Oracle® Communications

Network Analytics Data Director Security Guide

Release 22.0.0
F74095-01
December 2022

Oracle Communications Network Analytics Data Director Security Guide, Release 22.0.0

F74095-01

Contents

My Oracle Support (MOS)

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

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

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

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

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

Acronyms

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

Table Acronyms and Terminology

Acronym	Definition
CLI	Command Line Interface
MPS	Messages Per Second
NRF	Network Repository Function
OHC	Oracle Help Center
OSDC	Oracle Service Delivery Cloud
SCP	Service Communication Proxy
SVC	Services

What's New in This Guide

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

Release 22.0.0 - F74095-01, December 2022

This is the first release of the document.

1 Introduction

The Security Guide provides an overview of the security information applicable to the Oracle Communications Network Analytics Data Director (OCNADD).

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

Audience

- Technology consultants
- Installers
- Security consultants
- System administrators

References

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

- *Oracle Communications Cloud Native Environment and Installation Guide*
- *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide*
- *Oracle Communications Network Analytics Data Director Troubleshooting Guide*
- *Oracle Communications Network Analytics Data Director Disaster Recovery Guide*
- *Oracle Communications Network Analytics Data Director User Guide*

2 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 Data Director is deployed on the Kubernetes cluster managed by OCCNE.
Cloud	Customer CNE	Customer CNE(TANZU)	The customer provides the CNE and deploys the OCNADD directly into the environment. The Oracle provided DBTier is used. The list of all other required software is listed in the install guide *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide* section "Software Requirements".

OCNADD Services

Table 2-2 OCNADD Services

Services	Descriptio
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 (e.g: SCP 1 & 2 and/or SEPP, NRF, 3rd party NF, etc.).
Alarm Service	The OCNADD Alarm service is used to store the alarms generated by other OCNADD services.
Backend Router Service	The Backend Router Service is the gateway that forwards GUI requests to various other backed services like config, health, alarm, Prometheus, etc.

Table 2-2 (Cont.) OCNADD Services

Services	Descriptio
Configuration Service	This service is used by the Data Director UI service to configure 3rd party message feed.
Consumer Adapter Service	This Consumer Adapter Service allows OCNADD to send data feed to 3rd party consumer applications through the Egress Gateway service.
Egress Gateway Service	The Egress Gateway service is used for all the outbound communication towards the 3rd party consumer applications.The outgoing traffic is controlled and monitored by the Egress Gateway Service.
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.
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.

Secure Development Practices

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

Vulnerability Handling

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

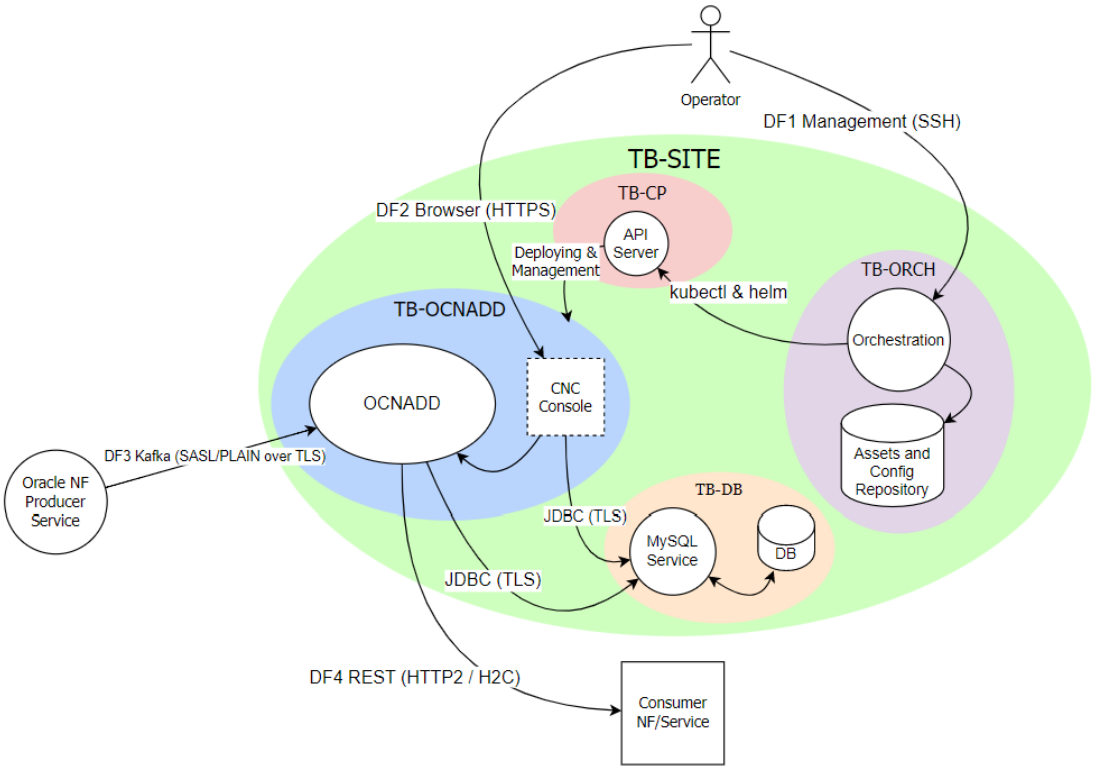
In general, the OCNADD will follow a quarterly release cycle, with each release providing feature updates and fixes, and updates to relevant third-party software. These quarterly releases provide cumulative patch updates.

Trust Model

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.

Context Diagram

Figure 2-1 Context Diagram



Key Trust Boundaries

Following are the key trust boundaries:

Table 2-3 Key Trust Boundaries

Trust Boundary	Access Control
OCNADD	Kubernetes Namespace for Data Director 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.
CNE Infra	Namespace containing all the infrastructure related services (like Prometheus). Provided by CNE.

Table 2-3 (Cont.) Key Trust Boundaries

Trust Boundary	Access Control
Orchestration	Includes the orchestration server and the Code and Image Repository.

External Data Flows

Table 2-4 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 CNCC to create, manage feed configuration and monitor OCNADD. CNCC will be accessed through browser and Operator will be authenticated with username, password before access is granted.
DF3 : Kafka	SASL	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 will forward the message feed to respective consumer NF/s as HTTP2 (over TLS) or H2C (HTTP2 clear text) messages according to the feed configurations.

3
Implementing Security Recommendations and Guidelines

This section provides specific recommendations and guidelines for Oracle Communications Network Analytics Data Director (OCNADD) security.

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. Refer to the next section for more info about certificate creation.

DF4: Consumer NF communication can be configured to use clear text or TLS through the GUI. Refer to the GUI guide for more info on how to do so.

Internal TLS Communication

In addition to the above communications, the customers can also opt to use TLS for internal OCNADD communication. To enable internal TLS, the following steps need to be undertaken:

- Enable internal TLS in Helm Charts.

  Change the value of global.ssl.intraTlsEnabled to *true* in the `<chart-path>/values.yaml`. By default it is set to *false*.

  ```
  ssl:
      intraTlsEnabled:true
  ```

- Create TLS certificates for all the internal services.
  - If the **generate_certs** script is used to create the certificates, add entries for internal services in the `ssl_certs/default_values/values` file. Refer to the "Certificate and Secret Generation" section for more details.
  - The following entries should be added to the values file, if not present:

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

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

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

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

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

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

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

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

Certificate and Secret Generation

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. For more information, refer to the *Oracle Communications Network Analytics Data Director Installation Guide*.

The generated service certificates are stored as the K8s secret.The Certificates for the NF producer and OCNADD Consumer can also be created separately, but it is essential that

- The NF producer includes OCNADD's CA certificate in their trust store.
- The third-party consumer certificate is also created by using the same OCNADD's CA.

> **Note:**
>
> The default certification creation assumes that internal TLS is enabled and creates the certificates for all the OCNADD services. The customers can choose to delete surplus entries from `ssl_certs/default_values/values` when not using internal TLS. This reduces the number of certificates to be signed by the CA.

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 Cluster name
- [ocnaddthirdpartyconsumer] : is the OCNADD consumer
- [oraclenfproducer] : is the Oracle 5G NF/producer

Without Internal TLS

```
# 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=*.ocnadd-deploy.svc.occne-ocnadd

[kafka-broker1]
client.commonName=kafka-broker1-zk
server.commonName=kafka-broker1
DNS.1=*.kafka-broker1.ocnadd-deploy.svc.occne-ocnadd
```

```
DNS.2=kafka-broker1

[kafka-broker2]
client.commonName=kafka-broker2-zk
server.commonName=kafka-broker2
DNS.1=*.kafka-broker2.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker2

[kafka-broker3]
client.commonName=kafka-broker3-zk
server.commonName=kafka-broker3
DNS.1=*.kafka-broker3.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker3

[kafka-broker4]
client.commonName=kafka-broker4-zk
server.commonName=kafka-broker4
DNS.1=*.kafka-broker4.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker4

[kafka-broker5]
client.commonName=kafka-broker5-zk
server.commonName=kafka-broker5
DNS.1=*.kafka-broker5.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker5

[kafka-broker6]
client.commonName=kafka-broker6-zk
server.commonName=kafka-broker6
DNS.1=*.kafka-broker6.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker6

[kafka-broker7]
client.commonName=kafka-broker7-zk
server.commonName=kafka-broker7
DNS.1=*.kafka-broker7.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker7

[egw]
client.commonName=egw-client
server.commonName=egw
DNS.1=*egw.ocnadd-deploy.svc.occne-ocnadd
DNS.2=ocnaddegressgateway

[ocnaddthirdpartyconsumer]
server.commonName=ocnaddthirdpartyconsumer
DNS.1=*.ocnaddthirdpartyconsumer.ocnadd-deploy.svc.occne-ocnadd
DNS.2=ocnaddthirdpartyconsumer

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

```
##end
```

With Internal TLS

```
# 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=*.ocnadd-deploy.svc.occne-ocnadd

[kafka-broker1]
client.commonName=kafka-broker1-zk
server.commonName=kafka-broker1
DNS.1=*.kafka-broker1.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker1

[kafka-broker2]
client.commonName=kafka-broker2-zk
server.commonName=kafka-broker2
DNS.1=*.kafka-broker2.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker2

[kafka-broker3]
client.commonName=kafka-broker3-zk
server.commonName=kafka-broker3
DNS.1=*.kafka-broker3.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker3

[kafka-broker4]
client.commonName=kafka-broker4-zk
```

```
server.commonName=kafka-broker4
DNS.1=*.kafka-broker4.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker4

[kafka-broker5]
client.commonName=kafka-broker5-zk
server.commonName=kafka-broker5
DNS.1=*.kafka-broker5.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker5

[kafka-broker6]
client.commonName=kafka-broker6-zk
server.commonName=kafka-broker6
DNS.1=*.kafka-broker6.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker6

[kafka-broker7]
client.commonName=kafka-broker7-zk
server.commonName=kafka-broker7
DNS.1=*.kafka-broker7.ocnadd-deploy.svc.occne-ocnadd
DNS.2=kafka-broker7

[zookeeper]
client.commonName=zookeeper-zk
server.commonName=zookeeper
DNS.1=*.zookeeper.ocnadd-deploy.svc.occne-ocnadd
DNS.2=zookeeper

[egw]
client.commonName=egw-client
server.commonName=egw
DNS.1=*egw.ocnadd-deploy.svc.occne-ocnadd
DNS.2=ocnaddegressgateway

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

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

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

 [ocnaddadminservice]
client.commonName=ocnaddadmin
service-clientserver.commonName=ocnaddadminservice
```

```
DNS.1=*.ocnaddadminservice.ocnadd-deploy.svc.occne-ocnadd
DNS.2=ocnaddadminservice

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

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

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

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

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

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

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

##end
```

Network Policy

The OCNADD is also shipped with Network Policies for ingress connections which are disabled by default.

The Network Policies can be categorized into three categories:

- **Internal service Connections:** The service connections that are internal to the OCNADD namespace. To enable network policies, change the value of global.network.policy.enable to *true* in `<chart-path>/values.yaml` file. By default, it is *false*.

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

- **External service Connections:** The service connections that are connecting to the OCNADD namespace. The policy gets applied on connections from OCCNCC, OCSCP, OCNRF and so on which are deployed outside the OCNADD namespace. To allow connections from specific namespaces (when network policies are enabled), add the namespace to the namespaces list in global.network.policy.ingress.namespaces of `<chart-path>/values.yaml` file as below:

```
#allow ingress network connections from below namespaces
namespaces:
  -occne-infra
  -occncc
```

  In the above example, the connections from all pods of occne-infra (prometheus, grafana, etc.) and occncc namespace are allowed. Similarly add the SCP, and NRF namespace to this list.

- **External service Connectionfrom specific IP address**: This can be enabled (disabled by default) to allow connections from specific IP address ranges when Network Policy is enabled. In the below example, the connections from the 10.0.0.0/8 subnet are allowed. In other words, all connections from IP address 10.0.0.0 - 10.255.255.255 are allowed.

```
# to allow external connections from postman, curl, etc.
# only needed for development/debugging purposes
external:
   enable:true
   cidrs:
     -10.0.0.0/8
```

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/scripts/kafka_server_jass.conf`

Customers can add more users in **<chart-path>/charts/ocnaddkafka/scripts/kafka_server_jass.conf** file.The "ocnadd" user is the default user for all Kafka communication. and should not be modified.

Below is the sample changes to add NRF and SCP user in Kafka:

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

user_NRF="NRF-secret"; is added for NRF user and **user_SCP="SCP-secret";** is added for SCP user.

After doing the above changes, continue with **step 4** of the section "**Kafka pre-install"** in the *Oracle Communications Network Analytics Data Director Installation Guide.*

> **Note:**
>
> - The 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** and during **jaas-secret** creation.

MySQL Configuration

The OCNADD uses CnDBtier service for the Database, which provides secure DB connectivity. For Configuration details, refer to the *Oracle Communications Network Analytics Data Director Installation Guide*.