Oracle® Communications

Network Analytics Suite Security Guide

Release 25.2.200
G48860-01
December 2025

Oracle Communications Network Analytics Suite Security Guide, Release 25.2.200

G48860-01

Contents

Preface

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
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	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 25.2.2xx.

Release 25.2.200 - G48860-01, December 2025

OCNADD

- Updated new services details in the OCNADD Overview section.
- Updated all context diagrams in the Context Diagram section.
- Added new data flow details in the External Data Flows section.
- Updated the entire section of TLS Configuration with new sub-sections and data flow details.
- Updated the Network Policies section with the updated custom template details.
- Updated steps for configuring SASL for Kafka in the Kafka Security Configuration section.
- Updated the steps for Renewing OCNADD Certificates.
- Added the steps for vCollector Security Configuration.

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 Network Analytics Suite Release Notes*
- *Oracle Communications Network Analytics Suite Licensing Information User Manual*
- *Oracle Communications Network Analytics Automated Testing Suite Guide*
- *Oracle Communications Network Analytics Suite Security Guide*
- *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Network Analytics Data Director User Guide*
- *Oracle Communications Network Analytics Data Director Outbound Interface Specification Guide*
- *Oracle Communications Network Analytics Data Director Benchmarking Guide*
- *Oracle Communications Network Analytics Data Director Diameter User Guide*
- *Oracle Communications Network Analytics Data Director vCollector Installation Guide*
- *Oracle Communications Network Analytics Data Director Troubleshooting Guide*
- *Oracle Communications Cloud Native Core, cnDBTier Installation,Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, OCI Deployment Guide*
- *Oracle Communication Certificate Manager Installation, Upgrade and Fault Recovery Guide*
- *Oracle Communication Certificate Manager 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	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 (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.
vCollector Service	The service enables the acquisition of Diameter traffic from various network nodes, such as Diameter Signaling Routers (DSR) or other Diameter-based applications. Packet capture is achieved via port mirroring.
Diameter Aggregation Service	The service collects and aggregates network traffic from multiple Diameter Nodes (e.g., cnDRA, PCF, PMF, 3rd-party Diameter Nodes, etc.).
Diameter Correlation Service	The service correlates Diameter messages of a network scenario that can be represented by a transaction, call, or session and generates xDRs.
Gateway Service	The service is designed to manage external communication beyond the cluster boundaries.

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.

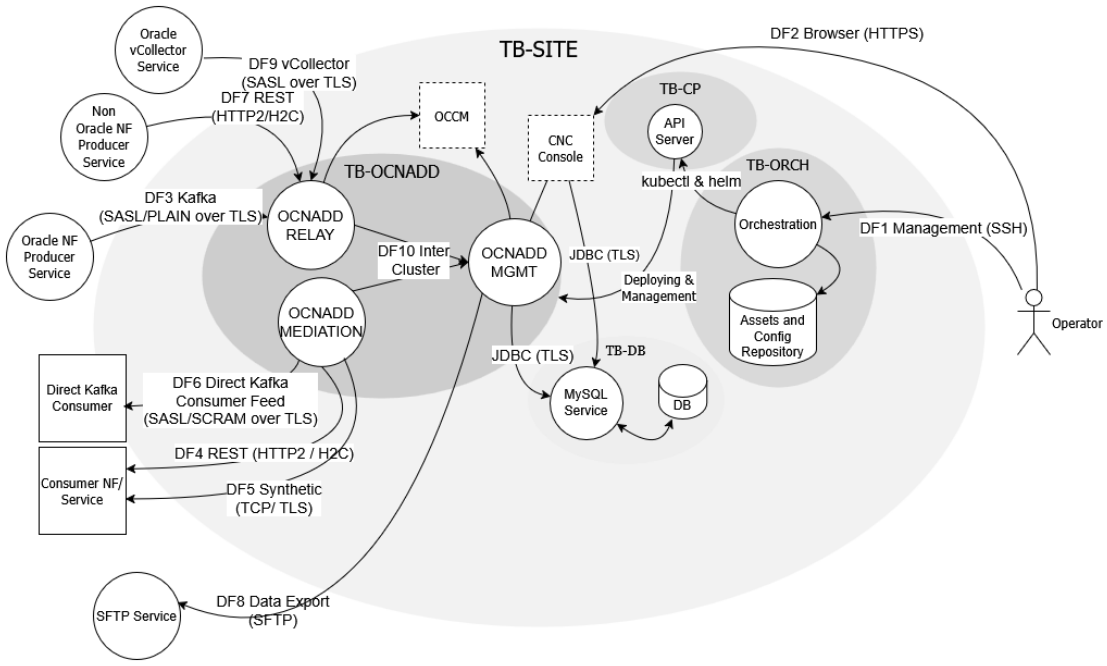
3.2 Trust Model for OCNADD

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

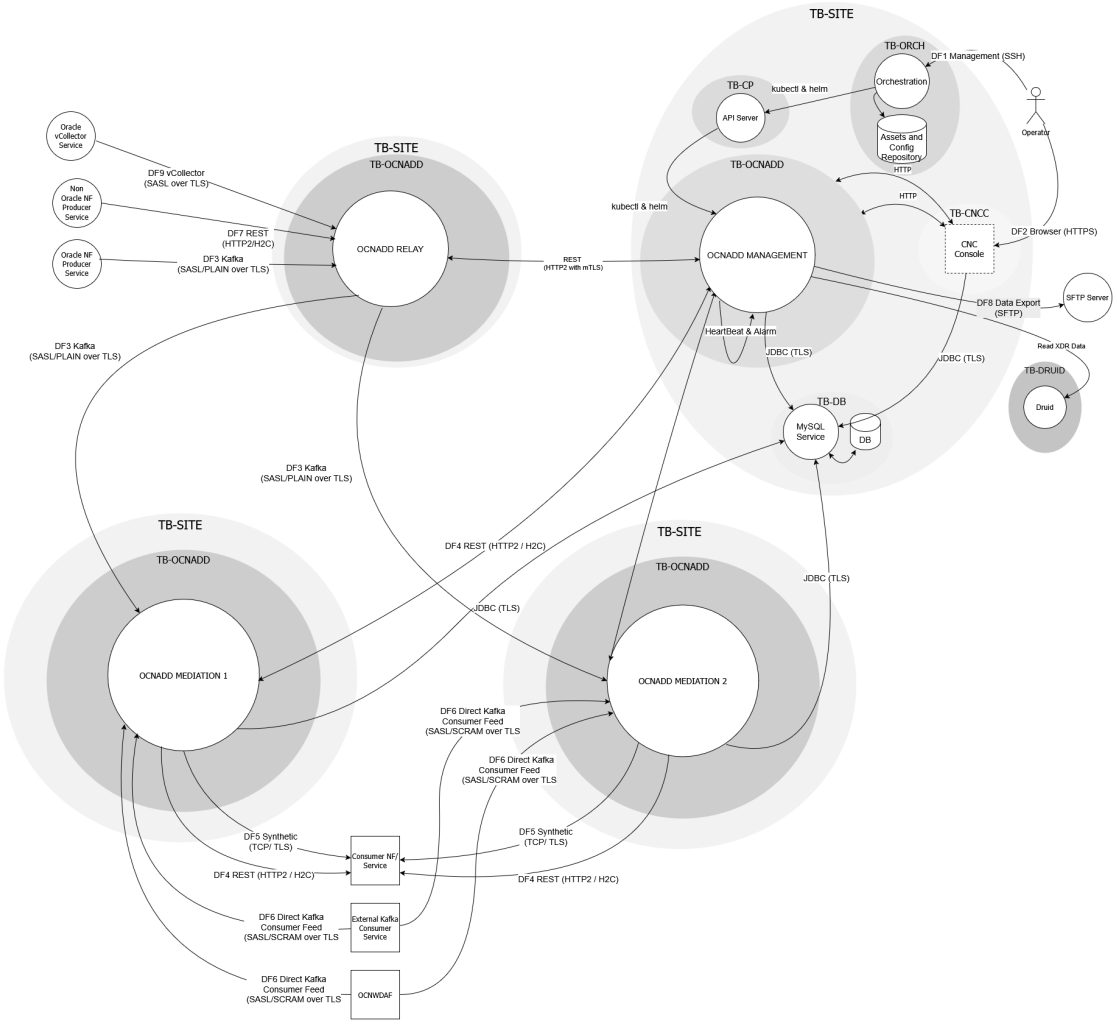
Single Cluster Deployment

Figure 3-1 Single Cluster Deployment



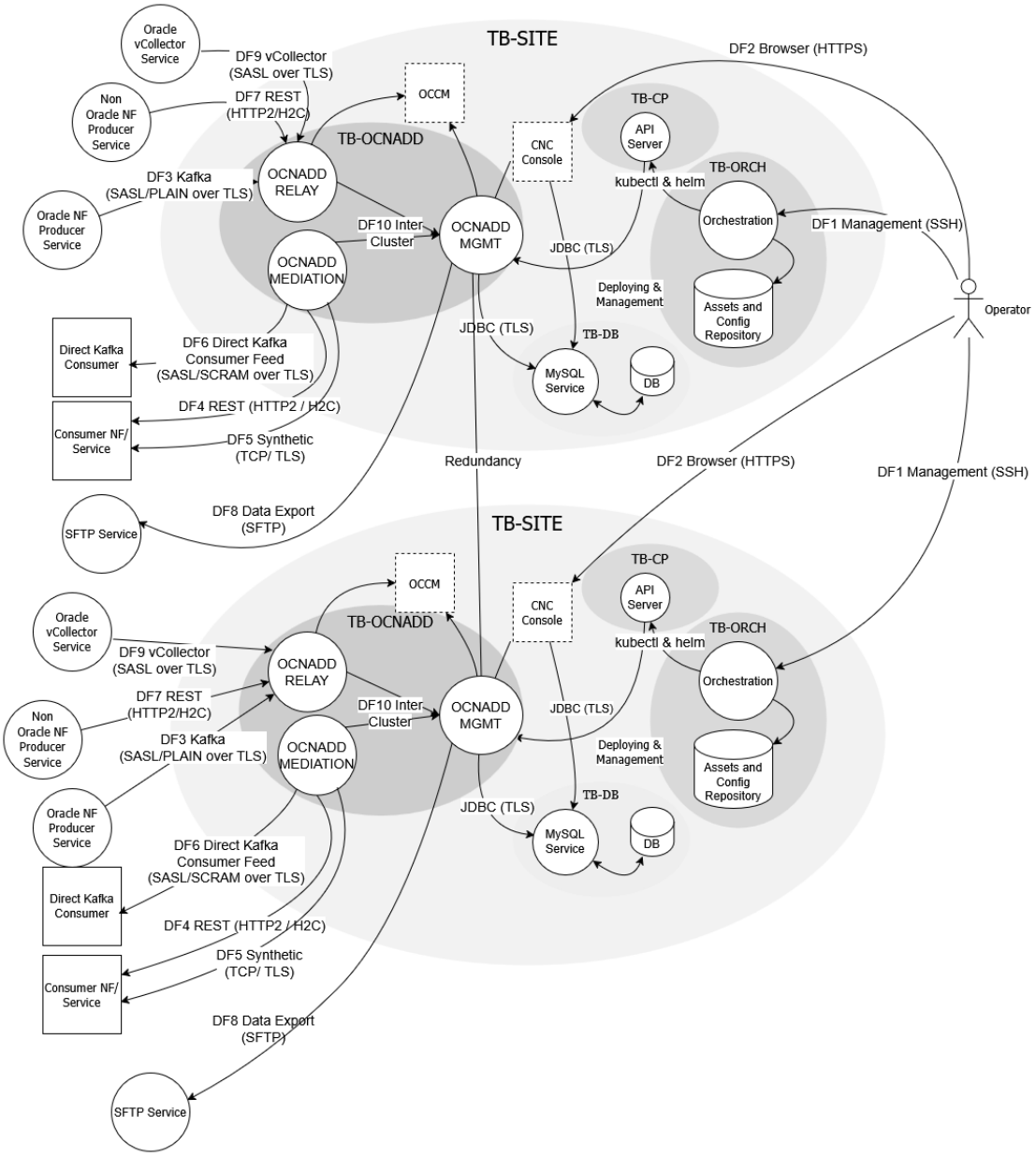
Multi-Cluster Deployment

Figure 3-2 Multi-Cluster Deployment



Two-Site Redundancy

Figure 3-3 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.
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).

Table 3-2 (Cont.) External Data Flows

Data Flow	Protocol	Description
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).
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.

Table 3-2 (Cont.) External Data Flows

Data Flow	Protocol	Description
DF9: vCollector	SASL_SSL	The vCollector receives Diameter traffic flow configurations from the OCNADD Configuration Service via secure, authenticated channels, then captures mirrored Diameter traffic from network nodes, converts it to JSON, and securely forwards it to the Kafka cluster of the Relay Agent group. Configuration and Kafka connection details are provided through secure notifications, ensuring end-to-end encryption and authentication. The communications are encrypted using TLS, and Oracle NF will authenticate themselves to Kafka through SASL/PLAIN or SASL/SCRAM (username and password).
DF10: Inter Cluster	HTTP2	Data Director components (Management, Relay Agent, and Mediation) can be deployed into different clusters. This communication is secured using HTTP/2 (over TLS).

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

4.1.1.1 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. See Certificate and Secret Generation section for more info about certificate creation.

DF7: Ingress Adapter feed communication is over TLS only. No additional steps apart from certificate creation need to be performed.

DF9: vCollector communications are encrypted using TLS and authenticate to Kafka through SASL/PLAIN or SASL/SCRAM. For more information, refer to the vCollector Installation Guide.

DF10: Data Director components—Management, Relay Agent, and Mediation—can be deployed across different clusters, with secure inter-cluster communication established using HTTP/2 over TLS.

See Certificate Creation for NFs and Third Party Consumers section before creating certificates for NFs, OCNADD, third party consumers, and Kafka Consumers.

4.1.1.2 Mutual TLS Communication

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

4.1.1.2.1 MTLS 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 Group Services

Management Group Services
ocnadduirouter
ocnaddalarm
ocnaddconfiguration
ocnaddhealthmonitoring
ocnaddbackuprestore
ocnaddredundancyagent
ocnaddexportservice
ocnaddmanagementgateway

Table 4-2 Relay Agent Services

Relay Agent Services
kafka-broker
kraft-controller
ocnaddscpaggregation
ocnaddnrfaggregation
ocnaddpcfaggregation
ocnaddbsfaggregation
ocnaddseppaggregation
ocnaddnonoracleaggregation
ocnaddvcollectordiameteraggregation
ocnaddpreupgradeaclhelmhook
ocnaddrelayagentgateway

Table 4-3 Mediation Group Services

Mediation Group Services
ocnaddmediationgateway
kafka-broker
kraft-controller
adapter
ocnaddfilter
ocnaddingressadapter
ocnaddcorrelation
ocnaddstorageadapterservice
ocnadddiametercorrelation
ocnaddpreupgradeaclhelmhook

4.1.1.2.2 MTLS Disabled

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

Table 4-4 Management Group Services

Management Group Services
ocnaddbackuprestore
ocnaddredundancyagent
ocnaddmanagementgateway

Table 4-5 Relay Agent Services

Relay Agent Services
kafka-broker
kraft-controller
ocnaddrelayagentgateway

Table 4-6 Mediation Group Services

Mediation Group Services
ocnaddmediationgateway
kafka-broker
kraft-controller
adapter
ocnaddingressadapter

Note

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

4.1.1.3 Enabling or Disabling mTLS in OCNADD

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

> **Note**
>
> For details on adding entries for service certificates, see the Certificate and Secret Generation section.

4.1.1.3.1 Enable mTLS in OCNADD

1. **Enable mutual TLS in Helm Charts.**
   Change the value of `global.ssl.mTLS` to `true` if not already set in the `ocnadd-common-custom-values-25.2.200.yaml`. It is set to `true` by default.

   ```
   ssl:
     mTLS: true
   ```

2. **Upgrade OCNADD deployments to enable mTLS**
   For Management deployment upgrade:

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

   For example:

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

   Upgrade the Relay Agent deployment:

   ```
   helm upgrade -n <relay-namespace> -f <ocnadd-common-custom-template> -f
   <relay-custom-values> <relay-release-name> <release-chart>
   ```

   For example:

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

3. Then, upgrade each Mediation group deployment after the Management group and Relay Agent upgrade is complete, with the Consumer Adapter, Correlation, Storage Adapter, and Ingress Adapter upgrade flags set to `TRUE` in the configuration.

   ```
   helm upgrade -n <mediation-group-namespace> -f <mediation-group-custom-
   values> <mediation-group-release-name> <release-chart> --set
   global.ocnaddmediation.env.admin.OCNADD_ADAPTER_UPGRADE_ENABLE=true,global.
   ocnaddmediation.env.admin.OCNADD_CORR_UPGRADE_ENABLE=true,global.ocnaddmedi
   ation.env.admin.OCNADD_INGRESS_ADAPTER_UPGRADE_ENABLE=true,global.ocnaddmed
   iation.env.admin.OCNADD_STORAGE_ADAPTER_UPGRADE_ENABLE=true
   ```

   For example:

   ```
   helm upgrade -n ocnadd-deploy-med1 -f ocnadd-common-custom-
   values-25.2.200.yaml -f ocnadd-mediation-custom-values-25.2.200.yaml
   ocnadd-med ocnadd --set
   ```

```
global.ocnaddmediation.env.admin.OCNADD_ADAPTER_UPGRADE_ENABLE=true,global.
ocnaddmediation.env.admin.OCNADD_CORR_UPGRADE_ENABLE=true,global.ocnaddmedi
ation.env.admin.OCNADD_INGRESS_ADAPTER_UPGRADE_ENABLE=true,global.ocnaddmed
iation.env.admin.OCNADD_STORAGE_ADAPTER_UPGRADE_ENABLE=true
```

4.1.1.3.2 Disable mTLS in OCNADD

1. **Disable mutual TLS in Helm Charts.**
   Change the value of `global.ssl.mTLS` to `false` if not already set in the `ocnadd-common-custom-values-25.2.200.yaml`. It is set to `true` by default.

   ```
   ssl:
     mTLS: false
   ```

2. **Upgrade OCNADD deployments to disable mTLS**
   For Management deployment upgrade:

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

   For example:

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

   Upgrade the Relay Agent deployment:

   ```
   helm upgrade -n <relay-namespace> -f <ocnadd-common-custom-template> -f
   <relay-custom-values> <relay-release-name> <release-chart>
   ```

   For example:

   ```
   helm upgrade -n ocnadd-deploy-relay -f ocnadd-common-custom-values.yaml -f
   ocnadd-relayagent-custom-values.yaml ocnadd-relay ocnadd
   ```

3. Then, upgrade each mediation deployment after management group and relay agent upgrade is complete, with the Consumer Adapter, Correlation, Storage Adapter, and Ingress Adapter upgrade flags set to `TRUE` in the configuration.

   ```
   helm upgrade -n <mediation-group-namespace> -f <mediation-group-custom-
   values> <mediation-group-release-name> <release-chart> --set
   global.ocnaddmediation.env.admin.OCNADD_ADAPTER_UPGRADE_ENABLE=true,global.
   ocnaddmediation.env.admin.OCNADD_CORR_UPGRADE_ENABLE=true,global.ocnaddmedi
   ation.env.admin.OCNADD_INGRESS_ADAPTER_UPGRADE_ENABLE=true,global.ocnaddmed
   iation.env.admin.OCNADD_STORAGE_ADAPTER_UPGRADE_ENABLE=true
   ```

   For example:

   ```
   helm upgrade -n ocnadd-deploy-med1 -f ocnadd-common-custom-
   values-25.2.200.yaml -f ocnadd-mediation-custom-values-25.2.200.yaml
   ocnadd-med ocnadd --set
   global.ocnaddmediation.env.admin.OCNADD_ADAPTER_UPGRADE_ENABLE=true,global.
   ```

```
ocnaddmediation.env.admin.OCNADD_CORR_UPGRADE_ENABLE=true,global.ocnaddmedi
ation.env.admin.OCNADD_INGRESS_ADAPTER_UPGRADE_ENABLE=true,global.ocnaddmed
iation.env.admin.OCNADD_STORAGE_ADAPTER_UPGRADE_ENABLE=true
```

> **Note**
>
> - External Kafka feed with filter and correlation features can not be used without enabling mutual TLS.
> - The Export feature will not work as it is dependent on the Correlation feeds, which requires mTLS to be enabled.
> - If there are multiple mediation groups follow the similar steps for upgrade.

4.1.1.4 Certificate and Secret Generation

OCNADD offers certificate creation with three options:

1. **Single Certs for each Component:** Each major component (management, relay agent, mediation) will receive one certificate for all the services in that component.
2. **Certificate for all Services:** Each service within the management, relay agent, and mediation components gets its own distinct certificate.
3. **TLS Enabled for External Interfaces:** Only certificates for services involved in external communication are created; all internal service communication is left unencrypted (no TLS).

> **Note**
>
> Certificate for all Services option is the most secure mode and is recommended.

4.1.1.4.1 Certificates Generated Using OCNADD Script

The OCNADD services can communicate with each other, as well as with external interfaces, in both secure (encrypted) and insecure modes. For establishing encrypted communication between the services, it is necessary to generate TLS certificates and private keys for each microservice. The service certificate is generated using the provided CA certificate see section "Configuring SSL or TLS Certificates".

The generated service certificates are stored as Kubernetes secrets.

Below are sample values files for the different modes of Data Director deployment:

The customer should update the `global` section as per the requirements. The namespace and the domain name will be handled by the script itself.

Management Group Deployment

mTLS mode is enabled (All services have separate certificates)

```
global.certificates.singlecert: false
```

Refer to file: `ssl_certs/default_values/management_service_values_template`

mTLS mode is enabled (single cert is selected)

```
global.certificates.singlecert: true
```

Refer to file: `ssl_certs/default_values/single_management_service_values_template`

mTLS mode is disabled (Only services indicated below should use the TLS certificates)

Refer to file: `ssl_certs/default_values/no_tls_management_service_values_template`

> **Note**
>
> - External Kafka feed with filter and correlation features cannot be used without enabling mTLS, e.g., **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 mTLS to be enabled.

Relay Agent Deployment Mode

mTLS mode is enabled (All services have separate certificates)

```
global.certificates.singlecert: false
```

Refer to file: `ssl_certs/default_values/relay_agent_service_values_template`

mTLS mode is enabled (single cert is selected)

```
global.certificates.singlecert: true
```

Refer to file: `ssl_certs/default_values/single_relay_agent_service_values_template`

> **Note**
>
> - External Kafka feed with filter and correlation features cannot be used without enabling internal TLS, e.g., **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.

mTLS mode is disabled

Refer to file:

`ssl_certs/default_values/no_tls_relay_agent_service_values_template`

> **Note**
>
> - External Kafka feed with filter and correlation features cannot be used without enabling internal TLS, for example, **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.

Mediation Group Deployment Mode (Multiple Mediation Group)

mTLS mode is enabled (All services have separate certificates)

```
global.certificates.singlecert: false
```

Refer to file: `ssl_certs/default_values/mediation_service_values_template`

mTLS mode is enabled (single cert is selected)

```
global.certificates.singlecert: true
```

Refer to file: `ssl_certs/default_values/single_mediation_service_values_template`

mTLS mode is disabled

Refer to file: `ssl_certs/default_values/no_tls_mediation_service_values_template`

Below is the service that needs to be disabled in `ocnadd-mediation-custom-values-25.2.200.yaml`:

```
ocnaddfilter:
    enabled: false
```

> **Note**
>
> - External Kafka feed with filter and correlation features cannot be used without enabling internal TLS, for example, **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.

ACL Configuration

To enable ACL change the value of `global.acl.kafkaClientAuth` to `required` and `global.acl.aclNotAllowed` as needed in `ocnadd-common-custom-values-25.2.200.yaml`:

```
global:
    acl:
        kafkaClientAuth: required
        aclNotAllowed: false
```

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

4.1.1.4.1.1 Configuring SSL or TLS Certificates

Data Director supports both secure and non-secure intra-service communication. To enable secure communication, turn on the mTLS flag in the Data Director Helm charts. When mTLS is enabled, each Data Director microservice must use its own TLS certificate. If mTLS is disabled, certain microservices or jobs still require a TLS certificate. For detailed guidance on TLS setup and requirements, see TLS Configuration.

To generate certificates using script, follow these steps:

1. In the extracted folder of OCNADD, go into the ssl_certs folder.

> **Note**
>
> Before configuring the SSL/TLS certificates, see the Customizing CSR and Certificate Extensions section in the *Oracle Communications Network Analytics Suite Security Guide*. This is a mandatory procedure and must be completed before proceeding with the installation.

2. Before generating certificates using `cacert` and `cakey`, the Kafka access mode needs to be finalized, and accordingly, provide the script response "y" in step 7 of Generate Certificates using CACert and CAKey section while running the `generate_certs` script for creating certificates.

> **Note**
>
> For changes required related to SAN Entries, see "Kafka Cluster Management Procedures " section in the *Oracle Communications Network Analytics Data Director User Guide*.

4.1.1.4.1.1.1 Generate Certificates using CACert and CAKey

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

Use the `ssl_certs` folder to generate the certificates for the Management Group, Relay Agent, and Mediation Group namespaces accordingly.

To generate certificates using CACert and CAKey:

1. Navigate to the <`ssl_certs`> folder.

> **Note**
>
> **(Optional)** Modify the `service_values_template` file to add or remove specific service blocks for which the certificate needs to be created or removed.
>
> - For example, to generate certificates for the management group, edit the *management_service_values_template* file.
> - Depending on the deployment group type, edit the respective template file for that group.

```
Global Params:

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

Root CA common name (e.g. rootca common_name=*.svc.domainName)
##root_ca

commonName=*.svc.domainName

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

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

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

2. Run the `generate_certs.sh` script:

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

   Where `<path to>` is the folder path containing the CACert and CAKey.

> **Note**
>
> If certificates for the relay agent and mediation group are generated separately, ensure that the same CA certificate and private keys used for the management group are used again:
>
> ```
> ./generate_certs.sh -cacert <path to>/cacert.pem -cakey <path to>/
> private/cakey.pem
> ```

3. Select the certificate generation method:

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------------
Options
Explanation:


1. Single Certs for each component              -> Each major component
(management, relay agent, mediation) will get one cert for all the
services in that component.
2. Certificate for all services                 -> Each service within
management, relay agent, and mediation components gets its own unique
certificate.
3. TLS enabled for External Interface Only      -> Only certificates for
services involved in external communication are created; all internal
service communication is left unencrypted (no TLS).
4. Simulator Certificates                       -> Generates certificates
intended for simulator deployments
only.

5. vCollector Certificates                      -> Generates certificates
intended for vcollector deployments
only.

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------------

Among the following certificate generation options:
(1) Single Certs for each component
(2) Certificate for all services
(3) Intra TLS Disabled
(4) Simulator Certificates
(5) vCollector Certificates
```

- To generate a single certificate per component:

  ```
  Select the option to generate certificates (1/2/3/4) : 1
  ```

- To generate certificates for all services:

  ```
  Select the option to generate certificates (1/2/3/4) : 2
  ```

4. Select the namespace for which the user wants to generate the certificates:

   ```
   Enter kubernetes namespace: <your_working_namespace>
   ```

5. Select the Data Director component to deploy:
   Example shown for Relay Agent:

   ```
   (1) Management
   (2) Relay Agent
   (3) Mediation
   Select the component to create certificates (1/2/3) : 2
   ```

6. Enter the domain name to change the default domain name (occne-ocdd):

   ```
   Please enter the domain name: <domain_name>
   ```

7. Enter SAN (DNS/IP entries) for any service if required:

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

   - If the user selects "y," a list of services will be shown and the user can add the SAN entries for any of the listed services by selecting the corresponding service number.
     - If option (1) Single Certs per component was selected:
       Example:

       ```
       For the following services:
       1. ocnadd

       Enter the number corresponding to the service for which you want to
       add IP: 1
       Please enter IP for the service ocnadd or enter "n" to exit :
       10.20.30.40
       Please enter IP for the service ocnadd or enter "n" to exit :
       10.20.30.41
       Please enter IP for the service ocnadd or enter "n" to exit :
       10.20.30.42
       Please enter IP for the service ocnadd or enter "n" to exit :
       10.20.30.43
       Please enter IP for the service ocnadd or enter "n" to exit : n
       ```

     - If option (2) Certificate for all services was selected:
       In the following example, the list of relay agent services is shown to the user to add SAN entries. Enter the number corresponding to the service for which the user wants to enter IP. After choosing the service, give IP addresses as input else enter "n" to exit.

Example list:

```
For the following services:
1. ocnaddrelayagentgateway
2. ocnaddbsfaggregation
3. ocnaddnrfaggregation
4. ocnaddscpaggregation
5. ocnaddpcfaggregation
6. ocnaddseppaggregation
7. ocnaddnonoracleaggregation
8. ocnaddvcollectordiameteraggregation
9. kafka-broker
10. kraft-controller
11. ocnaddpreupgradeaclhelmhook

Enter the number corresponding to the service for which you want to
add IP: 9
Please enter IP for the service kafka-broker or enter "n" to exit :
10.20.30.40
Please enter IP for the service kafka-broker or enter "n" to exit :
10.20.30.41
Please enter IP for the service kafka-broker or enter "n" to exit :
10.20.30.42
Please enter IP for the service kafka-broker or enter "n" to exit :
10.20.30.43
Please enter IP for the service kafka-broker or enter "n" to exit :
n
Do you want to add IP to any other service (y/n) : y
```

If user wants to add entries for other services then select y, and add the SAN entries.

8. Enter CA key passphrase when prompted:

   ```
   Enter passphrase for CA Key file: <passphrase>
   ```

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

   ```
   Create Certificate Signing Request (CSR) for each service? (y/n)
   ```

10. Sign CSR for each service with CA Key:

    ```
    Would you like to sign CSR for each service with CA key? (y/n)
    ```

11. After management certificates are generated, choose whether to continue with relay agent/ mediation group:

    ```
    Would you like to continue certificate creation for relay agent/mediation
    group? (y/n)
    ```

    - If **"y"**, the script repeats for relay agent and mediation group (beginning from step 4).
    - Select **"2/3"** in step 5 for relay agent/mediation group.
    - If **"n"**, the script finishes execution.

> **Note**
>
> The script can be used to create management certificates, relay agent certificates and the number of supported mediation group certificates in a single execution.

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

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

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

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

> **Note**
>
> If the Relay Agent and Mediation Group are located outside the cluster that hosts the Management service, copy the demoCA folder from the ssl_certs directory to the ssl_certs directory in the OCNADD package where you are installing the Relay Agent or Mediation Group.

4.1.1.4.1.1.2 Generate Certificate Signing Request (CSR)

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

Perform the following procedure to generate the CSR:

1. Navigate to the `<ssl_certs>/default_values` folder.
2. Copy the required `service_values_template` file (management_service_values_template, mediation_service_values_template, relay_agent_service_values_template, or simulator_values_template depending upon the mode of deployment) to another file named `backup_service_values_template`.
3. Edit the corresponding `service_values_template` file and update the global parameters, CN, and SAN (DNS/IP entries) for each service based on the requirements explained below.
4. Change the default domain name (`occne-ocdd`) and default namespace (`ocnadd-deploy`) in the corresponding `service_values_template` file with your cluster domain and namespace.
   Example commands:

   ```
   For management group: namespace = dd-mgmt-group clusterDomain =
   cluster.local.com sed -i "s/ocnadd-deploy/dd-mgmt-group/g"
   management_service_values_template sed -i "s/occne-ocdd/
   cluster.local.com/g" management_service_values_template

   For relay group: namespace = dd-relay-group clusterDomain =
   cluster.local.com sed -i "s/ocnadd-deploy/dd-relay-group/g"
   relay_agent_service_values_template sed -i "s/occne-ocdd/
   cluster.local.com/g" relay_agent_service_values_template

   For mediation group: namespace = dd-mediation-group clusterDomain =
   ```

```
cluster.local.com sed -i "s/ocnadd-deploy/dd-mediation-group/g"
mediation_service_values_template sed -i "s/occne-ocdd/
cluster.local.com/g" mediation_service_values_template
```

> **Note**
>
> Edit the corresponding service_values file for global parameters, RootCA common name, and Subject Alternative Name (SAN), keeping the service blocks of all the services for which the certificate needs to be generated.\
>
> ```
> Global Params:
>
> [global]
> countryName=<country>
> stateOrProvinceName=<state>
> localityName=<city>
> organizationName=<org_name>
> organizationalUnitName=<org_bu_name>
> defaultDays=<days to expiry>
>
>
> Root CA common name (e.g. rootca common_name=*.svc.domainName)
> ##root_ca
>
> commonName=*.svc.domainName
>
>
> Service common name for client and server and SAN(DNS/IP entries).
> (Make sure to follow exact same format and provide an empty line at
> the end of each service block)
>
> [service-name-1]
> server.commonName=server.cn.name.svc1
> IP.1=127.0.0.1
> DNS.1=localhost
>
> [service-name-2]
> server.commonName=server.cn.name.svc2
> IP.1= 10.20.30.40
> DNS.1 = *.svc2.namespace.svc.domainName
> .
> .
> .
> ##end
> ```

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

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

6. Select the certificate generation option:

   ```
   ------------------------------------------------------------------------
   ------------------------------------------------------------------------
   ```

```
-----------------------------------------------
Options
Explanation:


1. Single Certs for each component              -> Each major component
(management, relay agent, mediation) will get one cert for all the
services in that component.
2. Certificate for all services                 -> Each service within
management, relay agent, and mediation components gets its own unique
certificate.
3. TLS enabled for External Interface Only      -> Only certificates for
services involved in external communication are created; all internal
service communication is left unencrypted (no TLS).
4. Simulator Certificates                       -> Generates certificates
intended for simulator deployments
only.

5. vCollector Certificates                      -> Generates certificates
intended for vcollector deployments
only.

-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------------------------------------

Among the following certificate generation options:
(1) Single Certs for each component
(2) Certificate for all services
(3) Intra TLS Disabled
(4) Simulator Certificates
(5) vCollector Certificates
Select the option to generate certificates (1/2/3/4/5) : 2
```

7. Select the Data Director component to deploy:

```
(1) Management
(2) Relay Agent
(3) Mediation
Select the component to create certificates (1/2/3) : 1
```

8. Select the namespace where you would like to generate the certificates:

```
Enter kubernetes namespace: <your_working_namespace>
```

9. Once the service CSRs are generated the demoCA folder will be created. Navigate to CSR and keys in the `demoCA/dd_mgmt_relayagent_mediation_services/<your_namespace>/services`. The CSR can be signed using your own certificate signing mechanism to generate the certificates.
10. Make sure that the certificates and key names are created in the following format based on the service. For example, `<servicename>-servercert.pem and <servicename>-serverprivatekey.pem`
11. Once above certificates are generated by signing CSR with the Certificate Authority, copy those certificates into the respective *demoCA/dd_mgmt_relayagent_mediation_services/<your_namespace>/services* folder of each service.

> **Note**
>
> - Make sure to use the same CA key for management group, relay agent group and mediation group(s)
> - Make sure the certificates are copied into the respective folders for the server based on their generated CSRs

12. Run `generate_certs.sh` again with the CACert and the `--gensecret` or `-gs` flag to generate Kubernetes secrets:

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

13. Select the namespace where you would like to generate the certificates:

    ```
    Enter kubernetes namespace: <your_working_namespace>
    ```

14. Select “y” when prompted to generate secrets for the services:

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

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

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

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

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

17. Remove the modified `service_values_template` file after generating secrets. Rename `backup_service_values_template` back to the original `service_values_template` name.

4.1.1.4.2 Certificates produced by OCCM

OCCM Prerequisites for Installing OCNADD

Ensure the following prerequisites are satisfied by Oracle Communications Certificate Manager (OCCM):

1. **OCCM must be installed** and must have the necessary permissions to create secrets in the OCNADD Management, Relay Agent, and Mediation group namespaces.
   If OCCM is deployed in a separate namespace, it must have sufficient privileges to create secrets in these OCNADD namespaces.

   For more information, see “OCCM Deployment Models” in the *Oracle Communications Certificate Manager Installation, Upgrade and Fault Recovery Guide*.
2. **An Issuer (CA) must be configured in OCCM.** If multiple OCCMs are used, each must have at least one common issuer (CA) configured.
3. **OCCM must have sufficient capacity to generate the required number of certificates.** If all OCNADD features are enabled and the *single cert option is false*, then:
   - 9 certificates are required for Management services

- 12 certificates for Relay Agent services
- 11 certificates for **each** Mediation group

4. The `cncc-api-access` client must be enabled in the Oracle Communications Cloud Native Configuration Console (CNCC) UI. For details, see “Generate Access Tokens” in the *Oracle Communications Cloud Native Configuration Console User Guide*.

Create Secrets

Three secrets must be created in every OCNADD namespace to allow OCCM to create certificates.

1. `occm_secret`: This secret contains the username and password of a CNCC user with **OCCM_READ** and **OCCM_WRITE** roles. OCNADD uses this secret to communicate with OCCM through OCCNC.

```
kubectl create secret generic -n <ocnadd-namespace> \
  --from-literal=username=<cncc-user> \
  --from-literal=password=<cncc-password> \
  occm-secret
```

Where:

- `<ocnadd-namespace>` = OCNADD management, relay agent group, or mediation group namespace
- `<cncc-user>` = CNCC username
- `<cncc-password>` = CNCC password
- `occm-secret` = Name of the secret storing CNCC credentials

Example (management, relay agent, mediation):

```
kubectl create secret generic -n ocnadd-deploy-mgmt \
  --from-literal=username=occm-cncc \
  --from-literal=password=occm-cncc-secret \
  occm-secret

kubectl create secret generic -n ocnadd-deploy-relay \
  --from-literal=username=occm-cncc \
  --from-literal=password=occm-cncc-secret \
  occm-secret

kubectl create secret generic -n ocnadd-deploy-mediation \
  --from-literal=username=occm-cncc \
  --from-literal=password=occm-cncc-secret \
  occm-secret
```

2. `truststore_keystore_secret`: This secret contains the key used to encrypt the keystore and truststore created by OCNADD to store the service certificates and CA certificates.
Where:

- `<ocnadd-namespace>` = OCNADD namespace
- `<keystore-key>` = Password securing the service keystore
- `<truststore-key>` = Password securing the CA truststore
- `occm-truststore-keystore-secret` = Secret name

Example (management, relay agent, mediation):

```
kubectl create secret generic -n ocnadd-deploy-mgmt \
  --from-literal=keystorekey=keystorepassword \
  --from-literal=truststorekey=truststorepassword \
  occm-truststore-keystore-secret

kubectl create secret generic -n ocnadd-deploy-relay \
  --from-literal=keystorekey=keystorepassword \
  --from-literal=truststorekey=truststorepassword \
  occm-truststore-keystore-secret

kubectl create secret generic -n ocnadd-deploy-mediation \
  --from-literal=keystorekey=keystorepassword \
  --from-literal=truststorekey=truststorepassword \
  occm-truststore-keystore-secret
```

3. `occm_cacert`: This secret stores the CA certificate or certificate chain of the Issuer configured in OCCM.

```
kubectl create secret generic -n <ocnadd-namespace> \
  --from-file=cacert.pem=<ca-cert-file>.pem \
  occm-ca-secret
```

Where:

- `<ocnadd-namespace>` = OCNADD namespace
- `<ca-cert-file>` = PEM file containing CA certificate or chain
- `occm-ca-secret` = Secret storing CA cert(s)

Example (management, relay agent, mediation):

```
kubectl create secret generic -n ocnadd-deploy-mgmt \
  --from-file=cacert.pem=<ca-cert-file>.pem \
  occm-ca-secret

kubectl create secret generic -n ocnadd-deploy-relay \
  --from-file=cacert.pem=<ca-cert-file>.pem \
  occm-ca-secret

kubectl create secret generic -n ocnadd-deploy-mediation \
  --from-file=cacert.pem=<ca-cert-file>.pem \
  occm-ca-secret
```

OCCM Certificate Handling

Oracle Communications Certificate Management (OCCM) is an automated solution that manages the certificates required for Oracle Communications Network Analytics Data Director (OCNADD). OCCM continuously monitors and renews certificates based on expiration and validity periods.

The certificates generated by OCCM already include all required TLS/mTLS configurations, eliminating the need for additional TLS setup.

Steps to Generate Certificates Using OCCM

1. Create secrets as outlined in the Create Secrets section above.
2. Modify the `custom_values.yaml` file as described in “Helm Parameter Configuration for OCCM” in the *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.
3. Run the Helm install command by following the “Installing OCNADD Package” instructions in the *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.

4.1.1.4.3 OCCM Certificates for Apache Druid

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

4.1.1.5 Customizing CSR and Certificate Extensions

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

Modify the files based on requirements. For more information, see OpenSSL x509 v3 Certificate and CSR Configuration.

4.1.1.6 Key Usage Requirement for Client and Server CSR

Certificates Generated Using OCNADD Script

External CAs may require specific key usage and extended key usage to be included in the CSR. Add or modify the `[ req_ext ]` section in both `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 a reference to `req_ext` in the `[ 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.

Key Encipherment for Certificates Generated Using OCCM

In the certificate signing request (CSR) used 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.

4.1.1.7 Adding or Updating SAN Entries

Updating SAN Entries Using Script

SAN entries for OCNADD services can be updated by modifying the files referenced inside the **ssl_certs/default_values** folder. Depending on the component (Management, Relay Agent, or Mediation), use the respective template file.

Below is an example of a SAN entry update for the Redundancy Agent inside the Management Group:

1. Navigate to the `ssl_certs` folder of the target release.
2. Edit the `management_service_values` file located at `demoCA/dd_mgmt_relayagent_mediation_services/<management_group_namespace>` to add the IP address as a SAN entry in the Redundancy Agent service section.

```
...
...
[ocnaddredundancyagent]
server.commonName=ocnaddredundancyagent
DNS.1=*.ocnaddredundancyagent.ocnadd-deploy.svc.occne-ocdd
DNS.2=ocnaddredundancyagent
DNS.3=ocnaddredundancyagent.ocnadd-deploy
IP.1=124.x.x.1    <Loadbalancer IP of Redundancy Agent>     ## Add IP.1

##end
```

3. Follow the procedure Renewing OCNADD Certificates to update the SAN entries for the Redundancy Agent. Note that this section is generic; update the files based on the specific Redundancy Agent requirements.

SAN Entries Update for OCCM

To update or add Subject Alternative Name (SAN) entries for Kafka, the Redundancy Agent, or the Ingress Adapter, modify the respective `custom_values.yaml` file with the necessary SAN entries by referring to the "Adding/Updating Loadbalancer IPs in SAN when OCCM is used" section of the *Oracle Communications Network Analytics Data Director Installation, Upgrade and Fault Recovery Guide*. The certificates generated by OCCM will then incorporate the updated SAN configuration.

Below is an example from `custom-templates/ocnadd-relayagent-custom-values.yaml`:

```
san:
    kafka:
        update_required: false
        # Provide Kafka Load balancer IPs here
        ips:
            [
```

```
                "10.10.10.10",
                "10.10.10.11",
                "10.10.10.12",
                "10.10.10.13"
            ]
        uuid:
            server:
    relay_gateway:
        update_required: false
        # Provide Kafka Load balancer IPs here
        ips:
            ["10.10.10.10"]
        uuid:
            server:
    vcollector:
        enabled: false
        update_required: false
        # Provide Kafka Load balancer IPs here
        ips:
            ["10.10.10.10"]
        uuid:
            server:
```

4.1.2 Network Policies

OCNADD is shipped with Network Policies for ingress connections, which are disabled by default. Customers can configure the intended sources of connections by modifying the `ocnadd-management-custom-values.yaml`, `ocnadd-relayagent-custom-values.yaml`, and `ocnadd-mediation-custom-values.yaml` files as described below.

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

1. **Internal Service Connections**: Connections between services internal to the OCNADD namespace or internal to the OCNADD management, relay agent, or mediation group namespaces.
2. **External Service Connections**: Services from sources external to the OCNADD namespace, or external to the OCNADD management, relay agent, or mediation group namespaces. External connections can be further categorized into:
   - **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
   - **Connections from OCNADD when deployment is done on Multiple Clusters:** Management, Relay Agent, and Mediation are installed in separate clusters.

To enable network policies, change the value of `global..network.policy.enable` to `true` in the respective custom values file. By default, it is set to `false`. When enabled, network policies for Internal Service Connections are activated by default without requiring further modifications.

Where <deployment-type> and the respective custom values file are as follows:

- <deployment-type> is "ocnaddmanagement" and custom values file is ocnadd-management-custom-values.yaml
- <deployment-type> is "ocnaddrelayagent" and custom values file is ocnadd-relayagent-custom-values.yaml
- <deployment-type> is "ocnaddmediation" and custom values file is ocnadd-mediation-custom-values.yaml

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

Include the following in the ocnadd-management-custom-values.yaml file for the management group chart to allow connections from Worker Groups (including relay agents and mediation groups):

```
global:
  ocnaddmanagement:
    network:
      ingress:
        namespaces:
          # allow ingress network connections from below relay agents and
mediation groups
          ocnaddgroup:
            - ocnadd-relay
            - ocnadd-mediation
```

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

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

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

```
global:
  ocnaddrelayagent:
    network:
      ingress:
        namespaces:
```

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

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

Where `<deployment-type>` and the respective custom values file are as follows:

- `<deployment-type>` is "`ocnaddmanagement`" and custom values file is `ocnadd-management-custom-values.yaml`
- `<deployment-type>` is "`ocnaddrelayagent`" and custom values file is `ocnadd-relayagent-custom-values.yaml`
- `<deployment-type>` is "`ocnaddmediation`" and custom values file is `ocnadd-mediation-custom-values.yaml`

Below is an example for management group changes in `ocnadd-management-custom-values.yaml`:

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

Similarly, `ocnadd-relayagent-custom-values.yaml` and `ocnadd-mediation-custom-values.yaml` files can be updated.

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

> **Note**
>
> Allow connection from Druid IPs to Kafka if Druid is configured to ingest from Kafka.

```
global:
  ocnaddrelayagent:
    network:
      # Allow external network connections to kafka from below IPs/CIDRs/
Network
```

```
      # needed for external kafka consumer, redundancy agent, ingress adapter
and druid
      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
```

To enable connections from a relay agent in a different Kubernetes cluster (or from a specific IP/network), set `global.ocnaddmediation.network.ingress.external.enable` to `true` and specify the desired IP or network address in CIDR format under `global.ocnaddmediation.network.ingress.external.cidrs.kafka` in the `ocnadd-mediation-custom-values.yaml` file.

> **Note**
>
> Allow connection from Druid IPs to Kafka if Druid is configured to ingest from Kafka.

```
global:
  ocnaddmediation:
    network:
      # Allow external network connections to kafka from below IPs/CIDRs/
Network
      external:
        enable: true                    # Changed to true
        cidrs:
          kafka:
            - 10.10.10.10/8             # 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 the redundancy agent is enabled, communication from the peer redundancy agent in the primary/secondary site must be allowed. Update `global.ocnaddmanagement.network.ingress.external.enable` to `true` and add the peer agent's Loadbalancer IP address under `global.ocnaddmanagement.network.ingress.external.cidrs.agent` in the `ocnadd-management-custom-values.yaml`.

```
global:
  ocnaddmanagement:
    network:
      # Allow external network connections to kafka from below IPs/CIDRs/
Network
      # needed for external gateway and redundancy agent
      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

1. Install OCNADD.
2. Modify the `ocnadd-management-custom-values.yaml`, `ocnadd-relayagent-custom-values.yaml`, and `ocnadd-mediation-custom-values.yaml` files as needed:
   - Modify connections for NF namespaces or external IP/CIDRs independently for the relay agent and each mediation group in their respective custom values file.
3. Perform Helm upgrade of the OCNADD deployment:
   - The specific steps for the Helm upgrade will vary based on the selected OCNADD deployment model. See the *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide* for detailed instructions.
   - Start by performing the Helm upgrade for management group services. Subsequently, perform the Helm upgrade for each relay agent and mediation 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, between Kafka-client and Kafka-broker, and between Kafka and Kraft controller) happens with a common user (**ocnadd**) configured in `<chart-path>/charts/ocnaddrelayagent/charts/ocnaddkafka/config/kafka_server_jaas.conf` .

Customers can add more users in the same `kafka_server_jaas.conf` file.

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

Below are sample changes to add NF users in Kafka within the Relay Agent charts:

```
KafkaServer {
    org.apache.kafka.common.security.plain.PlainLoginModule required
        username="ocnadd"
        password="<change this>"
        user_ocnadd="<change this>"
        user_scpuser="scp"
        user_nrfuser="nrf"
        user_pcfuser="pcf"
        user_bsfuser="bsf"
        user_seppuser="sepp";

    org.apache.kafka.common.security.scram.ScramLoginModule required
        user_extuser1="extuser1"
        user_extuser2="extuser2";
};
```

In the above code snippet, `user_nrfuser="nrf"` is added for the `nrfuser` user, and `"nrf"` is the password. Similar rules apply for other NF users.

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.

> **Note**
>
> **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:

   > **Note**
   >
   > The renewal steps will work based on the services in each component. Below is an example for `management services certificate renewal`. (If the `ocnaddredundancy` service is enabled, keep it uncommented; otherwise, comment `ocnaddredundancy`.)

```
# This file contains 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=365

#kafka-broker
#kraft-controller
#ocnaddthirdpartyconsumer
#oraclenfproducer
ocnadduirouter
#ocnaddadminservice
ocnaddalarm
ocnaddconfiguration
ocnaddhealthmonitoring
#ocnaddscpaggregation
#ocnaddnrfaggregation
#ocnaddpcfaggregation
#ocnaddbsfaggregation
#ocnaddseppaggregation
#ocnaddnonoracleaggregation
#ocnaddvcollectordiameteraggregation
#vcollector
#adapter
#ocnaddcorrelation
#ocnadddiametercorrelation
#ocnaddingressadapter
#ocnaddfilter
ocnaddbackuprestore
ocnaddredundancyagent
#ocnaddstorageadapterservice
ocnaddexport
#ocnaddmediationgateway
```

```
ocnaddmanagementgateway
#ocnaddrelayagentgateway
```

Similar to the above example, keep relay agent and mediation group services uncommented while renewing certificates for services specific to relay agent or mediation group.

2. Run: `./generate_certs.sh -cacert <path to>/cacert.pem -cakey <path to>/cakey.pem --renew`
3. Choose the namespace for OCNADD (management group, relay agent, or mediation group services) when prompted: `Enter kubernetes namespace:`
4. Provide the password to your CA key when prompted: `Enter passphrase for CA Key file:`
5. Older certificates will be revoked, new 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_relayagent_mediation_services/<namespace>/services/<service-name>/server` for each `<service-name>` listed in the respective `default_values/values` file.
   Example: Navigating to the *server* folder for the `adapter` service in the mediation group namespace `ocnadd-deploy-med`:

   ```
   [ssl_certs]$ ls
   default_values demoCA generate_certs.sh generate_secrets.sh logs template

   [ssl_certs]$ cd demoCA/dd_mgmt_relayagent_mediation_services/ocnadd-deploy-med/services/adapter/server/

   [server]$ ls
   adapter-servercert.pem adapter-server.csr adapter-serverprivatekey.pem adapter_server_ssl.cnf
   ```

2. Delete the old CSR request. **Example:** Deleting the old CSR request for the `adapter` service:

   ```
   [server]$ ls
   adapter-servercert.pem adapter-server.csr adapter-serverprivatekey.pem adapter_server_ssl.cnf

   [server]$ rm adapter-server.csr

   [server]$ ls
   adapter-servercert.pem adapter-serverprivatekey.pem adapter_server_ssl.cnf
   ```

3. To create a new CSR (`server.csr`) from the old certificate (`servercert.pem`) and private key (`serverprivatekey.pem`), use the following snippet:

   ```
   openssl x509 -x509toreq -in servercert.pem -signkey serverprivatekey.pem -out server.csr
   ```

Example: Creating a new server CSR for the adapter service:

```
[server]$ ls
adapter-servercert.pem adapter-server.csr adapter-serverprivatekey.pem
adapter_server_ssl.cnf

[server]$ openssl x509 -x509toreq -in adapter-servercert.pem -signkey
adapter-serverprivatekey.pem -out adapter-server.csr
Getting request Private Key
Generating certificate request

[server]$ ls
adapter-servercert.pem adapter-server.csr adapter-serverprivatekey.pem
adapter_server_ssl.cnf
```

4. Repeat steps **1 through 3** for all other services.
5. Create new certificates from the newly created CSRs and follow the Generate Certificate Signing Request (CSR) procedure starting from step 9.
6. Run the following command to perform a rollout restart of all OCNADD services:

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

   Where `<ocnadd-deploy>` is the namespace where OCNADD management group services, Relay Agent, or Mediation group services are deployed.

   Begin with the rollout restart of the **management namespace** (only for deployments), followed by the **relay agent namespace**, then each **mediation group deployment** individually. Finally, restart each relay agent and mediation group **stateful set** one by one.

   Consider whether this process will impact traffic.

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 `ocnadd-common-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 privilleges
- <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_user> is the role created for OCNADD in Druid

4.1.10 vCollector Security Configuration

Configuration and Kafka connection details are provided through secure notifications, ensuring end-to-end encryption and authentication. The communications are encrypted using TLS and authenticate themselves to Kafka through SASL/PLAIN or SASL/SCRAM (username and password).

Generate Certificates for vCollector

Generates secure, service-specific SSL certificates for vCollector using either a script with CA credentials or through OCCM-managed Kubernetes secrets, ensuring encrypted communication and proper authentication for vCollector services.

To generate certificates for vCollector, see "Generate Certificates" section of *Oracle Communication Network Analytics vCollector Installation Guide* for certificate creation.

vCollector Secret Management

The vCollector secret management process secures sensitive credentials, such as passwords and keystore information, by encrypting them using a dedicated key and initialization vector, and then storing them in an encrypted local database, ensuring controlled and protected access. This approach uses generated certificates and PKCS12 keystores, with secrets encrypted via Java utilities, to maintain robust security for vCollectorConfig operations.

To generate secrets for vCollector, see "*vCollector Secret Managament*" section of *Oracle Communication Network Analytics vCollector Installation Guide* for secret management.