNC Console" section in the *Oracle Communications Network Analytics Data Director User Guide*.

**DF5: Consumer NF (Synthetic Feed)** communication can be configured to use TCP or TCP\_SECURED through the OCNADD GUI. For more information, see "Configuring OCNADD Using CNC Console" section in the *Oracle Communications Network Analytics Data Director User Guide*.

**DF6: Direct Kafka Consumer Feed** communication is over TLS only. No additional steps apart from certificate creation need to be performed. Refer to [Internal TLS Communication](#) section for more info about certificate creation.

See [Certificate Creation for NFs and Third Party Consumers](#) before creating certificates for NFs, OCNADD, third party consumers, and Kafka Consumers.

##### Internal TLS Communication

The internal communication between the various data director microservices can be with or without TLS.

##### IntraTLS Enabled

The internal communication between OCNADD services should be secured. All OCNADD services must use the SSL certificates, the services requiring certificates are mentioned below:

**Table 4-1 Management and Worker Group Services when IntraTLS is enabled**

Management Group Services	Worker Group Services
ocnadduirouter	kafka-broker
ocnaddredundancyagent	ocnaddbsfaggregation
ocnaddhealthmonitoring	ocnaddnrfaggregation

**Table 4-1 (Cont.) Management and Worker Group Services when IntraTLS is enabled**

Management Group Services	Worker Group Services
ocnaddexportservice	ocnaddseppaggregation
ocnaddconfiguration	ocnaddscpaggregation
ocnaddbackupprestore	ocnaddpcfaggregation
ocnaddalarm	kraft-controller
ocnaddadminservice	ocnaddpreupgradeaclhelmhook
-	adapter
-	ocnaddcorrelation
-	ocnaddfilter
-	ocnaddingressadapter
-	ocnaddnonoracleaggregation
-	ocnaddstorageadapterservice

**IntraTLS Disabled**

When intraTLS is disabled, the services mentioned in the below tables requires the certificates to be created for them

**Table 4-2 Management and Worker Group Services when IntraTLS is disabled**

Management Group Services	Worker Group Services
ocnaddredundancyagent	adapter
ocnaddbackupprestore	kafka-broker
-	kraft-controller
-	ocnaddingressadapter

**Note**

- In case intraTLS is disabled then ocnaddfilter service must also be disabled in the charts as ocnaddfilter service can not work without Kafka ACLs and that requires intraTLS as enabled

**Caution**

- Do not delete any secrets that do not correspond to specific services (i.e., the secret name does not contain a specific service name), for example, certdbfilesecret, ocnaddpreupgradeaclhelmhook-secret. These are internal secrets required by Data Director.

**Enabling/Disabling intraTLS in OCNADD**

In addition to the external communication, TLS is also used for internal OCNADD communication by default. To enable or disable internal TLS, the following steps need to be undertaken:

**Note**

The user must ensure before proceeding further that all required certificates as indicated above are created for the services using section "Configuring SSL or TLS Certificates" in *Oracle Communications Network Analytics Data Director Installation, Upgrade Guide and Fault Recovery Guide*.  
For details on adding entries for service certificates, see the "[Certificate and Secret Generation](#)" section.

**Enable intraTLS in OCNADD**

To enable or disable internal TLS, the following steps need to be undertaken:

- Enable internal TLS in Helm Charts.
  - Change the value of `global.ssl.intraTlsEnabled` to `true` if not already set in the `ocnadd-custom-values-25.2.100.yaml`. It is set to `true` by default.

```
ssl:
  intraTlsEnabled: true
```

- Upgrade OCNADD deployment/s to enable TLS for internal communications

- **If centralized deployment model is used:**

- \* First, upgrade management group deployment:

```
helm upgrade -n <management-namespace> -f <management-custom-values> <management-release-name> <release-chart>
```

For example:

```
helm upgrade -n ocnadd-deploy-mgmt -f ocnadd-custom-values-mgmt.yaml ocnadd-mgmt ocnadd
```

- \* Then, upgrade each worker group deployment after management group upgrade is complete with Consumer Adapter, Correlation, Storage Adapter, Ingress Adapter upgrade flags set to `TRUE` in the configuration:

```
helm upgrade -n <worker-group-namespace> -f <worker-group-custom-values> <worker-group-release-name> <release-chart>
```

**Note**

Use the `mgmt` group namespace for the default worker group.

For example:

```
helm upgrade -n ocnadd-deploy-wg1 -f ocnadd-custom-values-wg.yaml ocnadd-wg ocnadd
```

## Disable intraTLS in OCNADD

### 1. Disable Internal TLS in Helm Charts:

- a. Change the value of `global.ssl.intraTlsEnabled` to `false` in the `ocnadd-custom-values-25.2.100.yaml` file. It is set to `true` by default.
- b. Example setting:

```
ssl:
  intraTlsEnabled: false
```

### 2. Upgrade OCNADD Deployments to Disable TLS for Internal Communications:

#### a. If a Centralized Deployment Model is Used:

- i. First, upgrade the management group deployment:

```
helm upgrade -n <management-namespace> -f <management-custom-values> <management-release-name> <release-chart>
```

For example:

```
helm upgrade -n ocnadd-deploy-mgmt -f ocnadd-custom-values-mgmt.yaml ocnadd-mgmt ocnadd
```

- b. Then, upgrade each worker group deployment after the management group upgrade is complete:

```
helm upgrade -n <worker-group-namespace> -f <worker-group-custom-values> <worker-group-release-name> <release-chart>
```

#### Note

Use the management group namespace for the default worker group.

For example:

```
helm upgrade -n ocnadd-deploy-wg1 -f ocnadd-custom-values-wg.yaml ocnadd-wg ocnadd
```

#### Note

- External Kafka feeds with filter and correlation features cannot be used without enabling internal TLS.
- The Export feature will not work as it is dependent on the Correlation feeds, which require intraTLS to be enabled.

## Certificate and Secret Generation

- [Certificates produced Manually](#)

- [Certificates Produced by OCCM](#)

### Certificates produced Manually

The OCNADD services can communicate with each other as well as with external interfaces in both secure encrypted mode as well as in insecure mode. For establishing encrypted communication between the services, there is a necessity to generate TLS certificates and private keys for each microservice. The service certificate is generated using the provided CA certificate see the *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide* section "Configuring SSL or TLS Certificates"

The generated service certificates are stored as the Kubernetes secret.

Below are sample values files with/without internal TLS enabled:

where:

- ocnadd-deploy: is the namespace where OCNADD is deployed
- occne-ocnadd: is the Kubernetes domain name

Update the "global" section as per the requirements. The namespace and domain name will be handled by the script.

### For Centralized mode with default worker group

**intraTLS mode is enabled (all the services should use the TLS certificates).**

```
# Do not modify any keys in global section. Please edit only values present
# in global section.
# Edit only commonName value for Root CA. Do not modify key
# You can add multiple services in same manner as the sample services are
# added. The format should be as follows
#service name, common name for service and list of subject alternate name
#e.g.,
#[<service_name>]
#commonName=your.svc.common.name
#IP.1 = 127.0.0.1
#IP.2 = 10.72.31.4
#DNS.1 = localhost
#DNS.2 = svc.cluster.local
# Make sure to provide a single empty line (without space) after end of every
# section
# Do not add comments anywhere in this script to avoid parsing error

[global]
countryName=IN
stateOrProvinceName=KA
localityName=BLR
organizationName=ORACLE
organizationalUnitName=CGBU
defaultDays=365

##root_ca
commonName=*.svc.occne-ocdd

[kafka-broker]
client.commonName=kafka-broker-zk
server.commonName=kafka-broker
```

```
DNS.1=*.kafka-broker.ocnadd-deploy.svc.occne-ocdd
DNS.2=kafka-broker
DNS.3=*.kafka-broker
DNS.4=kafka-broker-0.kafka-broker
DNS.5=kafka-broker-1.kafka-broker
DNS.6=kafka-broker-2.kafka-broker
DNS.7=kafka-broker-3.kafka-broker
DNS.8=kafka-broker.ocnadd-deploy
DNS.9=kafka-broker-0.kafka-broker.ocnadd-deploy
DNS.10=kafka-broker-1.kafka-broker.ocnadd-deploy
DNS.11=kafka-broker-2.kafka-broker.ocnadd-deploy
DNS.12=kafka-broker-3.kafka-broker.ocnadd-deploy
DNS.13=*.kafka-broker-headless.ocnadd-deploy.svc.occne-ocdd
DNS.14=kafka-broker-headless
DNS.15=*.kafka-broker-headless
DNS.16=kafka-broker-0.kafka-broker-headless
DNS.17=kafka-broker-1.kafka-broker-headless
DNS.18=kafka-broker-2.kafka-broker-headless
DNS.19=kafka-broker-3.kafka-broker-headless
DNS.20=kafka-broker-headless.ocnadd-deploy
DNS.21=kafka-broker-0.kafka-broker-headless.ocnadd-deploy
DNS.22=kafka-broker-1.kafka-broker-headless.ocnadd-deploy
DNS.23=kafka-broker-2.kafka-broker-headless.ocnadd-deploy
DNS.24=kafka-broker-3.kafka-broker-headless.ocnadd-deploy
```

```
[ocnaddthirdpartyconsumer]
client.commonName=ocnaddthirdpartyconsumer-client
server.commonName=ocnaddthirdpartyconsumer
DNS.1=*.ocnaddthirdpartyconsumer.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddthirdpartyconsumer
DNS.3=ocnaddthirdpartyconsumer.ocnadd-deploy
DNS.4=ocnaddthirdpartyconsumer.ocnadd-deploy.svc.occne-ocdd
```

```
[oraclenfproducer]
client.commonName=oraclenfproducer
server.commonName=oraclenfproducer-server
DNS.1=*.oraclenfproducer.ocnadd-deploy.svc.occne-ocdd
DNS.2=oraclenfproducer
DNS.3=oraclenfproducer.ocnadd-deploy
```

```
[ocnadduirouter]
client.commonName=ocnadduirouter-client
server.commonName=ocnadduirouter
DNS.1=*.ocnadduirouter.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnadduirouter
DNS.3=ocnadduirouter.ocnadd-deploy
```

```
[ocnaddadminservice]
client.commonName=ocnaddadminservice-client
server.commonName=ocnaddadminservice
DNS.1=*.ocnaddadminservice.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddadminservice
DNS.3=ocnaddadminservice.ocnadd-deploy
```

```
[ocnaddalarm]
client.commonName=ocnaddalarm-client
server.commonName=ocnaddalarm
DNS.1=*.ocnaddalarm.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddalarm
DNS.3=ocnaddalarm.ocnadd-deploy

[ocnaddconfiguration]
client.commonName=ocnaddconfiguration-client
server.commonName=ocnaddconfiguration
DNS.1=*.ocnaddconfiguration.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddconfiguration
DNS.3=ocnaddconfiguration.ocnadd-deploy

[ocnaddhealthmonitoring]
client.commonName=ocnaddhealthmonitoring-client
server.commonName=ocnaddhealthmonitoring
DNS.1=*.ocnaddhealthmonitoring.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddhealthmonitoring
DNS.3=ocnaddhealthmonitoring.ocnadd-deploy

[ocnaddscppaggregation]
client.commonName=ocnaddscppaggregation-client
server.commonName=ocnaddscppaggregation
DNS.1=*.ocnaddscppaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddscppaggregation
DNS.3=ocnaddscppaggregation.ocnadd-deploy

[ocnaddnrfaggregation]
client.commonName=ocnaddnrfaggregation-client
server.commonName=ocnaddnrfaggregation
DNS.1=*.ocnaddnrfaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddnrfaggregation
DNS.3=ocnaddnrfaggregation.ocnadd-deploy

[ocnaddpcfaggregation]
client.commonName=ocnaddpcfaggregation-client
server.commonName=ocnaddpcfaggregation
DNS.1=*.ocnaddpcfaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddpcfaggregation
DNS.3=ocnaddpcfaggregation.ocnadd-deploy

[ocnaddbsfaggregation]
client.commonName=ocnaddbsfaggregation-client
server.commonName=ocnaddbsfaggregation
DNS.1=*.ocnaddbsfaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddbsfaggregation
DNS.3=ocnaddbsfaggregation.ocnadd-deploy

[ocnaddseppaggregation]
client.commonName=ocnaddseppaggregation-client
server.commonName=ocnaddseppaggregation
DNS.1=*.ocnaddseppaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddseppaggregation
DNS.3=ocnaddseppaggregation.ocnadd-deploy
```



```
[ocnaddnonoracleaggregation]
client.commonName=ocnaddnonoracleaggregation-client
server.commonName=ocnaddnonoracleaggregation
DNS.1=*.ocnaddnonoracleaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddnonoracleaggregation
DNS.3=ocnaddnonoracleaggregation.ocnadd-deploy
```

```
[ocnaddingressadapter]
client.commonName=ocnaddingress-client
server.commonName=ocnaddingress
DNS.1=*.ocnaddingressadapter.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddingressadapter
DNS.3=ocnaddingressadapter.ocnadd-deploy
DNS.4=*ingress-adapter.ocnadd-deploy.svc.occne-ocdd
DNS.5=*ingress-adapter.ocnadd-deploy
DNS.6=*ingress-adapter
```

```
[adapter]
client.commonName=adapter
server.commonName=adapter-server
DNS.1=*adapter.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddconsumeradapter
DNS.3=ocnaddconsumeradapter.ocnadd-deploy
```

```
[ocnaddcorrelation]
client.commonName=ocnaddcorrelation-client
server.commonName=ocnaddcorrelation
DNS.1=*.ocnaddcorrelation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddcorrelation
DNS.3=ocnaddcorrelation.ocnadd-deploy
```

```
[ocnaddfilter]
client.commonName=ocnaddfilter-client
server.commonName=ocnaddfilter
DNS.1=*.ocnaddfilter.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddfilter
DNS.3=ocnaddfilter.ocnadd-deploy
```

```
[ocnaddbackupprestore]
client.commonName=ocnaddbackupprestore
server.commonName=ocnaddbackupprestore-server
DNS.1=ocnaddbackupprestore
```

```
[ocnaddpreupgradeaclhelmhook]
client.commonName=ocnaddpreupgradeaclhelmhook-client
server.commonName=ocnaddpreupgradeaclhelmhook
DNS.1=*.ocnaddpreupgradeaclhelmhook.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddpreupgradeaclhelmhook
DNS.3=ocnaddpreupgradeaclhelmhook.ocnadd-deploy
```

```
[ocnaddredundancyagent]
client.commonName=ocnaddredundancyagent-client
server.commonName=ocnaddredundancyagent
DNS.1=*.ocnaddredundancyagent.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddredundancyagent
DNS.3=ocnaddredundancyagent.ocnadd-deploy
```

```
[ocnaddstorageadapterservice]
client.commonName=ocnaddstorageadapterservice-client
server.commonName=ocnaddstorageadapterservice
DNS.1=*.ocnaddstorageadapterservice.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddstorageadapterservice
DNS.3=ocnaddstorageadapterservice.ocnadd-deploy

[ocnaddexportservice]
client.commonName=ocnaddexportservice-client
server.commonName=ocnaddexportservice
DNS.1=*.ocnaddexportservice.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddexportservice
DNS.3=ocnaddexportservice.ocnadd-deploy

[kraft-controller]
client.commonName=kraft-controller-zk
server.commonName=kraft-controller
DNS.1=*.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.2=kraft-controller
DNS.3=*.kraft-controller
DNS.4=kraft-controller-0.kraft-controller
DNS.5=kraft-controller-1.kraft-controller
DNS.6=kraft-controller-2.kraft-controller
DNS.7=kraft-controller.ocnadd-deploy
DNS.8=kraft-controller-0.kraft-controller.ocnadd-deploy
DNS.9=kraft-controller-1.kraft-controller.ocnadd-deploy
DNS.10=kraft-controller-2.kraft-controller.ocnadd-deploy
DNS.11=kraft-controller-0.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.12=kraft-controller-1.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.13=kraft-controller-2.kraft-controller.ocnadd-deploy.svc.occne-ocdd

##end
```

### **intraTLS mode is disabled (only services indicated below should use the TLS certificates)**

```
# Do not modify any keys in global section. Please edit only values present
in global section.
# Edit only commonName value for Root CA. Do not modify key # You can add
multiple services in same manner as the sample services are added. The format
should be as follows
#service name, common name for service and list of subject alternate name
#e.g.,
#[<service_name>]
#commonName=your.svc.common.name
#IP.1 = 127.0.0.1
#IP.2 = 10.72.31.4
#DNS.1 = localhost
#DNS.2 = svc.cluster.local
# Make sure to provide a single empty line (without space) after end of every
section
# Do not add comments anywhere in this script to avoid parsing error

[global]
```

```
countryName=IN
stateOrProvinceName=KA
localityName=BLR
organizationName=ORACLE
organizationalUnitName=CGBU
defaultDays=90

##root_ca
commonName=*.svc.occne-ocdd

[kafka-broker]
client.commonName=kafka-broker-zk
server.commonName=kafka-broker
DNS.1=*.kafka-broker.ocnadd-deploy.svc.occne-ocdd
DNS.2=kafka-broker
DNS.3=*.kafka-broker
DNS.4=kafka-broker-0.kafka-broker
DNS.5=kafka-broker-1.kafka-broker
DNS.6=kafka-broker-2.kafka-broker
DNS.7=kafka-broker-3.kafka-broker
DNS.8=kafka-broker.ocnadd-deploy
DNS.9=kafka-broker-0.kafka-broker.ocnadd-deploy
DNS.10=kafka-broker-1.kafka-broker.ocnadd-deploy
DNS.11=kafka-broker-2.kafka-broker.ocnadd-deploy
DNS.12=kafka-broker-3.kafka-broker.ocnadd-deploy

[kraft-controller]
client.commonName=kraft-controller-zk
server.commonName=kraft-controller
DNS.1=*.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.2=kraft-controller
DNS.3=*.kraft-controller
DNS.4=kraft-controller-0.kraft-controller
DNS.5=kraft-controller-1.kraft-controller
DNS.6=kraft-controller-2.kraft-controller
DNS.7=kraft-controller.ocnadd-deploy
DNS.8=kraft-controller-0.kraft-controller.ocnadd-deploy
DNS.9=kraft-controller-1.kraft-controller.ocnadd-deploy
DNS.10=kraft-controller-2.kraft-controller.ocnadd-deploy
DNS.11=kraft-controller-0.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.12=kraft-controller-1.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.13=kraft-controller-2.kraft-controller.ocnadd-deploy.svc.occne-ocdd

[adapter]
client.commonName=adapter
server.commonName=adapter-server
DNS.1=*.adapter.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddconsumeradapter
DNS.3=ocnaddconsumeradapter.ocnadd-deploy

[ocnaddingressadapter]
client.commonName=ocnaddingress-client
server.commonName=ocnaddingress
```

```

DNS.1=*.ocnaddingressadapter.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddingressadapter
DNS.3=ocnaddingressadapter.ocnadd-deploy
DNS.4=*ingress-adapter.ocnadd-deploy.svc.occne-ocdd
DNS.5=*ingress-adapter.ocnadd-deploy
DNS.6=*ingress-adapter

[ocnaddbackupprestore]
client.commonName=ocnaddbackupprestore
server.commonName=ocnaddbackupprestore-server
DNS.1=ocnaddbackupprestore

[ocnaddredundancyagent]
client.commonName=ocnaddredundancyagent-client
server.commonName=ocnaddredundancyagent
DNS.1=*.ocnaddredundancyagent.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddredundancyagent
DNS.3=ocnaddredundancyagent.ocnadd-deploy

##end

```

#### Note

- External Kafka feed with filter and correlation features cannot be used without enabling internal TLS. Examples include CORRELATED FEED, FILTER CORRELATED FEED, AGGREGATED KAFKA FEED, etc.
- The Export feature will not work as it is dependent on the Correlation feeds, which require intraTLS to be enabled.

### For Centralized Deployment with non-default or more than one worker groups

**When intraTLS mode is enabled, all the services in both the management and worker group(s) should use the TLS certificates to ensure secure internal communication.**

- The `ssl_certs/default_values/management_service_values_template` or its copies should be configured as follows:

```

# Do not modify any keys in global section. Please edit only values
present in global section.
# Edit only commonName value for Root CA. Do not modify key
# You can add multiple services in same manner as the sample services are
added. The format should be as follows
#service name, common name for service and list of subject alternate name
#e.g.,
#[<service_name>]
#commonName=your.svc.common.name
#IP.1 = 127.0.0.1
#IP.2 = 10.72.31.4
#DNS.1 = localhost
#DNS.2 = svc.cluster.local
# Make sure to provide a single empty line (without space) after end of
every section

```

```
# Do not add comments anywhere in this script to avoid parsing error

[global]
countryName=IN
stateOrProvinceName=KA
localityName=BLR
organizationName=ORACLE
organizationalUnitName=CGBU
defaultDays=365

##root_ca
commonName=*.svc.occne-ocdd

[ocnadduirouter]
client.commonName=ocnadduirouter-client
server.commonName=ocnadduirouter
DNS.1=*.ocnadduirouter.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnadduirouter
DNS.3=ocnadduirouter.ocnadd-deploy

[ocnaddalarm]
client.commonName=ocnaddalarm-client
server.commonName=ocnaddalarm
DNS.1=*.ocnaddalarm.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddalarm
DNS.3=ocnaddalarm.ocnadd-deploy

[ocnaddconfiguration]
client.commonName=ocnaddconfiguration-client
server.commonName=ocnaddconfiguration
DNS.1=*.ocnaddconfiguration.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddconfiguration
DNS.3=ocnaddconfiguration.ocnadd-deploy

[ocnaddhealthmonitoring]
client.commonName=ocnaddhealthmonitoring-client
server.commonName=ocnaddhealthmonitoring
DNS.1=*.ocnaddhealthmonitoring.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddhealthmonitoring
DNS.3=ocnaddhealthmonitoring.ocnadd-deploy

[ocnaddbackuprestore]
client.commonName=ocnaddbackuprestore
server.commonName=ocnaddbackuprestore-server
DNS.1=ocnaddbackuprestore

[ocnaddredundancyagent]
client.commonName=ocnaddredundancyagent-client
server.commonName=ocnaddredundancyagent
DNS.1=*.ocnaddredundancyagent.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddredundancyagent
DNS.3=ocnaddredundancyagent.ocnadd-deploy

[ocnaddexportservice]
client.commonName=ocnaddexportservice-client
server.commonName=ocnaddexportservice
```

```
DNS.1=*.ocnaddexportservice.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddexportservice
DNS.3=ocnaddexportservice.ocnadd-deploy

##end
```

- The `ssl_certs/default_values/worker_group_service_values_template` or its copies should be configured as follows:

```
# Do not modify any keys in global section. Please edit only values
present in global section.
# Edit only commonName value for Root CA. Do not modify key
# You can add multiple services in same manner as the sample services are
added. The format should be as follows
#service name, common name for service and list of subject alternate name
#e.g.,
#[<service_name>]
#commonName=your.svc.common.name
#IP.1 = 127.0.0.1
#IP.2 = 10.72.31.4
#DNS.1 = localhost
#DNS.2 = svc.cluster.local
# Make sure to provide a single empty line (without space) after end of
every section
# Do not add comments anywhere in this script to avoid parsing error
```

```
[global]
countryName=IN
stateOrProvinceName=KA
localityName=BLR
organizationName=ORACLE
organizationalUnitName=CGBU
defaultDays=365
```

```
##root_ca
commonName=*.svc.occne-ocdd
```

```
[kafka-broker]
client.commonName=kafka-broker-zk
server.commonName=kafka-broker
DNS.1=*.kafka-broker.ocnadd-deploy.svc.occne-ocdd
DNS.2=kafka-broker
DNS.3=*.kafka-broker
DNS.4=kafka-broker-0.kafka-broker
DNS.5=kafka-broker-1.kafka-broker
DNS.6=kafka-broker-2.kafka-broker
DNS.7=kafka-broker-3.kafka-broker
DNS.8=kafka-broker.ocnadd-deploy
DNS.9=kafka-broker-0.kafka-broker.ocnadd-deploy
DNS.10=kafka-broker-1.kafka-broker.ocnadd-deploy
DNS.11=kafka-broker-2.kafka-broker.ocnadd-deploy
DNS.12=kafka-broker-3.kafka-broker.ocnadd-deploy
DNS.13=*.kafka-broker-headless.ocnadd-deploy.svc.occne-ocdd
DNS.14=kafka-broker-headless
DNS.15=*.kafka-broker-headless
DNS.16=kafka-broker-0.kafka-broker-headless
```

```
DNS.17=kafka-broker-1.kafka-broker-headless
DNS.18=kafka-broker-2.kafka-broker-headless
DNS.19=kafka-broker-3.kafka-broker-headless
DNS.20=kafka-broker-headless.ocnadd-deploy
DNS.21=kafka-broker-0.kafka-broker-headless.ocnadd-deploy
DNS.22=kafka-broker-1.kafka-broker-headless.ocnadd-deploy
DNS.23=kafka-broker-2.kafka-broker-headless.ocnadd-deploy
DNS.24=kafka-broker-3.kafka-broker-headless.ocnadd-deploy

[ocnaddadminservice]
client.commonName=ocnaddadminservice-client
server.commonName=ocnaddadminservice
DNS.1=*.ocnaddadminservice.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddadminservice
DNS.3=ocnaddadminservice.ocnadd-deploy

[ocnaddscppaggregation]
client.commonName=ocnaddscppaggregation-client
server.commonName=ocnaddscppaggregation
DNS.1=*.ocnaddscppaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddscppaggregation
DNS.3=ocnaddscppaggregation.ocnadd-deploy

[ocnaddnrfaggregation]
client.commonName=ocnaddnrfaggregation-client
server.commonName=ocnaddnrfaggregation
DNS.1=*.ocnaddnrfaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddnrfaggregation
DNS.3=ocnaddnrfaggregation.ocnadd-deploy

[ocnaddpcfaggregation]
client.commonName=ocnaddpcfaggregation-client
server.commonName=ocnaddpcfaggregation
DNS.1=*.ocnaddpcfaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddpcfaggregation
DNS.3=ocnaddpcfaggregation.ocnadd-deploy

[ocnaddbsfaggregation]
client.commonName=ocnaddbsfaggregation-client
server.commonName=ocnaddbsfaggregation
DNS.1=*.ocnaddbsfaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddbsfaggregation
DNS.3=ocnaddbsfaggregation.ocnadd-deploy

[ocnaddseppaggregation]
client.commonName=ocnaddseppaggregation-client
server.commonName=ocnaddseppaggregation
DNS.1=*.ocnaddseppaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddseppaggregation
DNS.3=ocnaddseppaggregation.ocnadd-deploy

[ocnaddnonoracleaggregation]
client.commonName=ocnaddnonoracleaggregation-client
server.commonName=ocnaddnonoracleaggregation
DNS.1=*.ocnaddnonoracleaggregation.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddnonoracleaggregation
```

```
DNS.3=ocnaddnonoracleaggregation.ocnadd-deploy
```

```
[ocnaddfilter]
```

```
client.commonName=ocnaddfilter-client
```

```
server.commonName=ocnaddfilter
```

```
DNS.1=*.ocnaddfilter.ocnadd-deploy.svc.occne-ocdd
```

```
DNS.2=ocnaddfilter
```

```
DNS.3=ocnaddfilter.ocnadd-deploy
```

```
[ocnaddingressadapter]
```

```
client.commonName=ocnaddingress-client
```

```
server.commonName=ocnaddingress
```

```
DNS.1=*.ocnaddingressadapter.ocnadd-deploy.svc.occne-ocdd
```

```
DNS.2=ocnaddingressadapter
```

```
DNS.3=ocnaddingressadapter.ocnadd-deploy
```

```
DNS.4=*ingress-adapter.ocnadd-deploy.svc.occne-ocdd
```

```
DNS.5=*ingress-adapter.ocnadd-deploy
```

```
DNS.6=*ingress-adapter
```

```
[adapter]
```

```
client.commonName=adapter
```

```
server.commonName=adapter-server
```

```
DNS.1=*adapter.ocnadd-deploy.svc.occne-ocdd
```

```
DNS.2=ocnaddconsumeradapter
```

```
DNS.3=ocnaddconsumeradapter.ocnadd-deploy
```

```
[ocnaddcorrelation]
```

```
client.commonName=ocnaddcorrelation-client
```

```
server.commonName=ocnaddcorrelation
```

```
DNS.1=*.ocnaddcorrelation.ocnadd-deploy.svc.occne-ocdd
```

```
DNS.2=ocnaddcorrelation
```

```
DNS.3=ocnaddcorrelation.ocnadd-deploy
```

```
[ocnaddstorageadapterservice]
```

```
client.commonName=ocnaddstorageadapterservice-client
```

```
server.commonName=ocnaddstorageadapterservice
```

```
DNS.1=*.ocnaddstorageadapterservice.ocnadd-deploy.svc.occne-ocdd
```

```
DNS.2=ocnaddstorageadapterservice
```

```
DNS.3=ocnaddstorageadapterservice.ocnadd-deploy
```

```
[ocnaddpreupgradeaclhelmhook]
```

```
client.commonName=ocnaddpreupgradeaclhelmhook-client
```

```
server.commonName=ocnaddpreupgradeaclhelmhook
```

```
DNS.1=*.ocnaddpreupgradeaclhelmhook.ocnadd-deploy.svc.occne-ocdd
```

```
DNS.2=ocnaddpreupgradeaclhelmhook
```

```
DNS.3=ocnaddpreupgradeaclhelmhook.ocnadd-deploy
```

```
[kraft-controller]
```

```
client.commonName=kraft-controller-zk
```

```
server.commonName=kraft-controller
```

```
DNS.1=*.kraft-controller.ocnadd-deploy.svc.occne-ocdd
```

```
DNS.2=kraft-controller
```

```
DNS.3=*.kraft-controller
```

```
DNS.4=kraft-controller-0.kraft-controller
```

```
DNS.5=kraft-controller-1.kraft-controller
```

```
DNS.6=kraft-controller-2.kraft-controller
```



```

DNS.7=kraft-controller.ocnadd-deploy
DNS.8=kraft-controller-0.kraft-controller.ocnadd-deploy
DNS.9=kraft-controller-1.kraft-controller.ocnadd-deploy
DNS.10=kraft-controller-2.kraft-controller.ocnadd-deploy
DNS.11=kraft-controller-0.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.12=kraft-controller-1.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.13=kraft-controller-2.kraft-controller.ocnadd-deploy.svc.occne-ocdd

##end

```

### Note

- External Kafka feeds with filter and correlation features cannot be used without enabling internal TLS. Examples include CORRELATED FEED, FILTER CORRELATED FEED, AGGREGATED KAFKA FEED, etc.
- The Export feature will not function as it depends on Correlation feeds, which require intraTLS to be enabled.

**When intraTLS mode is disabled, the services indicated below in both the management and worker group(s) should still use the TLS certificates to maintain secure communication.**

- The `ssl_certs/default_values/management_service_values_template` or its copies should be configured as follows:

```

# Do not modify any keys in global section. Please edit only values
present in global section.
# Edit only commonName value for Root CA. Do not modify key
# You can add multiple services in same manner as the sample services are
added. The format should be as follows #service name, common name for
service and list of subject alternate name
#e.g.,
#[<service_name>]
#commonName=your.svc.common.name
#IP.1 = 127.0.0.1
#IP.2 = 10.72.31.4
#DNS.1 = localhost
#DNS.2 = svc.cluster.local
# Make sure to provide a single empty line (without space) after end of
every section # Do not add comments anywhere in this script to avoid
parsing error

[global]
countryName=IN
stateOrProvinceName=KA
localityName=BLR
organizationName=ORACLE
organizationalUnitName=CGBU
defaultDays=90

##root_ca
commonName=*.svc.occne-ocdd

```

```
[ocnaddbackupprestore]
client.commonName=ocnaddbackupprestore
server.commonName=ocnaddbackupprestore-server
DNS.1=ocnaddbackupprestore

[ocnaddredundancyagent]
client.commonName=ocnaddredundancyagent-client
server.commonName=ocnaddredundancyagent
DNS.1=*.ocnaddredundancyagent.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddredundancyagent
DNS.3=ocnaddredundancyagent.ocnadd-deploy

##end
```

- The `ssl_certs/default_values/worker_group_service_values_template` or its copies should be configured as follows:

```
# Do not modify any keys in global section. Please edit only values
present in global section.
# Edit only commonName value for Root CA. Do not modify key
# You can add multiple services in same manner as the sample services are
added. The format should be as follows
#service name, common name for service and list of subject alternate name
#e.g.,
#[<service_name>]
#commonName=your.svc.common.name
#IP.1 = 127.0.0.1 #IP.2 = 10.72.31.4
#DNS.1 = localhost
#DNS.2 = svc.cluster.local
# Make sure to provide a single empty line (without space) after end of
every section # Do not add comments anywhere in this script to avoid
parsing error
```

```
[global]
countryName=IN
stateOrProvinceName=KA
localityName=BLR
organizationName=ORACLE
organizationalUnitName=CGBU
defaultDays=365

##root_ca
commonName=*.svc.occne-ocdd

[kafka-broker]
client.commonName=kafka-broker-zk
server.commonName=kafka-broker
DNS.1=*.kafka-broker.ocnadd-deploy.svc.occne-ocdd
DNS.2=kafka-broker
DNS.3=*.kafka-broker
DNS.4=kafka-broker-0.kafka-broker
DNS.5=kafka-broker-1.kafka-broker
DNS.6=kafka-broker-2.kafka-broker
DNS.7=kafka-broker-3.kafka-broker
DNS.8=kafka-broker.ocnadd-deploy
```

```

DNS.9=kafka-broker-0.kafka-broker.ocnadd-deploy
DNS.10=kafka-broker-1.kafka-broker.ocnadd-deploy
DNS.11=kafka-broker-2.kafka-broker.ocnadd-deploy
DNS.12=kafka-broker-3.kafka-broker.ocnadd-deploy

[kraft-controller]
client.commonName=kraft-controller-zk
server.commonName=kraft-controller
DNS.1=*kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.2=kraft-controller
DNS.3=*kraft-controller
DNS.4=kraft-controller-0.kraft-controller
DNS.5=kraft-controller-1.kraft-controller
DNS.6=kraft-controller-2.kraft-controller
DNS.7=kraft-controller.ocnadd-deploy
DNS.8=kraft-controller-0.kraft-controller.ocnadd-deploy
DNS.9=kraft-controller-1.kraft-controller.ocnadd-deploy
DNS.10=kraft-controller-2.kraft-controller.ocnadd-deploy
DNS.11=kraft-controller-0.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.12=kraft-controller-1.kraft-controller.ocnadd-deploy.svc.occne-ocdd
DNS.13=kraft-controller-2.kraft-controller.ocnadd-deploy.svc.occne-ocdd

[adapter]
client.commonName=adapter
server.commonName=adapter-server
DNS.1=*adapter.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddconsumeradapter
DNS.3=ocnaddconsumeradapter.ocnadd-deploy

[ocnaddingressadapter]
client.commonName=ocnaddingress-client
server.commonName=ocnaddingress
DNS.1=*ocnaddingressadapter.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddingressadapter
DNS.3=ocnaddingressadapter.ocnadd-deploy
DNS.4=*ingress-adapter.ocnadd-deploy.svc.occne-ocdd
DNS.5=*ingress-adapter.ocnadd-deploy
DNS.6=*ingress-adapter

##end

```

## MTLS Configuration

The customers can opt to use MTLS for internal OCNADD communication. To enable the internal MTLS, run the following steps:

- Enable internal MTLS in Helm Charts.  
Change the value of `global.ssl.mTLS` to "true" in the `ocnadd-custom-values-25.2.100.yaml`. By default it is set to false.

```

ssl:
  intraTlsEnabled: true
  mTLS: true

```

Change the value of `global.acl.kafkaClientAuth` to `required` in `ocnadd-custom-values-25.2.100.yaml`

```
global
  acl
    kafkaClientAuth: required
```

- The certificate creation step remains the same as mentioned in the "TLS Configuration" section.

**Note**

It is mandatory to enable internal TLS and to create certificates and secrets for mTLS to work.

### Certificates Produced by OCCM

Oracle Communications Certificate Management (OCCM) is an automated solution for managing the certificates needed for Oracle Communications Network Analytics Data Director services. OCCM constantly monitors and renews the certificates based on their validity or expiry period.

The created certificates will already have the required TLS/mTLS configuration, eliminating the need for additional steps to configure mTLS and TLS requirements.

To generate certificates using OCCM, follow these steps:

1. Create secrets as outlined in the "OCCM Secrets" section of the *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide*.
2. Modify the `custom_values.yaml` file according to the "Helm Parameter Configuration for OCCM" section of the *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide*.
3. Run `helm install` command. See "Installing OCNADD Package" section in the *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide*.

### Certificates for Apache Druid

Certificate for Apache Druid must be created using the same CA that was used to create OCNADD's certificate.

### Customizing CSR and Certificate Extensions

OpenSSL creates the Certificate Signing Requests (CSRs) and certificates in the `generate_certs.sh` script of OCNADD. OpenSSL configuration file templates `ssl_certs/templates/services_client_openssl.cnf` and `ssl_certs/templates/services_server_openssl.cnf` are used during CSR or certificate creation.

Modify the files based on customer requirements. For more information, see [OpenSSL x509 v3 Certificate and CSR Configuration](#).

### Key Usage Requirement for Client and Server CSR

- [Certificates Produced Manually](#)
- [Certificates Produced by OCCM](#)

## Certificates Produced Manually

External CA may require specific key usage and external key usage to be mentioned in the CSR. Add or modify the [ req\_ext ] section in both `ssl_certs/templates/services_client_openssl.cnf` and `ssl_certs/templates/services_server_openssl.cnf` files as follows:

```
[ req_ext ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE
keyUsage=critical,digitalSignature,keyEncipherment # Modify Key Usage
extendedKeyUsage=serverAuth,clientAuth # Add Extended Key Usage
```

Also add reference to `req_ext` in [ req ] section as follows:

```
[ req ]
default_bits          = 2048
default_keyfile       = privkey.pem
distinguished_name    = req_distinguished_name
attributes            = req_attributes
x509_extensions       = usr_cert # The extensions to add to the self signed cert
req_extensions        = req_ext # Add extensions required for CSR
```

The above code snippets add the standard key and extended key usage required by the CA to sign the certificate. The user can similarly add other specific usage requirements.

## Certificates Produced by OCCM

In the certificate signing request (CSR) utilized for certificate creation through OCCM, the X509v3 Extended Key Usage is set to "TLS Web Client Authentication" and "TLS Web Server Authentication," specifying the certificate's purpose for client and server authentication. Additionally, the X509v3 Key Usage is set to "DIGITAL\_SIGNATURE" and "KEY\_ENCIPHERMENT," indicating the permitted functions for the key's usage.

## Adding or Updating SAN Entries

- [Certificates Produced Manually](#)
- [Certificates Produced by OCCM](#)

## Certificates Produced Manually

These steps should be followed when Subject Alternative Name (SAN) entries need to be updated or added to certificates, and certificates are generated using CACert and CAKey, or only Certificate Signing Requests (CSR) are generated using the `generate_certs.sh` script.

- If external access is enabled, then while running script, select "y" when prompted to add SAN entries.

```
Do you want to add any IP for adding SAN entries to existing dd services
(y/n): y
```

If user selects "yes," a list of services is shown and user can add the SAN entries for any of the listed services by selecting the corresponding service number.

In the following example, the list of management services is shown for user to add SAN entries. Enter the number corresponding to the service for which user wants to enter IP. After choosing the service, give IP addresses as input, else enter "n" to exit.

For the following services:

1. ocnadduirouter
2. ocnaddadminservice
3. ocnaddalarm
4. ocnaddconfiguration
5. ocnaddhealthmonitoring
6. ocnaddbackuprestore
7. ocnaddredundancyagent

Enter the number corresponding to the service for which you want to add

IP: 3

Please enter IP for the service ocnaddalarm or enter "n" to exit :

10.20.30.40

Please enter IP for the service ocnaddalarm or enter "n" to exit :

10.20.30.41

Please enter IP for the service ocnaddalarm or enter "n" to exit : n

Do you want to add IP to any other service (y/n) : n

- Follow the steps mentioned in section [Renewing OCNADD Certificates](#) to renew or recreate the certificates with the modified SAN entries.

### Certificates Produced by OCCM

To update or add Subject Alternative Name (SAN) entries for Kafka, the redundancy agent, or the ingress adapter, modify the "global.certificates.occm.san" section in the `custom_values.yaml` file with the necessary SAN entries. See "Adding/Updating Loadbalancer IPs in SAN through OCCM" section of the Oracle Communications Network Analytics Data Director Installation and Upgrade Guide. The certificates generated by OCCM will then incorporate the updated SAN configuration.

### Updating SAN Entries

```
san:
  kafka:
    update_required: false
    # Provide Kafka Load balancer IPs here
    ips: ["10.10.10.10"]
    uuid:
      client:
      server:
  redundancy_agent:
    update_required: false
    # Provide Redundancy Agent Load balancer IP here
    ips: ["10.10.10.11"]
    uuid:
      client:
      server:
  ingress_adapter:
    update_required: false
    # Provide every Ingress Adapter's Load Balancer IP here
```

```
ips: ["10.10.10.12"]
uuid:
  client:
  server:
```

## 4.1.2 Network Policies

OCNADD is also shipped with Network Policies for ingress connections which are disabled by default. Customers can configure the intended sources of connections by modifying the `ocnadd-custom-values.yaml` file as mentioned below.

Network policies can be enabled to provide an additional security layer restricting both internal and external connections. The Network Policies are categorized into two categories:

- **Internal Service Connections:** The connections between services that are internal to the OCNADD namespace or internal to the OCNADD management and worker group namespaces.
- **External Service Connections:** The services from sources that are external to the OCNADD namespace or external to both the OCNADD management and worker group namespaces. The external connections can further be categorized into 4 categories:
  - **Connections From CNCC:** For rendering OCNADD Dashboard
  - **Connections From NF:** For connections to Kafka from NF acting as Kafka producer or consumer
  - **Connections From CNE Infrastructure Services:** For fetching metrics and generating alerts
  - **Connections From IP blocks/CIDRS:** For connections to Kafka from NF acting as Kafka producer or consumer

To enable network policies, change the value of `global.network.policy.enable` to `true` in `ocnadd-custom-values.yaml` file, by default it is `false`. By enabling this field, network policies for **Internal Service Connections** are enabled by default, no modifications are required.

```
network:
  # enable network policies
  policy:
    enable: true
```

If using the centralized mode of deployment, include the following in the `ocnadd-custom-values.yaml` file for the management group chart to allow connections from Worker Group:

```
global:
  network:
    ingress:
      namespaces:
        # allow ingress network connnections from below worker group
namespaces/s
  workergroups:      # Considering two worker groups namespaces
    - ocnadd-deploy-wg1
    - ocnadd-deploy-wg2
```

To specify the CNCC namespace, update the values in `global.network.ingress.namespaces.cncc` to the CNCC namespace in the `ocnadd-custom-values.yaml` file.

```
global:
  network:
    ingress:
      namespaces:
        # allow ingress network connections to gui from below cncc
namespace/s
  cncc:                # Allowing connections from CNCC namespace:
occncc
  - ocncnc
```

To specify the NF namespaces, update the values in `global.network.ingress.namespaces.kafka` to the NF namespaces in the `ocnadd-custom-values.yaml` file.

```
global:
  network:
    ingress:
      namespaces:
        # allow ingress network connections to kafka from below
namespace/s
  kafka:                # Allowing connections from NRF, SCP, SEPP
namespaces
  - ocnrf
  - ocsepp
  - ocscp
```

To specify the namespace containing CNE infrastructure services like Prometheus, update the values in `global.network.ingress.namespaces.infra` to the CNE infra namespace in the `ocnadd-custom-values.yaml` file.

```
global:
  network:
    ingress:
      namespaces:
        # allow ingress network connections from below infra
namespace/s
  infra:                # Allowing connection from ocne-infra
  - ocne-infra
```

If connections to Kafka from sources external to the Kubernetes cluster are required, or if connections from specific IP/network are required, update the value in `global.network.ingress.external.enable` to `true`. Then, add the required IP/Network address in CIDR format in `global.network.ingress.external.cidrs.kafka` in the `ocnadd-custom-values.yaml`.



**Note**

Allow connection from Druid IPs to Kafka when Druid is configured to ingest from Kafka.

```
global:
  network:
    # Allow external network connections to kafka from below IPs/CIDRs/
Network
    # needed for external kafka consumer
    external:
      enable: true                                # Changed to true
      cidrs:
        kafka:
          - 10.10.10.10/32                        # Allows connections from
10.10.10.10 IP address
          - 192.168.10.0/24                      # Allows connections from
192.168.10.0 - 192.168.10.255 IP addresses
```

If redundancy agent is enabled, communication from peer redundancy agent in primary or secondary site needs to be allowed. Update the value in `global.network.ingress.external.enable` to `true` and update the peer agent's Loadbalancer IP address in `global.network.ingress.external.cidrs.agent` in the `ocnadd-custom-values.yaml`.

```
global:
  network:
    # Allow external network connections to kafka from below IPs/CIDRs/
Network
    # needed for external kafka consumer
    external:
      enable: true                                # Changed to true
      cidrs:
        agent:
          - 100.10.123.1/32                      # Loadbalancer IP address of
peer redundancy agent
```

**Steps to enable Network Policy**

- Install OCNADD.
- Modify `ocnadd-custom-values.yaml` as needed:
  - For a centralized deployment model, modify connections for NF namespaces or external IP/CIDRs independently for each worker group in their respective `ocnadd-custom-values.yaml` files.
- Perform Helm upgrade of OCNADD deployment:
  - The specific steps for the helm upgrade will vary based on the selected OCNADD deployment model. See *Oracle Communications Network Analytics Data Director Installation, Upgrade Guide, and Fault Recovery Guide* for detailed instructions.
  - In a centralized deployment model, start by performing the helm upgrade for management group services. Subsequently, perform the helm upgrade for each worker group namespace.

**⚠ Caution**

The Kafka and Kraft controller deployments use an external health client for registration with the OCNADD health monitoring service. This external health client uses port 18989 to expose health verification endpoints. These endpoints are accessed by internal Data Director services to retrieve the health status of Kafka and Kraft controllers.

Currently, the network policy does not account for this internal port (18989) for Kafka and Kraft controllers. If a network policy is applied to Kafka and Kraft controllers, care must be taken to ensure that port 18989 is either explicitly allowed in the network policy or that the network policy is disabled during any upgrade or rollback.

Additionally, if services such as the Admin service have a dependency on Kafka during their restart, and the network policy on Kafka does not allow port 18989, those services (for example, Admin service) may get stuck in the `init` state.

## 4.1.3 Kafka Security Configuration

The basic configuration for Kafka security in order to be compliant with product delivery is described below.

### Prerequisites

- SSL certificates & keys (generated using SSL scripts which are provided as part of OCNADD release)

### Configuring SASL for Kafka

Kafka uses the Java Authentication and Authorization Service (JAAS) for SASL configuration. In OCNADD, by default, all Kafka communication (inter-broker, Kafka-client, and Kafka-broker and between Kafka and Zookeeper) happens with a common user (ocnadd) configured in `<chart-path>/charts/ocnaddkafka/config/kafka_server_jaas.conf`

Customers can add more users in `<chart-path>/charts/ocnaddkafka/config/kafka_server_jaas.conf` file. In case of Centralized Deployment, `<chart-path>` should be the path to the worker group services chart, and each worker group can have different users added to them independently of each other.

The "ocnadd" user is the default user for all internal Kafka communication and should not be modified.

Below are the sample changes to add NRF, SCP, and SEPP users in Kafka:

```
KafkaServer {
org.apache.kafka.common.security.plain.PlainLoginModule required
username="ocnadd"
password="<change this>"
user_ocnadd="<change this>";
user_NRF="NRF-secret";
user_SCP="SCP-secret";
user_SEPP="SEPP-secret";
};
Client {
org.apache.zookeeper.server.auth.DigestLoginModule required
username="ocnadd"
```

```
password="<change this>";  
};
```

In the above code snippet `user_NRF="NRF-secret"` is added for NRF user, `user_SEPP="SEPP-secret"` is added for SEPP user, and `user_SCP="SCP-secret"` is added for SCP user. Where "NRF-secret", "SEPP-secret" and "SCP-secret" are the passwords.

After making the above changes, continue with the installation steps as mentioned in the *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.

#### Note

- Field marked as `<change this>` needs to be updated by the user to set the Kafka admin password.
- User addition is one time operation and can be done only during installation.
- It is recommended to use a strong password in `kafka_server_jass.conf`.

### External Kafka Feed

To enable Kafka Feed support, see the steps mentioned in "Enable Kafka Feed Configuration Support" section of *Oracle Communications Network Analytics Data Director User Guide*.

## 4.1.4 MySQL Configuration

The OCNADD uses cnDBtier service for the database, which provides secure database connectivity. For configuration details, see *Oracle Communications Network Analytics Data Director Installation Guide, Upgrade, and Fault Recovery Guide*.

## 4.1.5 Certificate Creation for NFs and Third Party Consumers

The following sections provide information about certificate creation rules to be considered by the NFs and Third Party Consumers.

### Certification Creation for NFs

NFs must ensure the following conditions are met prior to sending data to OCNADD over TLS:

- NFs must trust the Certificate Authority(CA) that issues the certificate to the OCNADD. The CA certificate for OCNADD's Certificate Authority must be included in the trust store, `cacert` file, or the `ca-bundle` of the NF.
- When **mTLS** is enabled in OCNADD and an NF in the same Kubernetes cluster wishes to send data to the SSL endpoint instead of SASL\_SSL endpoint, the TLS certificate of the NF should be issued by the same CA issuing certificate to the OCNADD or any one of the trusted CAs.

### Certification Creation for Third Party Consumers

When TLS encrypted feeds (HTTP2 or TCP\_SECURED) are configured, it is essential that the third party endpoint given in the feed configuration supports TLS. In addition to supporting TLS, consider the following while creating certificates for the consumer:

- The certificate should be signed by any of the trusted CAs or by the CA issuing certificate for the OCNADD.
- The certificates Common Name (CN) or Subject Alternative Name (SAN) contains the host name or IP address provided as part of the feed configuration.  
For example, if Feed A is configured to send data to Consumer A (*https://thirdpartyconsumer/*) and Feed B is configured to send data to Consumer B (*https://10.10.10.10*), the certificate for Consumer A must contain the *thirdpartyconsumer* in CN or SAN of its certificate and similarly Consumer B's certificate must contain 10.10.10.10 in the SAN field of its certificate.
- If **mTLS** is enabled by the Third Party Consumer, it must trust the CA issuing certificate for OCNADD. The CA certificate for OCNADD's CA must be included in the trust store, *cacert* file or *ca-bundle* of the Third Party Consumer.

### CA Certificate Expiry

Suppose the CA certificate expires before the OCNADD certificate expiry. In that case, the customer can renew the CA (if self-managed) and reuse the certificate provided the CA certificate's hash, private key, public key, and so on remain the same.

Renewing the OCNADD certificate and performing a rollout restart after renewing the CA certificate will ensure that OCNADD services are using a valid, non-expired CA certificate. For more information, see, [Renewing OCNADD Certificates](#).

## 4.1.6 Renewing OCNADD Certificates

Follow the steps in the sections below to renew the OCNADD certificates.

### Note

Steps to renew the certificate vary based on the method used to create the certificates.

### Certificate Generated Using CACert and CAKey

If the certificate is generated using CACert and CAKey:

1. Add the services whose certificates must be renewed in the `ssl_certs/default_values/renew_cert_files` file.  
The following code snippet displays the services to be added if Internal TLS is enabled:

```
# This files contain the list of services for which certificate needs to
# be renewed
# The service name should be exactly same for which the certificates has
# been initially generated
# defaultDays is number of days upto which certificate should be renewed.
# Certificate for all listed
# service will be updated with this value.

defaultDays=90

kafka-broker
kraft-controller
ocnadduirouter
ocnaddadminservice
```

```

ocnaddalarm
ocnaddconfiguration
ocnaddhealthmonitoring
ocnaddscppaggregation
ocnaddnrfaggregation
ocnaddseppaggregation
ocnaddpcfaggregation
ocnaddbsfaggregation
ocnaddnonoracleaggregation
adapter
ocnaddingressadapter
ocnaddcorrelation
ocnaddstorageadapterservice
ocnaddexport
ocnaddfilter
ocnaddbackuprestore
ocnaddpreupgradeacshelhook

```

2. Run the following command:

```

./generate_certs.sh -cacert <path to>/cacert.pem -cakey <path to>/
cakey.pem --renew

```

3. Choose the mode of deployment when prompted *Select the mode of deployment*.
4. Choose the namespace where OCNADD, OCNADD management group services or OCNADD worker group services is/are deployed when prompted *Enter kubernetes namespace*:
5. Provide password to your CA key when prompted *Enter passphrase for CA Key file*:
6. Older certificates will be revoked and newer certificates will be created and the rollout restart of services will be performed.

### Only CSR is Generated

If only the CSR is generated:

1. Navigate to `ssl_certs/demoCA/dd_mgmt_worker_services/<namespace>/services/<service-name>/client` OR `ssl_certs/demoCA/dd_mgmt_worker_services/<namespace>/services/<service-name>/server` `ssl_certs/` for each `<service-name>` given in `default_values/values` file.

For example, navigating to client folder for adapter service in worker group namespace `ocnadd-deploy-wg1`

```

[ssl_certs]$ ls
default_values demoCA generate_certs.sh generate_secrets.sh logs template
[ssl_certs]$ cd demoCA/dd_mgmt_worker_services/ocnadd-deploy-wg1/services/
adapter/client/
[client]$ ls
adapter-clientcert.pem adapter-client.csr adapter-clientprivatekey.pem
adapter_client_ssl.cnf client_rsa_certificate.crt

```

2. Delete the old CSR request.  
The following example displays how to delete the old CSR request for the adapter service:

```

[client]$ ls
adapter-clientcert.pem adapter-client.csr adapter-clientprivatekey.pem

```

```

adapter_client_ssl.cnf  client_rsa_certificate.crt
[client]$ rm adapter-client.csr
[client]$ ls
adapter-clientcert.pem  adapter-clientprivatekey.pem
adapter_client_ssl.cnf  client_rsa_certificate.crt

```

3. To create a new CSR (client.csr) from the old certificate (clientcert.pem) and private key (clientprivatekey.pem), run the following command:

```

openssl x509 -x509toreq -in clientcert.pem -signkey clientprivatekey.pem -
out client.csr

```

The following example displays how to create a new client CSR for the adapter service:

```

[client]$ ls
adapter-clientcert.pem  adapter-clientprivatekey.pem
adapter_client_ssl.cnf  client_rsa_certificate.crt
[client]$ openssl x509 -x509toreq -in adapter-clientcert.pem -signkey
adapter-clientprivatekey.pem -out adapter-client.csr
Getting request Private Key
Generating certificate request
[client]$ ls
adapter-clientcert.pem  adapter-client.csr  adapter-clientprivatekey.pem
adapter_client_ssl.cnf  client_rsa_certificate.crt
[client]$

```

4. Repeat the steps 1 up to 3 for all other services.
5. Create new certificates from the newly created CSR and follow the "Generating Certificate Signing Request (CSR)" procedure from step 9 in the *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.
6. Run the following commands to perform rollout restart of all OCNADD services:

```
kubectl rollout restart -n <ocnadd-deploy> deployment
```

Where <ocnadd-deploy> is the namespace where OCNADD, OCNADD management group services or OCNADD worker group services is/are deployed.

### Certificate is Generated Using OCCM

OCCM monitors the certificate validity and initiates automatic certificate renewal based on the renew before period configuration.

Update the `renewBefore (Days)` parameter in the `custom_values.yaml` file to specify the desired number of days before the certificate expiration when renewal should occur. Refer to the Oracle Communications Certificate Management guide to determine the certificate expiration period. After updating the parameter and renewing the certificates, restart the deployments and statefulsets for both the management group and worker group using the following commands:

```

kubectl rollout restart -n <ocnadd-deploy> deployment
kubectl rollout restart -n <ocnadd-deploy> statefulset

```

## 4.1.7 Signing OCI Container Registry Images for Security

To meet security requirements, images can be signed and stored in Oracle Cloud Infrastructure Registry (also known as Container Registry).

Signed images provide a way to verify both the source of an image and its integrity.

Signing an image requires an Encryption Key.

The Encryption Key can be created as follows:

1. Log in to the OCI Console.
2. Click the **Hamburger** menu and select **Identity and Security**.
3. Under **Identity and Security**, select **Vault**.
4. Click the **Create Vault** button.
5. Enter the name of the vault to be kept.
6. Click the **Create Vault** button.
7. After some time, the vault is created.
8. Click the newly created vault.
9. Click **Master Key Encryption**.
10. Click **Create Key**:
  - a. Select **Protection Mode** as **Software**.
  - b. Enter the name of the key to be kept.
  - c. Select **Key Shape Algorithm** as **RSA** and **KeyShape Length** as 2048.
  - d. Click **Create Key**.

To sign the image pushed into Container Registry using the master key and key version in the Vault service:

1. Open the Developer Tool and click **Cloud Shell**.
2. Run the following command to sign the image:

```
oci artifacts container image-signature sign-upload --compartment-id
<compartment-ocid> --kms-key-id <key-ocid> --kms-key-version-id <key-
version-ocid> --signing-algorithm <signing-algorithm> --image-id <image-
ocid> --description <signature-description> --metadata <image-metadata-
json>
```

Where,

- `--compartment-id <compartment-ocid>`: Copy from compartment OCID from Identity >> Compartments >> <YOUR COMPARTMENT> >> OCID Copy.
- `--kms-key-id <key-ocid>`: Copy the Version of key from Key Management & Secret Management >> Vaults >> <VAULT\_NAME> >> KeyDetails >> OCID Copy.
- `--kms-key-version-id <key-version-ocid>`: Copy the Version of key from Key Management & Secret Management >> Vaults >> <VAULT\_NAME> >> KeyDetails >> Key Version Copy.

- `--signing-algorithm <signing-algorithm>`: Any one of the following:  
SHA\_224\_RSA\_PKCS\_PSS, SHA\_256\_RSA\_PKCS\_PSS,  
SHA\_384\_RSA\_PKCS\_PSS, SHA\_512\_RSA\_PKCS\_PSS.
- `--image-id <image-ocid>`: Copy image OCID from Container Registry >>  
<IMAGE\_NAME> >> <VERSION> >> OCID Copy.
- `<Optional>--description <signature-description>`: Description of the signature.
- `<Optional>--metadata <image-metadata-json>`: JSON file that contains metadata of the image.

To verify if the image is signed:

1. Open the navigation menu and click **Developer Services**.
2. Under **Containers & Artifacts**, click **Container Registry**.
3. Select the region that contains the repository.
4. Select the compartment that contains the repository.  
The **Repositories and images** field lists the repositories in the selected region and compartment to which you have access.
5. Click the plus (+) button beside the name of a repository containing signed images.  
Below the name of the repository, the images in the repository are listed that can be identified by the version identifier of each image.

The text **"(Signed)"** appears beside images that have been signed.

## 4.1.8 Cleaning Up Sensitive Information After Installation

The Helm chart contains critical information, which is required only during installation. Hence, take backup of Helm chart and remove it from bastion server after deployment.

## 4.1.9 Druid User Management

By default, Apache Druid has two users: **admin** and **druid\_system**. The **druid\_system** user is used internally by Druid services to communicate with each other. For OCNADD, a user with fewer privileges is sufficient.

The steps to create a new user for OCNADD and assign the required permissions using the REST endpoints exposed by Druid are described below.

### Create OCNADD user

Run the following commands to create the Druid authentication and authorization user for OCNADD:

```
$ curl -k --request POST 'https://<10.10.10.10>:9088/proxy/coordinator/druid-ext/basic-security/authentication/db/db/users/<ocnadd>' -u 'admin:<druid_admin_pwd>'
```

```
$ curl -k --request POST 'https://<10.10.10.10>:9088/proxy/coordinator/druid-ext/basic-security/authentication/db/db/users/<ocnadd>/credentials' -u 'admin:<druid_admin_pwd>' --header 'Content-Type: application/json' --data-raw '{ "password": "<druid_ocnadd_pwd>" }'
```

```
$ curl -k --request POST 'https://<10.10.10.10>:9088/proxy/coordinator/druid-
```



```
ext/basic-security/authorization/db/db/users/<ocnadd_user>' -u  
'admin:<druid_admin_pwd>'
```

Where,

- <10.10.10.10> is the IP of the *druid-router* service.
- <druid\_admin\_pwd> is the password for Druid user with admin privileges
- <ocnadd\_user> is the user created for OCNADD in Druid
- <druid\_ocnadd\_pwd> is the password for OCNADD user

### Create OCNADD Role, Add Permissions and Add User to Role

Run the following command to create a role in Druid for OCNADD, add the required permissions and then add user to that role:

```
$ curl -k --request POST 'https://<10.10.10.10>:9088/proxy/coordinator/druid-  
ext/basic-security/authorization/db/db/roles/<ocnadd_role>' -u  
'admin:<druid_admin_pwd>'
```

```
$ curl -k --request POST 'https://<10.10.10.10>:9088/proxy/coordinator/druid-  
ext/basic-security/authorization/db/db/roles/<ocnadd_role>/permissions' -u  
'admin:<druid_admin_pwd>' --header 'Content-Type: application/json' --data  
'[{"resource":{"name":".*","type":"DATASOURCE"},"action":"READ"}, {"resource":  
{"name":".*","type":"DATASOURCE"},"action":"WRITE"}, {"resource":  
{"name":"CONFIG","type":"CONFIG"},"action":"WRITE"}, {"resource":  
{"name":"CONFIG","type":"CONFIG"},"action":"READ"}]'
```

```
$ curl -k --request POST https://<10.10.10.10>:9088/proxy/coordinator/druid-  
ext/basic-security/authorization/db/db/users/<ocnadd_user>/roles/  
<ocnadd_role> -u 'admin:<druid_admin_pwd>'
```

Where,

- <10.10.10.10> is the CNLB or LoadBalancer IP of the *druid-ocnadd-router* service.
- <druid\_admin\_pwd> is the password for 'admin' user
- <ocnadd\_user> is the user created for OCNADD in Druid
- <ocnadd\_role> is the role created for OCNADD in Druid

## A

# Network Port Flows

This section describes network port flows for CNE and OCNWDAF.

## Network Port Flows

- Cluster IP addresses are reachable outside of the cluster and are typically assigned by using a Network Load Balancer.
- Node IP addresses are reachable from the bastion host (and may be exposed outside of the cluster).

## CNE Port Flows

**Table A-1 CNE Port Flows**

Name	Server/Container	Ingress Port ext[:int]/Proto	TLS	Cluster IP (Service IP)	Node IP	Notes
<b>SSH</b>	ALL	22/TCP	Y		SSH Access	Administrative SSH Access. Only root or key is not allowed.
<b>Repository</b>	Bastion Host	80/TCP,443/TCP,5000/TCP			Repository Access	Access repositories (YUM, Docker, Helm, and so on.)
<b>MySQL Query</b>	MySQL SQL Node	3306/TCP	N		Microservice SQL Access	The SQL Query interfaces are used for NWDAF to access the database.
<b>ETCD Client</b>	Kubernetes Master Nodes	2379/TCP	Y		Client Access	Keystore DB used by Kubernetes
<b>Kubelet API</b>	Kubernetes Nodes	10250/TCP	Y		Control Plane Node Access	API which allows full node access.
<b>Kubelet State</b>	Kubernetes Nodes	10255/TCP	Y		Node State Access	Unauthenticated read-only port, allowing access to node state.
<b>Kube-proxy</b>	Kubernetes Nodes	10256/TCP	N		Health Check	Health check server for Kube Proxy.
<b>Kube-controller</b>	Kubernetes Nodes	10257/TCP	Y		Controller Access	HTTPS Access
<b>Kube-Scheduler</b>	Kubernetes Node	10259/TCP	Y		Scheduler Access	HTTPS Access

Table A-1 (Cont.) CNE Port Flows

Name	Server/Container	Ingress Port ext[:int]/Proto	TLS	Cluster IP (Service IP)	Node IP	Notes
<b>Jaeger Agent</b>	Kubernetes Nodes	5775/UDP	N		Agent	Accepts zipkin.thrift in compact Thrift protocol (deprecated; only used by very old Jaeger clients, circa 2016).
<b>Jaeger Agent</b>	Kubernetes Nodes	5778/TCP	N		Agent	Serves SDK configs, namely sampling strategies at /sampling.
<b>Jaeger Agent</b>	Kubernetes Nodes	6831/UDP	N		Agent	UDP Accepts jaeger.thrift in compact Thrift protocol used by most current Jaeger clients.
<b>Jaeger Agent</b>	Kubernetes Nodes	6831/UDP	N		Agent	UDP Accepts jaeger.thrift in binary Thrift protocol used by Node.js Jaeger client (because thriftrw npm package does not support compact protocol).
<b>Jaeger Agent</b>	Kubernetes Nodes	14271/TCP	N		Agent	Admin port: health check at / and metrics at /metrics.
<b>Jaeger Collector</b>	Kubernetes Nodes	9411/TCP	N		Collector	Accepts Zipkin spans in Thrift, JSON and Proto (disabled by default).
<b>Jaeger Collector</b>	Kubernetes Nodes	14250/TCP	N		Collector	Used by jaeger-agent to send spans in model.proto format.
<b>Jaeger Collector</b>	Kubernetes Nodes	14268/TCP	N		Collector	Accepts spans directly from clients in jaeger.thrift format with binary thrift protocol (POST to /api/traces). Also serves sampling policies at /api/sampling, similar to Agent's port 5778.
<b>Jaeger Collector</b>	Kubernetes Nodes	14269/TCP	N		Collector	Admin port: health check at / and metrics at /metrics.
<b>Jaeger-Query</b>	Kubernetes Nodes	80/TCP	N	GUI		Service frontend
<b>Prometheus Server</b>	Kubernetes Nodes	80/TCP	N	GUI		Prometheus Server

Table A-1 (Cont.) CNE Port Flows

Name	Server/Container	Ingress Port ext[:int]/Proto	TLS	Cluster IP (Service IP)	Node IP	Notes
<b>Prometheus-Exporter</b>	Kubernetes Nodes	9100/TCP	N		Prometheus Exporter	Prometheus Exporter
<b>Alertmanager</b>	Kubernetes Nodes	80/TCP	N	GUI		The Alertmanager handles alerts sent by client applications such as the Prometheus server.

## OCNWDAF Port Flows

Table A-2 OCNWDAF Port Flows

Name	Server / Container	Ingress Port [external]:internal	TLS	Cluster IP (Service IP)	Node IP	Notes
<b>NWDAF</b>	Kubernetes Nodes/ NWDAF Service	8080/TCP	Y	Ingress Gateway	ocn-nwdaf-gateway-service	NWDAF
<b>NWDAF Portal</b>	Kubernetes Nodes/ NWDAF Service	80/TCP	Y	GUI	nwdaf-portal	NWDAF GUI web.