Oracle® Communications

Network Analytics Data Director Troubleshooting Guide

Release 23.4.0.0.1
F86975-02
July 2024

ORACLE

Oracle Communications Network Analytics Data Director Troubleshooting Guide, Release 23.4.0.0.1

F86975-02

Contents

1 Introduction

1.1 Overview 1
1.2 References 1

2 Troubleshooting OCNADD

2.1 Generic Checklist 1
2.2 Helm Install and Upgrade Failure 12
2.2.1 Incorrect Image Name in ocnadd-custom-values.yaml File 12
2.2.2 Failed Helm Installation/Upgrade Due to Prometheus Rules Applying Failure 13
2.2.3 Docker Registry is Configured Incorrectly 13
2.2.4 Continuous Restart of Pods 14
2.2.5 Adapter Deployment Removed during Upgrade or Rollback 14
2.2.6 ocnadd-custom-values.yaml File Parse Failure 15
2.2.7 Kafka Brokers Continuously Restart after Reinstallation 15
2.2.8 Kafka Brokers Continuously Restart After the Disk is Full 16
2.2.9 Kafka Brokers Restart on Installation 17
2.2.10 Kafka Brokers in Crashloop State After Rollback 18
2.2.11 Database Goes into the Deadlock State 18
2.2.12 The Pending Rollback Issue Due to PreRollback Database Rollback Job Failure 19
2.2.13 Upgrade fails due to unsupported changes 22
2.2.14 Upgrade Failed from Source Release to Target Release Due to Helm Hook Upgrade Database Job 24
2.2.15 Upgrade fails due to Database MaxNoOfAttributes exhausted 24
2.2.16 Webhook Failure During Installation or Upgrade 25
2.2.17 Adapters Do Not Restart after Rollback 25
2.2.18 Adapters Do Not Restart after Rollback 26
2.2.19 Third-Party Endpoint DOWN State and New Feed Creation 26
2.2.20 Adapter Feed Not Coming Up After Rollback 27
2.2.21 Adapters pods are in the "init:CrashLoopBackOff" error state after rollback 27
2.2.22 Configuration Service Pod Goes into The "CrashLoopBackOff" State After Upgrade When the cnDBTier Version is 23.2.x or Above 28
2.2.23 Feed/Filter Configurations are Missing From Dashboard After Upgrade 28

2.2.24 Kafka Feed Cannot be Created in Dashboard After Upgrade 29
2.2.25 OCNADD Services Status Not Correct in Dashboard After Upgrade 29

3 Logs

3.1 Log Levels 1
3.2 Configuring Log Levels 1
3.3 Collecting Logs 2
3.4 Collect logs using Deployment Data Collector Tool 2

4 Capturing tcpdump from CNE

5 OCNADD Alerts

5.1 Configuring Alerts 1
5.2 Alert Forwarding Using Simple Network Management Protocol (SNMP) 1
5.3 List of Alerts 6
5.3.1 System Level Alerts 6
5.3.2 Application Level Alerts 12

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

Acronym	Description
ACL	Access Control List
CLI	Command Line Interface
CNE	Cloud Native Environment
KPI	Key Performance Indicator
MPS	Messages Per Second
NRF	Oracle Communications Cloud Native Core, Network Repository Function
OHC	Oracle Help Center
OSDC	Oracle Service Delivery Cloud
SCP	Oracle Communications Cloud Native Core, Service Communication Proxy
SEPP	Oracle Communications Cloud Native Core, Security Edge Protection Proxy
SVC	Services
URI	Uniform Resource Identifier

What's New in This Guide

This section lists the documentation updates for Release *23.4.x* in *Oracle Communications Network Analytics Data Director Troubleshooting Guide*.

Release 23.4.0.0.1 - F86975-02, July 2024

There are no updates to this document in this release.

Release 23.4.0 - F86975-01, December 2023

- Updated Generic Checklist:
  - Updated example command to Check if the microservices can access each other through a REST interface.
  - Updated the list of URIs for all the microservices.
  - Added a note in Logs Verification section.
  - Updated Scenario for Identifying the Network IP "Host" ACLs in Kafka Feed.
  - Updated the scenario where producer is unable to send traffic to OCNADD when an External Kafka Feed is enabled.
  - Updated the scenario where external Kafka consumer is unable to consume messages from OCNADD.
- Updated the following topics in the Helm Install and Upgrade Failure section:
  - Kafka Brokers Continuously Restart after Reinstallation
  - Kafka Brokers Continuously Restart After the Disk is Full
  - Kafka Brokers in Crashloop State After Rollback
  - The Pending Rollback Issue Due to PreRollback Database Rollback Job Failure
  - Upgrade fails due to unsupported changes
  - Upgrade Failed from Source Release to Target Release Due to Helm Hook Upgrade Database Job
  - Adapters Do Not Restart after Rollback
  - Adapter Feed Not Coming Up After Rollback
  - Configuration Service Pod Goes into The "CrashLoopBackOff" State After Upgrade When the cnDBTier Version is 23.2.x or Above
- Added the following topics in the Helm Install and Upgrade Failure section:
  - Failed Helm Installation/Upgrade Due to Prometheus Rules Applying Failure
  - Kafka Brokers in Crashloop State After Rollback
  - Adapters pods are in the "init:CrashLoopBackOff" error state after rollback
  - Feed/Filter Configurations are Missing From Dashboard After Upgrade
  - Kafka Feed Cannot be Created in Dashboard After Upgrade
  - OCNADD Services Status Not Correct in Dashboard After Upgrade

1 Introduction

This document provides Oracle Communications Network Analytics Data Director (OCNADD) troubleshooting information.

1.1 Overview

Oracle Communications Network Analytics Data Director (OCNADD) is a specialized Network Data Broker (NDB) that receives network data from various data sources (such as 5G NFs and Non-5G NFs) and sends the data securely to the subscribed consumers (such as third-party tools) after applying mechanisms such as data filtering, data replication, and data aggregation. All these mechanisms are configurable by the consumers.

OCNADD provides curated data (either filtered or replicated) for network analytics and monitoring. OCNADD supports robust, configurable filtering and aggregation options which enables the operator to sort data, create comprehensive dashboards, and generate Key Performance Indicators (KPIs) for all departments within the service provider framework. OCNADD also provides a GUI that enables users to create, edit, and delete data feed.

For more information about OCNADD architecture and features, see *Oracle Communications Network Analytics Data Director User Guide*.

1.2 References

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

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

2
Troubleshooting OCNADD

This chapter provides information to troubleshoot the common errors, which can be encountered during the preinstallation, installation, and upgrade procedures of OCNADD.

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

2.1 Generic Checklist

The following sections provide a generic checklist for troubleshooting OCNADD.

Deployment Checklist

Perform the following pre-deployment checks:

- Failure in Certificate or Secret generation.
  There may be a possibility of an error in certificate generation when the Country, State, or Organization name is different in CA and service certificates.

```
Error Code/Error Message:

  The countryName field is different between
  CA certificate (US) and the request (IN)

  (similar error message will be reported forState or Org name)
```

  To resolve this error:

  1. Navigate to "ssl_certs/default_values/" and edit the "values" file.
  2. Change the following values under "[global]" section:
     - countryName
     - stateOrProvinceName
     - organizationName
  3. Ensure the values match the CA configurations, for example:
     If the CA has country name as "US", state as "NY", and Org name as "ORACLE" then, set the values under [global] parameter as follows:

```
[global]
```

```
countryName=US
stateOrProvinceName=NY
localityName=BLR
organizationName=ORACLE
organizationalUnitName=CGBU
defaultDays=365
```

4. Rerun the script and verify the certificate and secret generation.

- Run the following command to verify if OCNADD deployment, pods, and services created are running and available:

  ```
  # kubectl -n <namespace> get deployments,pods,svc
  ```

  Verify the output, and check the following columns:

  – READY, STATUS, and RESTARTS
  – PORT(S) of service

  > **Note**
  >
  > It is normal to observe the Kafka broker restart during deployment.

- Verify if the correct image is used and correct environment variables are set in the deployment.
  To check, run the following command:

  ```
  # kubectl -n <namespace> get deployment <deployment-name> -o yaml
  ```

- Check if the microservices can access each other through a REST interface.
  To check, run the following command:

  ```
  # kubectl -n <namespace> exec <pod name> -- curl <uri>
  ```

  Example:

  ```
  kubectl exec -it pod/ocnaddconfiguration-6ffc75f956-wnvzx -n ocnadd-
  system  -- curl 'http://ocnaddadminservice:9181/ocnadd-admin-svc/v2/
  {workerGroup}/topic'
  ```

  > **Note**
  >
  > These commands are in their simple format and display the logs only if ocnaddconfiguration and ocnadd-admin-svc pods are deployed.

The list of URIs for all the microservices:

- http://ocnaddconfiguration:<port>/ocnadd-configuration/v1/subscription
- http://ocnaddalarm:<port>/ocnadd-alarm/v1/alarm?&startTime=<start-time>&endTime=<end-time>

  use off-set date time format: e.g 2022-07-12T05:37:26.954477600Z

- <ip>:<port>/ocnadd-admin-svc/v1/{workerGroup}/topic/
- <ip>:<port>/ocnadd-admin-svc/v1/{workerGroup}/describe/topic/<topicName>
- <ip>:<port>/ocnadd-admin-svc/v1/{workerGroup}/alter
- <ip>:<port>/ocnadd-admin-svc/v1/{workerGroup}/broker/expand/entry
- <ip>:<port>/ocnadd-admin-svc/v1/{workerGroup}/broker/health

Application Checklist

Logs Verification

> **Note**
>
> The below procedures should be verified or run corresponding to the applicable worker group or the management group.

Run the following command to check the application logs and look for exceptions:

```
# kubectl -n <namespace> logs -f <pod name>
```

Use the option `-f` to follow the logs or `grep` option to obtain a specific pattern in the log output.

Example:

```
# kubectl -n ocnadd-system  logs -f $(kubectl -n ocnadd-system get pods -o
name |cut -d'/' -f 2|grep nrfaggregation)
```

Above command displays the logs of the `ocnaddnrfaggregation` service.

Run the following command to search for a specific pattern in the log output:

```
# kubectl logs -n ocnadd-system <pod name> | grep <pattern>
```

> **Note**
>
> These commands are in their simple format and display the logs only if there is atleast one `nrfaggregation` pod deployed.

Kafka Consumer Rebalancing

The Kafka consumers can rebalance in the following scenarios:

- The number of partitions changes for any of the subscribed topics.
- A subscribed topic is created or deleted.
- An existing member of the consumer group is shutdown or fails.
- In the Kafka consumer application,
  1. Stream threads inside the consumer app skipped sending heartbeat to Kafka.
  2. The batch of messages took longer time to process and causes the time between the two polls to take longer.

3. Any stream thread in any of the consumer application pods dies because of some error and it is replaced with a new Kafka Stream thread.
4. Any stream thread is stuck and not processing any message.

- A new member is added to the consumer group (for example, new consumer pod spins up).

When the rebalancing is triggered, there is a possibility that offsets are not committed by the consumer threads as offsets are committed periodically. This can result in messages corresponding to non-committed offsets being sent again or duplicated when the rebalancing is completed and consumers started consuming again from the partitions. This is a normal behavior in the Kafka consumer application. However, because of frequent rebalancing in the Kafka consumer applications, the counts of messages in the Kafka consumer application and 3rd party application can mismatch.

Data Feed not accepting updated endpoint

Problem

If a Data feed is created for synthetic packets with an incorrect endpoint, updating the endpoint afterward has no effect.

Solution

Delete and recreate the data feed for synthetic packets with the correct endpoint.

Kafka Performance Impact (due to disk limitation)

Problem

When source topics (SCP, NRF, and SEPP) and MAIN topic are created with Replication Factor = 1

For a low performance disk, the Egress MPS rate drops/fluctuates with the following traces in the Kafka broker logs:

```
Shrinking ISR from 1001,1003,1002 to 1001. Leader: (highWatermark: 1326,
endOffset: 1327). Out of sync replicas: (brokerId: 1003, endOffset: 1326)
(brokerId: 1002, endOffset: 1326). (kafka.cluster.Partition)
ISR updated to 1001,1003 and version updated to 28(kafka.cluster.Partition)
```

Solution

The following steps can be performed (or verified) to optimize the Egress MPS rate:

1. Try to increase the disk performance in the cluster where OCNADD is deployed.
2. If the disk performance cannot be increased, then perform the following steps for OCNADD:
   a. Navigate to the Kafka helm charts values file (<helm-chart-path>/ocnadd/charts/ocnaddkafka/values.yaml)
   b. Change the below parameter in the values.yaml:
      i. offsetsTopicReplicationFactor: 1
      ii. transactionStateLogReplicationFactor: 1

c. Scale down the Kafka and zookeeper deployment by modifying the following lines in the corresponding worker group helm chart and custom values:

```
ocnaddkafka:
      enabled: false
```

d. Perform helm upgrade for the worker group:

```
helm upgrade <release name> <chart path of worker group> -n <namespace
of the worker group>
```

e. Delete PVC for Kafka and Zookeeper using the following commands:

i. `kubectl delete pvc -n <namespace> kafka-volume-kafka-broker-0`

ii. `kubectl delete pvc -n <namespace> kafka-volume-kafka-broker-1`

iii. `kubectl delete pvc -n <namespace> kafka-volume-kafka-broker-2`

iv. `kubectl delete pvc -n <namespace> kafka-broker-security-zookeeper-0`

v. `kubectl delete pvc -n <namespace> kafka-broker-security-zookeeper-1`

vi. `kubectl delete pvc -n <namespace> kafka-broker-security-zookeeper-2`

f. Modify the value of the following parameter to true, in the corresponding worker group helm chart and custom values

```
ocnaddkafka:
      enabled: true
```

g. Perform helm upgrade for the worker group:

```
helm upgrade <release name> <chart path of worker group> -n <namespace
of the worker group>
```

> **Note**
>
> The following points are to be considered while applying the above procedure:
>
> 1. In case a Kafka broker becomes unavailable, then you may experience an impact on the traffic on the Ingress side.
> 2. Verify the Kafka broker logs or describe the Kafka/zookeeper pod which is unavailable and take the necessary action based on the error reported.

500 Server Error on GUI while creating/deleting the Data Feed

Problem

Occasionally, due to network issues, the user may observe a "500 Server Error" while creating/deleting the Data Feed.

Solution

The following actions generally resolve the issue:

- Delete and recreate the feed if it is not created properly.
- Retry the action after logging out from the GUI and login back again.
- Retry creating/deleting the feed after some time.

Kafka resources reaching more than 90% utilization

Problem

Kafka resources(CPU, Memory) reached more than 90% utilization due to a higher MPS rate or slow disk I/O rate

Solution

Add additional resources to the following parameters that are reaching high utilization.

File name: `ocnadd-custom-values.yaml` corresponding to the worker group

Parameter name: `ocnaddkafka.ocnadd.kafkaBroker.resource`

```
kafkaBroker:
     name:kafka-broker
     resource:
        limit:
           cpu:5    ===> change it to require number of CPUs
           memory:24Gi   ===> change it to require number of memory size
```

Kafka ACLs: Identifying the Network IP "Host" ACLs in Kafka Feed

Problem

User is unable to identify the Network IP "Host" ACLs in Kafka Feed.

Solution

The following procedure can be referred to in the case of the user being unable to identify the Network IP address.

Adding network IP address for Host ACL

This set of instructions explains how to add a network IP address to the host ACL. The procedure is illustrated using the following example configuration:

- Kafka Feed Name: demofeed
- Kafka ACL User: joe
- Kafka Client Hostname: 10.1.1.15

> **Note**
>
> The below steps should be run corresponding to the worker group against which the Kafka feed is created/modified

Retrieving Current ACLs

To obtain the current ACLs configured for the "demofeed" feed on the Kafka service, run the following command from any POD within the OCNADD deployment:

```
curl -k --location --request GET 'https://ocnaddadminservice:9181/ocnadd-
admin-svc/v2/{workerGroup}/acls'
```

The output will be in JSON format and might look like this:

```
["(pattern=ResourcePattern(resourceType=GROUP, name=demofeed,
patternType=LITERAL), entry=(principal=User:joe, host=10.1.1.15,
operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=TOPIC,
name=MAIN, patternType=LITERAL), entry=(principal=User:joe, host=10.1.1.15,
operation=READ, permissionType=ALLOW))"]
```

Authorization Error in Kafka Logs

With the Kafka feed "demofeed" configured with acl user "joe" and client host IP address "10.1.1.15" the Kafka reports the following authorization error in the Kafka Logs.

```
[2023-07-31 05:34:22,063] INFO Principal = User:joe is Denied Operation =
Read from host = 10.1.1.0 on resource = Group:LITERAL:demofeed for request =
JoinGroup with resourceRefCount = 1 (kafka.authorizer.logger)
```

In the above output, it can be seen that host ACL is allowing specific client IP address "10.1.1.15", whereas the Kafka server is expecting the ACL for the network IP "10.1.1.0" too, which is the network IP address.

Steps to Create Network IP Address ACL

1. **Check Kafka Logs**
   To identify the network IP address that Kafka is denying against the configured feed, follow these steps:
   a. Check the Kafka logs using the command:

      ```
      kubectl logs -n <namespace> -c kafka-broker kafka-broker-1 -f
      ```

      For example:

      ```
      kubectl logs -n ocnadddeploy -c kafka-broker kafka-broker-1 -f
      ```

   b. Look for traces similar to this:

      ```
      Principal = User:joe is Denied Operation = Read from host = 10.1.1.0 on
      resource = Group:LITERAL:demofeed for request = JoinGroup with
      resourceRefCount = 1 (kafka.authorizer.logger)  take the ip address
      which is being denied by Kafka, in this case, it is "10.1.1.0"
      ```

      Identify the denied IP address; in this case, it is "10.1.1.0."
2. **Create Host ACL for Network IP**

a. Access any Pod within the OCDD deployment, such as kafka-broker-0:

```
kubectl exec -it kafka-broker-0 -n <namespace> -- bash
```

b. Run the provided curl commands to configure the host network IP ACLs:

```
curl -k --location --request POST 'https://ocnaddconfiguration:12590/
ocnadd-configuration/v2/{workerGroup}/client-acl' --header 'Content-
Type: application/json' --data-raw '{
    "principal": "joe",
    "hostName": "10.1.1.0",
    "resourceType": "TOPIC",
    "resourceName": "MAIN",
    "aclOperation": "READ"
}'

curl -k --location --request POST 'https://ocnaddconfiguration:12590/
ocnadd-configuration/v2/{workerGroup}/client-acl' --header 'Content-
Type: application/json' --data-raw '{
    "principal": "joe",
    "hostName": "10.1.1.0",
    "resourceType": "GROUP",
    "resourceName": "demofeed",
    "aclOperation": "READ"
}'
```

3. **Verify ACLs**
   Use the following curl command to verify the ACLs:

```
curl -k --location --request GET 'https://ocnaddadminservice:9181/ocnadd-
admin-svc/v2/{workerGroup}/acls'
```

Here is an example of the expected output, indicating ACLs for `With Feed Name: demofeed, ACL user: joe, Host Name:10.1.1.15, Network IP:10.1.1.0`:

```
["(pattern=ResourcePattern(resourceType=GROUP, name=demofeed,
patternType=LITERAL), entry=(principal=User:joe, host=10.1.1.0,
operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=GROUP,
name=demofeed, patternType=LITERAL), entry=(principal=User:joe,
host=10.1.1.15, operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=TOPIC,
name=MAIN, patternType=LITERAL), entry=(principal=User:joe, host=10.1.1.0,
operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=TOPIC,
name=MAIN, patternType=LITERAL), entry=(principal=User:joe,
host=10.1.1.15, operation=READ, permissionType=ALLOW))"]
```

Producer Unable to Send traffic to OCNADD when an External Kafka Feed is enabled

Problem

Producer is unable to send traffic to OCNADD when an External Kafka Feed is enabled.

Solution

Follow the below steps to debug and investigate if the producer is unable to send traffic to DD when ACL is enabled and there are unauthorization errors coming in producer NF logs.

Debug and Investigation Steps:

> **Note**
>
> The below steps should be run corresponding to the worker group against which the Kafka feed is being enabled

1. Begin by creating the `admin.properties` file within the Kafka broker, following Step 2 of "Update SCRAM Configuration with Users" as outlined in the *Oracle Communications Network Analytics Data Director User Guide*.
2. If the producer's security protocol is SASL_SSL (port 9094), verify whether the users have been created in SCRAM. Use the following command for verification:

   ```
   ./kafka-configs.sh --bootstrap-server kafka-broker:9094 --describe --
   entity-type users --command-config ../../admin.properties
   ```

   If no producer's SCRAM ACL users are found, see to the "Prerequisites for External Consumers" section in the *Oracle Communications Network Analytics Data Director User Guide* to create the necessary Client ACL users.
3. In case the producer's security protocol is SSL (port 9093), ensure that the Network Function (NF) producer's certificates have been correctly generated as per the instructions provided in the *Oracle Communications Network Analytics Suite Security Guide*.
4. Check whether the producer client ACLs have been set up based on the configured security protocol (SASL_SSL or SSL) in the NF Kafka Producers. To verify this:
   a. Access any Pod from the OCNADD deployment. For instance, `kafka-broker-0`:

      ```
      kubectl exec -it kafka-broker-0 -n <namespace> -- bash
      ```

   b. Run the following curl command to list all the ACLs:

      ```
      curl -k --location --request GET 'https://ocnaddadminservice:9181/
      ocnadd-admin-svc/v2/{workerGroup}/acls'
      ```

      The expected output might resemble the following example, indicating `With Feed Name: demofeed, ACL user: joe, Host Name:10.1.1.15, Network IP:10.1.1.0`:

      ```
      ["(pattern=ResourcePattern(resourceType=GROUP, name=demofeed,
      patternType=LITERAL), entry=(principal=User:joe, host=10.1.1.0,
      operation=READ,
      permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=GROUP,
      name=demofeed, patternType=LITERAL), entry=(principal=User:joe,
      host=10.1.1.15, operation=READ,
      permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=TOPIC,
      name=MAIN, patternType=LITERAL), entry=(principal=User:joe,
      host=10.1.1.0, operation=READ,
      permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=TOPIC,
      ```

```
name=MAIN, patternType=LITERAL), entry=(principal=User:joe,
host=10.1.1.15, operation=READ, permissionType=ALLOW))"]
```

5. If no ACLs are found as observed in step 4, follow the "Create Client ACLs" section provided in the *Oracle Communications Network Analytics Data Director User Guide* to establish the required ACLs.
   By following these steps, you will be able to diagnose and address issues related to the producer's inability to send traffic to OCNADD when an External Kafka Feed is enabled, and ACL-related authorization errors are encountered.

External Kafka Consumer Unable to consume messages from DD

Problem

External Kafka Consumer is unable to consume messages from OCNADD.

Solution

If you are experiencing issues where an external Kafka consumer is unable to consume messages from OCNADD, especially when ACL is enabled and unauthorized errors are appearing in the Kafka feed's logs, follow the subsequent steps for debugging and investigation:

Debug and Investigation Steps:

> **Note**
>
> The below steps should be run corresponding to the worker group against which the Kafka feed is created/modified.

1. Verify that ACL Users created for the Kafka feed, along with SCRAM users, are appropriately configured in the JAAS config by executing the following command:

   ```
   ./kafka-configs.sh --bootstrap-server kafka-broker:9094 --describe --
   entity-type users --command-config  ../../admin.properties
   ```

2. Validate that the Kafka feed parameters have been correctly configured in the consumer client. If not, ensure proper configuration and perform an upgrade on the Kafka feed's consumer application.
3. Inspect the logs of the external consumer application.
   a. If you encounter an error related to "XYZ Group authorization failure" in the consumer application logs, follow these steps:
      i. Access any Pod within the OCNADD deployment. For example, `kafka-broker-0`:

         ```
         kubectl exec -it kafka-broker-0 -n <namespace> -- bash
         ```

      ii. Run the curl command below to retrieve ACLs information and verify the existence of ACLs for the Kafka feed:

         ```
         curl -k --location --request GET 'https://ocnaddadminservice:9181/
         ocnadd-admin-svc/v2/{workerGroup}/acls'
         ```

Sample output with `Feed Name: demofeed, ACL user: joe, Host Name:10.1.1.15, Network IP:10.1.1.0`:

```
["(pattern=ResourcePattern(resourceType=GROUP, name=demofeed,
patternType=LITERAL), entry=(principal=User:joe, host=10.1.1.0,
operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=GROUP
, name=demofeed, patternType=LITERAL), entry=(principal=User:joe,
host=10.1.1.15, operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=TOPIC
, name=MAIN, patternType=LITERAL), entry=(principal=User:joe,
host=10.1.1.0, operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=TOPIC
, name=MAIN, patternType=LITERAL), entry=(principal=User:joe,
host=10.1.1.15, operation=READ, permissionType=ALLOW))"]
```

iii. If no ACL is found for the Kafka feed with the resource type "Group," run the following curl command to create the Group resource type ACLs:

```
  curl -k --location --request POST 'https://
ocnaddconfiguration:12590/ocnadd-configuration/v2/{workerGroup}/
client-acl' --header 'Content-Type: application/json' --data-raw '{
     "principal": "<ACL-USER-NAME>",
     "resourceType": "GROUP",
     "resourceName": "<KAFKA-FEED-NAME>",
     "aclOperation": "READ"
     }'
```

b. If you encounter an error related to "XYZ TOPIC authorization failure" in the consumer application logs, follow these steps:

i. Access any Pod within the OCNADD deployment. For example, `kafka-broker-0`:

```
kubectl exec -it kafka-broker-0 -n <namespace> -- bash
```

ii. Run the curl command below to retrieve ACLs information and verify the existence of ACLs for the Kafka feed:

```
curl -k --location --request GET 'https://ocnaddadminservice:9181/
ocnadd-admin-svc/v2/{workerGroup}/acls'
```

Sample output with `Feed Name: demofeed, ACL user: joe, Host Name:10.1.1.15, Network IP:10.1.1.0`:

```
["(pattern=ResourcePattern(resourceType=GROUP, name=demofeed,
patternType=LITERAL), entry=(principal=User:joe, host=10.1.1.0,
operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=GROUP
, name=demofeed, patternType=LITERAL), entry=(principal=User:joe,
host=10.1.1.15, operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=TOPIC
, name=MAIN, patternType=LITERAL), entry=(principal=User:joe,
host=10.1.1.0, operation=READ,
permissionType=ALLOW))","(pattern=ResourcePattern(resourceType=TOPIC
```

```
, name=MAIN, patternType=LITERAL), entry=(principal=User:joe,
host=10.1.1.15, operation=READ, permissionType=ALLOW))"]
```

iii. If no ACL is found for the Kafka feed with the resource type "TOPIC," run the following curl command to create the TOPIC resource type ACLs:

```
curl -k --location --request POST 'https://
ocnaddconfiguration:12590/ocnadd-configuration/v2/{workerGroup}/
client-acl' --header 'Content-Type: application/json' --data-raw '{
     "principal": "<ACL-USER-NAME>",
     "resourceType": "TOPIC",
     "resourceName": "MAIN",
     "aclOperation": "READ"
     }'
```

Database Error During Kafka Feed Update

Problem:

When attempting to update a Kafka feed, you may encounter an error similar to the following:

Error Message: Updating the Kafka feed in the database has failed.

Solution:

To address this issue, retry the Kafka feed update using the Update Kafka feed option from OCNADD UI with the same information as in the previous attempt.

2.2 Helm Install and Upgrade Failure

This section describes the various helm installation or upgrade failure scenarios and the respective troubleshooting procedures:

2.2.1 Incorrect Image Name in *ocnadd-custom-values.yaml* File

Problem

`helm install` fails if an incorrect image name is provided in the `ocnadd-custom-values.yaml` file or if the image is missing in the image repository.

Error Code or Error Message

When you run `kubectl get pods -n <ocnadd_namespace>`, the status of the pods might be `ImagePullBackOff` or `ErrImagePull`.

Solution

Perform the following steps to verify and correct the image name:

1. Edit the `ocnadd-custom-values.yaml` file and provide the release specific image names and tags.
2. Run the `helm install` command.
3. Run the `kubectl get pods -n <ocnadd_namespace>` command to verify if all the pods are in **Running** state.

2.2.2 Failed Helm Installation/Upgrade Due to Prometheus Rules Applying Failure

Scenario:

Helm installation or upgrade fails due to Prometheus rules applying failure.

Problem:

Helm installation or upgrade might fail if Prometheus service is down or Prometheus PODs are not available during the helm installation or upgrade of the OCNADD.

Error Code or Error Message:

```
Error: UPGRADE FAILED: cannot patch "ocnadd-alerting-rules" with kind
PrometheusRule: Internal error occurred: failed calling webhook
"prometheusrulemutate.monitoring.coreos.com": failed to call webhook: Post
"https://occne-kube-prom-stack-kube-operator.occne-infra.svc:443/admission-
prometheusrules/mutate?timeout=10s": context deadline exceeded
```

Solution:

Perform the following steps to proceed with the OCNADD helm install or upgrade:

1. Move the "`ocndd-alerting-rules.yaml`" and "`ocnadd-mgmt-alerting-rules.yaml`" from the `<chart_path>/helm-charts/templates` to some other directory outside the OCNADD charts.
2. Continue with the helm install/upgrade.
3. Run the following command to verify if the status of all the pods are running:

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

The OCNADD Prometheus alerting rules must be applied again when the Prometheus service and PODs are available back in service. Ensure to apply the alerting rules using the Helm upgrade procedure itself by moving back the "`ocndd-alerting-rules.yaml`" and "`ocnadd-mgmt-alerting-rules.yaml`" files in the `<chart_path>/helm-charts/templates` directory.

2.2.3 Docker Registry is Configured Incorrectly

Problem

`helm install` might fail if the Docker Registry is not configured in all primary and secondary nodes.

Error Code or Error Message

When you run `kubectl get pods -n <ocnadd_namespace>`, the status of the pods might be ImagePullBackOff or ErrImagePull.

Solution

Configure the Docker Registry on all primary and secondary nodes. For information about Docker Registry configuration, see *Oracle Communications Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

2.2.4 Continuous Restart of Pods

Problem

`helm install` might fail if MySQL primary or secondary hosts are not configured properly in `ocnadd-custom-values.yaml`.

Error Code or Error Message

When you run `kubectl get pods -n <ocnadd_namespace>`, the pods shows restart count increases continuously, or there is a Prometheus alert for continuous pod restart.

Solution

1. Verify MySQL connectivity.

   MySQL servers may not be configured properly. For more information about the MySQL configuration, see *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.

2. Describe the POD to check more details on the error, troubleshoot further based on the reported error.
3. Check the POD log for any error, troubleshoot further based on the reported error.

2.2.5 Adapter Deployment Removed during Upgrade or Rollback

Problem

Adapter(data feed) is deleted during upgrade or rollback.

Error Code or Error Message

Sample error message displayed:

Figure 2-1 Error Message

```
OCL 2023-04-26 14:07:11.421 [boundedElastic-3] ERROR com.oracle.cgbu.cne.ocnadd.admin.svc.service.consumerAdapter.ConsumerAdapterService - exception in deploying
io.fabric8.kubernetes.client.KubernetesClientException: Operation: [list]  for kind: [Deployment]  with name: [null]  in namespace: [ocnadd-sg]  failed.
    at io.fabric8.kubernetes.client.KubernetesClientException.launderThrowable(KubernetesClientException.java:159) ~[kubernetes-client-api-6.3.1.jar!/:?]
    at io.fabric8.kubernetes.client.dsl.internal.BaseOperation.list(BaseOperation.java:420) ~[kubernetes-client-6.3.1.jar!/:?]
    at io.fabric8.kubernetes.client.dsl.internal.BaseOperation.list(BaseOperation.java:383) ~[kubernetes-client-6.3.1.jar!/:?]
    at io.fabric8.kubernetes.client.dsl.internal.BaseOperation.list(BaseOperation.java:93) ~[kubernetes-client-6.3.1.jar!/:?]
    at com.oracle.cgbu.cne.ocnadd.admin.svc.service.consumerAdapter.ConsumerAdapterService.deployConsumerAdapter(ConsumerAdapterService.java:677) ~[classes!/:1.0.72]
    at com.oracle.cgbu.cne.ocnadd.admin.svc.controller.AdminControllerImpl.lambda$createConsumerGroup$0(AdminControllerImpl.java:352) ~[classes!/:1.0.72]
    at reactor.core.publisher.MonoCallable.call(MonoCallable.java:92) ~[reactor-core-3.4.26.jar!/:3.4.26]
    at reactor.core.publisher.FluxSubscribeOnCallable$CallableSubscribeOnSubscription.run(FluxSubscribeOnCallable.java:227) ~[reactor-core-3.4.26.jar!/:3.4.26]
    at reactor.core.scheduler.SchedulerTask.call(SchedulerTask.java:68) ~[reactor-core-3.4.26.jar!/:3.4.26]
    at reactor.core.scheduler.SchedulerTask.call(SchedulerTask.java:28) ~[reactor-core-3.4.26.jar!/:3.4.26]
    at java.util.concurrent.FutureTask.run(FutureTask.java:264) ~[?:?]
    at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:304) ~[?:?]
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136) ~[?:?]
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635) ~[?:?]
    at java.lang.Thread.run(Thread.java:833) ~[?:?]
Caused by: java.io.InterruptedIOException
    at io.fabric8.kubernetes.client.dsl.internal.OperationSupport.waitForResult(OperationSupport.java:523) ~[kubernetes-client-6.3.1.jar!/:?]
    at io.fabric8.kubernetes.client.dsl.internal.BaseOperation.list(BaseOperation.java:418) ~[kubernetes-client-6.3.1.jar!/:?]
    ... 13 more
```

Solution

This can be fixed by running the following commands:

- Run the following command to verify the data feeds:

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

- If data feeds are missing, verify the above mentioned error message in the admin service log, by running the following command:

  ```
  kubectl logs <admin svc pod name> -n <namespace>
  ```

- If the error message is present, run the following command:

  ```
  kubectl rollout restart deploy <configuration app name> -n <namespace>
  ```

2.2.6 *ocnadd-custom-values.yaml* File Parse Failure

This section explains the troubleshooting procedure in case of failure while parsing the `ocnadd-custom-values.yaml` file.

Problem

Unable to parse the `ocnadd-custom-values.yaml` file or any other `values.yaml` while running `Helm install`.

Error Code or Error Message

Error: failed to parse `ocnadd-custom-values.yaml`: error converting YAML to JSON: yaml

Symptom

When parsing the `ocnadd-custom-values.yaml` file, if the above mentioned error is received, it indicates that the file is not parsed because of the following reasons:

- The tree structure may not have been followed.
- There may be a tab space in the file.

Solution

Download the latest OCNADD custom templates zip file from MoS. For more information, see *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.

2.2.7 Kafka Brokers Continuously Restart after Reinstallation

Problem

When re-installing OCNADD in the same namespace without deleting the PVC that was used for the first installation, Kafka brokers will go into crashloopbackoff status and keep restarting.

Error Code or Error Message

When you run, `kubectl get pods -n <ocnadd_namespace>` the broker pod's status might be Error/crashloopbackoff and it might keep restarting continuously, with "no disk space left on the device" errors in the pod logs.

Solution

1. Delete the Stateful set (STS) deployments of the brokers. Run `kubectl get sts -n <workergroup_namespace>` to obtain the Stateful sets in the namespace.

2. Delete the STS deployments of the services with disk full issue. For example, run the command `kubectl delete sts -n <workergroup_namespace> kafka-broker1 kafka-broker2`.
3. After deleting the STS of the brokers delete the pvc. Delete the PVCs in the namespace, which is used by kafka-brokers. Run `kubectl get pvc -n <workergroup_namespace>` to get the PVCs in that namespace.
   The number of PVCs used is based on the number of brokers deployed. Therefore, select the PVCs that have the name kafka-broker or zookeeper, and delete them. To delete the PVCs, run `kubectl delete pvc -n <workergroup_namespace> <pvcname1> <pvcname2>`.

For example:

For a three broker setup in worker group namespace ocnadd-wg1, you will need to delete these PVCs:

```
kubectl delete pvc -n ocnadd-wg1 broker1-pvc-kafka-broker1-0, broker2-pvc-
kafka-broker2-0, broker3-pvc-kafka-broker3-0, kafka-broker-security-
zookeeper-0, kafka-broker-security-zookeeper-1 kafka-broker-security-
zookeeper-2
```

2.2.8 Kafka Brokers Continuously Restart After the Disk is Full

Problem

While there is no disk space left on the broker or zookeeper in a corresponding worker group.

Error Code or Error Message

When you run `kubectl get pods -n <ocnadd_namespace>`, the broker pod's status might be error, or crashloopbackoff and it might keep restarting continuously.

Solution

> **Note**
>
> The below steps should be run corresponding to the worker group against which the Kafka is reporting disk full error.

1. Delete the STS(stateful set) deployments of the brokers:
   a. Get the STS in the namespace with the following command:

      ```
      kubectl get sts -n <ocnadd_namespace>
      ```

   b. Delete the STS deployments of the services with disk full issue:

      ```
      kubectl delete sts -n <ocnadd_namespace> <sts1> <sts2>
      ```

      For example, for three broker setup:

      ```
      kubectl delete sts -n ocnadd-deploy kafka-broker1 kafka-broker2 kafka-
      broker3 zookeeper
      ```

2. Delete the PVCs in that namespace that is used by the removed kafka-brokers. To get the PVCs in that namespace:

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

   The number of PVCs used will be based on the number of brokers you deploy. Select the PVCs that have the name kafka-broker or zookeeper and delete them.

   a. To delete PVCs, run:

      ```
      kubectl delete pvc -n <ocnadd_namespace> <pvcname1>  <pvcname2>
      ```

      For example, for a three broker setup in namespace ocnadd-deploy, you must delete these PVCs;

      ```
      kubectl delete pvc -n ocnadd-deploy broker1-pvc-kafka-broker1-0
      broker2-pvc-kafka-broker2-0  broker3-pvc-kafka-broker3-0  kafka-broker-
      security-zookeeper-0  kafka-broker-security-zookeeper-1 kafka-broker-
      security-zookeeper-2
      ```

3. Once the STS and PVC's are deleted for the services, edit the respective broker's values.yaml to increase the PV size of the brokers at the location: `<chartpath>/charts/ocnaddkafka/values.yaml`.

   If any formatting or indentation issues occur while editing, refer to the files in

   ```
   <chartpath>/charts/ocnaddkafka/default
   ```

   To increase the storage, edit the fields pvcClaimSize for each broker. For recommendation of PVC storage, see *Oracle Communications Network Analytics Data Director Benchmarking Guide*.

4. Upgrade the Helm chart after increasing the PV size

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

5. Create the required topics.

2.2.9 Kafka Brokers Restart on Installation

Problem

Kafka brokers re-start during OCNADD installation.

Error Code or Error Message

The output of the command `kubectl get pods -n <ocnadd_namespace>` displays the broker pod's status as restarted.

Solution

The Kafka Brokers wait for a maximum of 3 minutes for the Zookeepers to come online before they are started. If the Zookeeper cluster does not come online within the given interval, the broker will start before the Zookeeper and will error out as it does not have access to the Zookeeper.This may Zookeeper may start after the 3 interval as the node may take more time

to pull the images due to network issues.Therefore, when the zookeeper does not come online within the given time this issue may be observed.

2.2.10 Kafka Brokers in Crashloop State After Rollback

Problem

Kafka brokers pods in crashloop state on rollback from 23.4.0

Error Code or Error Message

The Kafka broker pods shows the following error traces

```
kubectl logs -n ocnadd-deploy kafka-broker-3 auto-discovery
```

```
E1117 17:02:02.770723 9 memcache.go:238] couldn't get current server API
group list: Get "http://localhost:8080/api?timeout=32s": dial tcp [::1]:8080:
connect: connection refused
E1117 17:02:02.771065 9 memcache.go:238] couldn't get current server API
group list: Get "http://localhost:8080/api?timeout=32s": dial tcp [::1]:8080:
connect: connection refused
```

Solution

Perform the following steps to resolve the issue:

1. Edit the service account using the below command
   a. `kubectl edit sa -n ocnadd-deploy ocnadd-deploy-sa-ocnadd`
   b. Update the parameter "`automountServiceAccountToken`" to `true`
      `automountServiceAccountToken: true`
2. Save the service account.
3. Run the `kubectl get pods -n <ocnadd_namespace>` command to verify if the status of all the Kafka pods is now Running.

2.2.11 Database Goes into the Deadlock State

Problem

MySQL locks get struck.

Error Code or Error Message

ERROR 1213 (40001): Deadlock found when trying to get lock; try restarting the transaction.

Symptom

Unable to access MySQL.

Solution

Perform the following steps to remove the deadlock:

1. Run the following command on each SQL node:

```
SELECT
CONCAT('KILL ', id, ';')
FROM INFORMATION_SCHEMA.PROCESSLIST
WHERE `User` = <DbUsername>
AND `db` = <DbName>;
```

This command retrieves the list of commands to kill each connection.
Example:

```
select
 CONCAT('KILL ', id, ';')
 FROM INFORMATION_SCHEMA.PROCESSLIST
 where `User` = 'ocnadduser'
      AND `db` = 'ocnadddb';
+-------------------------+
| CONCAT('KILL ', id, ';') |
+-------------------------+
| KILL 204491;            |
| KILL 200332;            |
| KILL 202845;            |
+-------------------------+
3 rows in set (0.00 sec)
```

2. Run the kill command on each SQL node.

2.2.12 The Pending Rollback Issue Due to PreRollback Database Rollback Job Failure

Scenario: Rollback to previous release or above versions are failing due to pre-rollback-db job failure and OCNADD is entering into pending-rollback status

Problem: In this case, OCNADD gets stuck and can't proceed with any other operation like install/upgrade/rollback.

Error Logs - Example with 23.2.0.0.1:

```
>helm rollback ocnadd 1 -n <namespace>

Error: job failed: BackoffLimitExceeded
```

```
$ helm history ocnadd -n ocnadd-deploy

REVISION        UPDATED                         STATUS
CHART                   APP VERSION     DESCRIPTION
1               Fri Jul 14 10:38:31 2023        superseded
ocnadd-23.2.0           23.2.0          Install complete
2               Fri Jul 14 11:38:31 2023        superseded
ocnadd-23.2.0           23.2.0.0.1      Upgrade complete
3               Fri Jul 14 11:40:17 2023        superseded
```

```
ocnadd-23.3.0            23.3.0.0.0      Upgrade complete
4               Mon Jul 17 07:22:04 2023        pending-rollback
ocnadd-23.2.0           23.2.0.0.1      Rollback to 1


$ helm upgrade ocnadd <helm chart> -n ocnadd-deploy

Error: UPGRADE FAILED: another operation (install/upgrade/rollback) is in
progress
```

Solution:

To resolve the pending-rollback issue, delete the secrets related to the 'pending-rollback' revision. Follow these steps:

1. **Get secrets using kubectl**

```
$kubectl get secrets -n ocnadd-deploy
NAME
                                TYPE                                  DATA
  AGE
adapter-secret
                                                                      Opaque
                     8      14d
certdbfilesecret
                                                                      Opaque
                     1      14d
db-secret
                                Opaque                                6
     47h
default-token-dmrqq
                            kubernetes.io/service-account-token   3      14d
egw-secret
                                                                      Opaque
                    8      14d
jaas-secret
                                                                       Opaque
                      1      4d19h
kafka-broker-secret
                                           Opaque
     8      14d
ocnadd-deploy-admin-sa-token-mfh6l
                                             kubernetes.io/service-account-
token   3      4d19h
ocnadd-deploy-cache-sa-token-g7rd2
                                          kubernetes.io/service-account-token
  3      4d19h
ocnadd-deploy-gitlab-admin-token-qfmmf
                                              kubernetes.io/service-account-
token   3      4d19h
ocnadd-deploy-kafka-sa-token-2qqs9
                                          kubernetes.io/service-account-token
  3      4d19h
ocnadd-deploy-sa-ocnadd-token-tj2zf
                                          kubernetes.io/service-account-token
```

```
  3      47h
ocnadd-deploy-zk-sa-token-9x2rb
                                    kubernetes.io/service-account-token
3      4d19h
ocnaddadminservice-secret
                                                          Opaque
                      8      14d
ocnaddalarm-secret
                                                         Opaque
                     8      14d
ocnaddcache-secret
                                                         Opaque
                     8      14d
ocnaddconfiguration-secret
                                                         Opaque
                     8      14d
ocnaddhealthmonitoring-secret
                                                          Opaque
                      8      14d
ocnaddnrfaggregation-secret
                                                          Opaque
                      8      14d
ocnaddscpaggregation-secret
                                                          Opaque
                      8      14d
ocnaddseppaggregation-secret
                                                         Opaque
                     8      14d
ocnaddthirdpartyconsumer-secret
                                                          Opaque
                      8      14d
ocnadduirouter-secret
                                                          Opaque
                      8      14d
oraclenfproducer-secret
                                                          Opaque
                      8      14d
regcred-sim
                      kubernetes.io/dockerconfigjson        1      8d
secret
                                                         Opaque
                     7      4d19h

sh.helm.release.v1.ocnadd.v1
                                                         helm.sh/release.v1
                     1      4d19h
sh.helm.release.v1.ocnadd.v2
                                                         helm.sh/release.v1
                     1      4d19h
sh.helm.release.v1.ocnadd.v3
                                                         helm.sh/release.v1
                     1      4d19h
sh.helm.release.v1.ocnadd.v4
                                                           helm.sh/
release.v1                    1      47h
sh.helm.release.v1.ocnaddsim.v1
```

```
                                                                helm.sh/
release.v1                    1      8d
zookeeper-secret
                                                               Opaque
                    8      14d
```

2. **Delete Secrets Related to Pending-Rollback Revision:** In this case the secrets of revision 4, that is, `'sh.helm.release.v1.ocnadd.v4'` need to be deleted since the data director entered 'pending-rollback' status in revision 4:

   ```
   kubectl delete secrets sh.helm.release.v1.ocnadd.v4 -n ocnadd-deploy
   ```

   Sample output:

   ```
   secret "sh.helm.release.v1.ocnadd.v4" deleted
   ```

3. **Check Helm History:** Verify that the pending-rollback status has been cleared using the following command:

   ```
   helm history ocnadd -n ocnadd-deploy
   ```

   Sample output:

   ```
   REVISION        UPDATED                         STATUS
   CHART                   APP VERSION     DESCRIPTION

   1               Fri Jul 14 10:38:31 2023        superseded
   ocnadd-23.2.0           23.2.0          Install complete
   2               Fri Jul 14 11:38:31 2023        superseded
   ocnadd-23.2.0           23.2.0.0.1      Upgrade complete
   3               Fri Jul 14 11:40:17 2023        superseded
   ocnadd-23.3.0-rc.2      23.3.0.0.0      Upgrade complete
   ```

4. **Restore Database Backup:** Restore the database backup taken before the upgrade started. Follow the "Create OCNADD Restore Job" section of the "Fault Recovery" from the *Oracle Communications Network Analytics Data Director Installation, Upgrade and Fault Recovery Guide*.

5. **Perform Rollback:** Perform rollback again using the following command:

   ```
   helm rollback <release name> <revision number> -n <namespace>
   ```

   For example:

   ```
   helm rollback ocnadd 1 -n ocnadd-deploy --no-hooks
   ```

6. **Verification:** Verify that end-to-end traffic is running between the DD and the corresponding third-party application.

2.2.13 Upgrade fails due to unsupported changes

Problem

Upgrade failed from the source release to the target release due to unsupported changes in the target release

The upgrade failed from the source release to the target release due to unsupported changes in the target release (during the upgrade the Database Job was successful but the upgrade failed due to an error).

> **Note**
>
> This issue is not a generic issue, however, may occur if users are unable to sync up the target release charts.

Example:

Scenario: If there is a PVC size mismatch from the source release to the target release.

Error Message in Helm History: Example with 23.2.0.0.1 to 23.3.0

```
Error: UPGRADE FAILED: cannot patch "zookeeper" with kind StatefulSet:
StatefulSet.apps "zookeeper" is invalid: spec: Forbidden: updates to
statefulset spec for fields other than 'replicas', 'template',
'updateStrategy', 'persistentVolumeClaimRetentionPolicy' and
'minReadySeconds' are forbidden
```

Run the following command: (with 23.2.0.0.1)

```
helm history <release-name> -n <namespace>
```

Sample output with version 23.2.0.0.1:

```
REVISION        UPDATED                         STATUS
CHART                   APP VERSION     DESCRIPTION
1               Tue Aug  1 07:01:29 2023        superseded
ocnadd-23.2.0           23.2.0          Install complete
2               Tue Aug  1 07:08:29 2023        deployed
ocnadd-23.2.0           23.2.0.0.1      Upgrade complete
3               Tue Aug  1 07:24:08 2023        failed
ocnadd-23.3.0           23.3.0.0.0      Upgrade "ocnadd" failed: cannot patch
"zookeeper" with kind StatefulSet: StatefulSet.apps "zookeeper" is invalid:
spec: Forbidden: updates to statefulset spec for fields other than
'replicas', 'template', 'updateStrategy',
'persistentVolumeClaimRetentionPolicy' and 'minReadySeconds' are forbidden
```

Solution

Perform the following steps:

1. Correct and sync the target release charts as per the source release, and ensure that no new feature of the new release is enabled.
2. Perform an upgrade. For more information about the upgrade procedure, see "Upgrading OCNADD" in the *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.

2.2.14 Upgrade Failed from Source Release to Target Release Due to Helm Hook Upgrade Database Job

Problem

Upgrade failed from patch source release to target release due to helmhook upgrade DB job (Upgrade job fails).

Error Code or Error Message: Example with 23.2.0.0.1 to 23.3.0

```
Error:  UPGRADE FAILED: pre-upgrade hooks failed: job failed:
BackoffLimitExceeded
```

Run the following command:

```
helm history <release-name> -n <namespace>
```

Sample Output:

```
REVISION        UPDATED                         STATUS
CHART                   APP VERSION         DESCRIPTION

1             Tue Aug  1 07:01:27 2023        superseded
ocnadd-23.2.0             23.2.0              Install complete
2             Tue Aug  1 07:08:29 2023        deployed
ocnadd-23.2.0             23.2.0.0.1          Upgrade complete
2             Tue Aug  1 07:24:08 2023        failed
ocnadd-23.3.0             23.3.0.0.0          Upgrade "ocnadd" failed: pre-
upgrade hooks failed: job failed: BackoffLimitExceeded
```

Solution

Rollback to patch, correct the errors, and then run the upgrade once again.

1. Run helm rollback to previous release revision:

   ```
   helm rollback <helm release name> <revision number> -n <namespace>
   ```

2. Restore the Database backup taken before upgrade. For more information see, the procedure "Create OCNADD Restore Job" in the "Fault Recovery" section in the *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.
3. Correct the upgrade issue and run a fresh upgrade. For more information, see "Upgrading OCNADD" in the *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.

2.2.15 Upgrade fails due to Database MaxNoOfAttributes exhausted

Scenario:

Upgrade fails due to Database MaxNoOfAttributes exhausted

Problem:

Helm upgrade may fail due to maximum number for attributes allowed to be created has reached maximum limit.

Error Code or Error Message:

```
Executing::::::: /tmp/230300001.sql
mysql: [Warning] Using a password on the command line interface can be
insecure.
ERROR 1005 (HY000) at line 51: Can't create destination table for copying
alter table (use SHOW WARNINGS for more info).
error in executing upgrade db scripts
```

Solution:

Delete few database schemas that are not being used or the ones which are stale.

For example, from MySQL prompt, `drop database xyz;`

> **Note**
>
> Dropping unused or stale database schemas is a valid approach. However, exercise caution when doing this to ensure you are not deleting important data. Make sure to have proper backups before proceeding.

2.2.16 Webhook Failure During Installation or Upgrade

Problem

Installation or upgrade unsuccessful due to webhook failure.

Error Code or Error Message

Sample error log:

```
Error: INSTALLATION FAILED: Internal error occurred: failed calling webhook
"prometheusrulemutate.monitoring.coreos.com": failed to call webhook: Post
"https://occne-kube-prom-stack-kube-operator.occne-infra.svc:443/admission-
prometheusrules/mutate?timeout=10s": context deadline exceeded
```

Solution

Retry installation or upgrade using Helm.

2.2.17 Adapters Do Not Restart after Rollback

Scenario:

When upgraded from one version to another version and created a new adapter in new version, the new adapter is still present even after rollback.

Problem:

When rolled back to an older version, every resource in OCNADD should go back to their previous state. So, any adapter resources created in new version should be deleted as well as they didn't exist before in older versions.

Solution:

If they failed to get deleted on their own, use the following command to delete all the resources of the adapter manually:

```
$ kubectl delete service,deploy,hpa <adapter-name> -n ocnadd-deploy
```

2.2.18 Adapters Do Not Restart after Rollback

Scenario:

When Data Feeds are created in the source release and an upgrade is performed to the latest release later if the Rollback to the previous release was performed then the Adapters pods restart is expected.

Problem:

The Adapters do not restart. The Admin Service throws the following exception:

```
io.fabric8.kubernetes.client.KubernetesClientException: Operation: [update]
for kind: [Deployment]  with name: [app-http-adapter]  in namespace: [ocnadd-
deploy]  failed.
    at
io.fabric8.kubernetes.client.KubernetesClientException.launderThrowable(Kubern
etesClientException.java:159) ~[kubernetes-client-api-6.5.1.jar!/:?]
    at
io.fabric8.kubernetes.client.dsl.internal.HasMetadataOperation.update(HasMetad
ataOperation.java:133) ~[kubernetes-client-6.5.1.jar!/:?]
    at
io.fabric8.kubernetes.client.dsl.internal.HasMetadataOperation.update(HasMetad
ataOperation.java:109) ~[kubernetes-client-6.5.1.jar!/:?]
    at
io.fabric8.kubernetes.client.dsl.internal.HasMetadataOperation.update(HasMetad
ataOperation.java:39) ~[kubernetes-client-6.5.1.jar!/:?]
    at
com.oracle.cgbu.cne.ocnadd.admin.svc.service.consumerAdapter.ConsumerAdapterSe
rvice.lambda$ugradeConsumerAdapterOnStart$1(ConsumerAdapterService.java:1068)
~[classes!/:2.2.3]
    at java.util.ArrayList.forEach(ArrayList.java:1511) ~[?:?]
    at
com.oracle.cgbu.cne.ocnadd.admin.svc.service.consumerAdapter.ConsumerAdapterSe
rvice.ugradeConsumerAdapterOnStart(ConsumerAdapterService.java:1022)
~[classes!/:2.2.3]
```

Solution:

Restart the Admin Service after the rollback, which in turn will restart the Adapters to revert them to the older version.

2.2.19 Third-Party Endpoint DOWN State and New Feed Creation

Scenario:

When a third-party endpoint is in a DOWN state and a new third-party Feed (HTTP/ SYNTHETIC) is created with the "Proceed with Latest Data" configuration, data streaming is

expected to resume once the third-party endpoint becomes available and connectivity is established.

Problem

A third-party endpoint is in a DOWN state and a new third-party Feed (HTTP/SYNTHETIC) is created.

Solution

It is recommended to use "Resume from Point of Failure" configuration in case of third-party endpoint unavailability during feed creation.

2.2.20 Adapter Feed Not Coming Up After Rollback

Scenario:

When Data Feeds are created in the previous release and an upgrade is performed to the latest release, and in the latest release both the data feeds are deleted and rollback was carried out to the previous release. Then the feeds that are created in the older Release should have come back after Rollback.

Problem

- After the Rollback only one of the Data Feeds is created back.
- The Data Feeds get created after the rollback, however, it is stuck in "Inactive" state.

Solution:

1. After the rollback, clone the feeds and delete the old feeds.
2. Update the Cloned Feeds with the Data Stream Offset as EARLIEST to avoid data loss.

2.2.21 Adapters pods are in the "init:CrashLoopBackOff" error state after rollback

Scenario:

This issue occurs when the adapter pods, created before or after the upgrade, encounter errors due to missing fixes related to data feed types available only in the latest releases.

Problem:

Adapter pods are stuck in the "init:CrashLoopBackOff" state after a rollback from the latest release to an older release.

Solution Steps:

1. **Delete the Adapter Manually:**
   Run the following command to delete the adapter:

   ```
   kubectl delete service,deploy,hpa <adapter-name> -n ocnadd-deploy
   ```

2. **Clone or Recreate the Data Feed:**
   Clone or recreate the Data Feed again with the same configurations. While creating the data feed, the Data Stream Offset option can be set as "EARLIEST" to avoid data loss.

2.2.22 Configuration Service Pod Goes into The "CrashLoopBackOff" State After Upgrade When the cnDBTier Version is 23.2.x or Above

Scenario:

When OCNADD is upgraded from the previous release to the latest release and the dbTier version is 23.2.x or above.

Problem:

After the upgrade, the configuration service pod goes into the "CrashLoopBackOff" state with DB-related errors in the logs.

Solution:

Update the "`ndb_allow_copying_alter_table`" parameter in dbTier to "`ON`".

```
update ndb_allow_copying_alter_table=ON
```

> **Note**
>
> - If cnDBTier 23.2.x release is installed, set the "`ndb_allow_copying_alter_table`" parameter to '`ON`' in the cnDBTier custom values `dbtier_23.2.0_custom_values_23.2.0.yaml` file before performing any install, upgrade, rollback, or disaster recovery procedure for OCNADD. Revert the parameter to its default value, '`OFF`', after the activity is completed and perform the cnDBTier upgrade to apply the parameter changes.
> - For cnDBTier upgrade instructions, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

2.2.23 Feed/Filter Configurations are Missing From Dashboard After Upgrade

Scenario:

After OCNADD is upgraded from the previous release to the latest release, it may be observed that some feed or filter configurations are missing from dashboard.

Problem:

When this issue occurs after upgrade, the configuration service log will have the following message "*Table definition has changed, please retry transaction*".

Solution:

1. Delete the configuration service pod manually:

   ```
   kubectl delete pod  n ocnadd-deploy <configuration-name>
   ```

2. Check the logs of the new configuration service pod, if "*Table definition has changed, please retry transaction*" is still present in the log, repeat step 1.

2.2.24 Kafka Feed Cannot be Created in Dashboard After Upgrade

Scenario:

After OCNADD is upgraded from the previous release to the latest release, it may be observed that new Kafka feed configurations cannot be created in dashboard.

Problem:

When this issue occurs after upgrade, the configuration service log will have the following message "*Table definition has changed, please retry transaction*".

Solution:

1. Delete the configuration service pod manually:

   ```
   kubectl delete pod  n ocnadd-deploy <configuration-name>
   ```

2. Check the logs of the new configuration service pod, if "*Table definition has changed, please retry transaction*" is still present in the log, repeat step 1.

2.2.25 OCNADD Services Status Not Correct in Dashboard After Upgrade

Scenario:

After OCNADD is upgraded from the previous release to the latest release, it may be observed that the OCNADD microservices status is not correct.

Problem:

When this issue occurs after upgrade, the OCNADD micro services log will have the similar traces as indicated below:

For Statefullset deployment such as zookeeper or Kafka use below commands to check the logs:

```
kubectl exec -it -n <namespace> zookeeper-0 -- bash
[ocnadd@zookeeper-0 ~]$ cat extService.txt |grep -i retry
```

For normal deployment use below command to check the logs:

```
kubectl logs -n <namespace> <ocnadd-service-podname> -f |grep -i retry
```

Services Log:

```
OCL 2023-12-11 07:23:53.919 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 0
OCL 2023-12-11 07:25:12.046 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 1
OCL 2023-12-11 07:27:11.904 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 2
```

```
OCL 2023-12-11 07:30:09.591 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 3
OCL 2023-12-11 07:32:33.906 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 4
OCL 2023-12-11 07:37:26.195 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 5
OCL 2023-12-11 07:54:05.282 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 6
OCL 2023-12-11 08:21:37.994 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 7
OCL 2023-12-11 09:19:50.729 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 8
OCL 2023-12-11 11:24:15.125 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 9
OCL 2023-12-11 15:32:03.699 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 10
OCL 2023-12-11 21:28:51.323 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 11
OCL 2023-12-12 04:01:02.025 [parallel-1] ERROR
com.oracle.cgbu.cne.ocdd.healthmonitoringclient.service.Scheduler - Health
Profile Registration is not successful, retry number 12
```

Solution:

1. Restart all the OCNADD microservices using the command below. This command should be run for all the worker groups and the management group separately in case deployment is upgraded to the centralized mode:

   ```
   kubectl rollout restart -n <namespace> deployment,sts
   ```

2. Check the health status of each of the OCNADD services on OCNADD UI, the status should become active of each of the services.

3
Logs

This chapter explains the process to retrieve the logs and status that can be used for effective troubleshooting.

3.1 Log Levels

Logs register system events along with their date and time of occurrence. They also provide important details about a chain of events that could have led to an error or problem.

A log level helps in defining the severity level of a log message. For OCNADD, the log level of a microservice can be set to any one of the following valid values:

- **TRACE**: A log level that describes events, as a step by step execution of code. This can be ignored during the standard operation, but may be useful during extended debugging sessions.
- **DEBUG**: A log level used for events during software debugging when more granular information is needed.
- **INFO**: A standard log level indicating that something has happened, an application has entered a certain state, etc.
- **WARN**: A log level indicates that something unexpected has happened in the application, a problem, or a situation that might disturb one of the processes. But this does not mean that the application has failed. The WARN level should be used in situations that are unexpected, but the code can continue to work.
- **ERROR**: A log level that should be used when an application hits an issue preventing one or more functionalities from functioning.

Using this information, the logs can be filtered based on the system requirements. For instance, if you want to filter the critical information about your system from the informational log messages, set a filter to view messages with only WARN log level in Kibana.

3.2 Configuring Log Levels

To view logging configurations and update logging levels, check the respective service child `values.yaml`.

Following is an example from the Configurations service:

```
env:
     CONFIGURATION_ROOT_LOG_LEVEL: INFO
     CONFIGURATION_WEB_LOG_LEVEL: INFO
```

Once the service child `values.yaml` is modified, perform helm upgrade for the OCNADD charts.

3.3 Collecting Logs

This section describes the steps to collect logs from PODs or containers. Perform the following steps:

1. Run the following command to get the POD details:

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

2. Collect the logs from the specific pods or containers:

   ```
   $ kubectl logs <podname> -n <namespace>
   ```

   Example:

   ```
   $ kubectl logs ocnaddconfiguration-xxxxxxxxxx-xxxxx -n ocnadd
   ```

3. Store the log in a file using the following command:

   ```
   $ kubectl logs <podname> -n <namespace> > <filename>
   ```

   Example:

   ```
   $ kubectl logs ocnaddconfiguration-xxxxxxxxxx-xxxxx -n ocnadd > logs.txt
   ```

4. (Optional) You can also use the following commands for the log stream with file redirection starting with the last 100 lines of the log:

   ```
   $ kubectl logs <podname> -n <namespace> -f --tail <number of lines> >
   <filename>
   ```

   Example:

   ```
   $ kubectl logs ocnaddconfiguration-xxxxxxxxxx-xxxxx -n ocnadd -f --tail
   100 > logs.txt
   ```

For information on the OCNADD GUI user logs, see *Oracle Communications Cloud Native Configuration Console Troubleshooting Guide*.

3.4 Collect logs using Deployment Data Collector Tool

Perform this procedure to start the NF Deployment Data Collector module and generate the tarballs. If the user does not specify the output storage path, then this module generates the output in the same directory where the module is executed.

nfDataCapture.sh is a script which can be used for collecting all required logs from NF deployment for debugging issues. The script will collect logs from all Micro-Service PODs of specified helm input, helm deployment details,the status, description of all the kafka topics, offset details server properties and description of all the pods, services and events.

- Ensure that you have appropriate privileges to access the system and execute kubectl and helm commands.

- Perform this procedure on the same machine where the OCNADD is deployed using helm or kubectl.
- Execute the **chmod +x nfDataCapture.sh** command on the tool to provide the executable permission.
- Execute the following command to start the module:

```
  ./nfDataCapture.sh -n|--k8Namespace=[K8 Namespace] -u|--username=[User
    Name] -p|--password=[Password] -k|--kubectl=[KUBE_SCRIPT_NAME]  -h|--
helm=[HELM_SCRIPT_NAME]
     -s|--size=[SIZE_OF_EACH_TARBALL]  -d|--cnDBTierStatus=[CN DB TIER
STATUS]
    -x|--kafkaDetails=[KAFKA  DETAILS] -b|--binlogCollectionStatus=[BIN
LOG COLLECTION STATUS]
    -o|--toolOutputPath -helm3=false
```

Examples:

```
./nfDataCapture.sh -k="kubectl --kubeconfig=admin.conf" -h="helm --kubeconfig
admin.conf" -n=ocnrf -s=5M -o=/tmp/
```

```
./nfDataCapture.sh -n=ocnadd -s=5M -o=/tmp/
```

```
./nfDataCapture.sh -n=ocnadd
```

```
./nfDataCapture.sh -n=ocnadd -x=false
```

```
./nfDataCapture.sh -n=ocnadd -helm3=true
```

```
./nfDataCapture.sh -n=ocnadd -u=username -p=password -s=5M -o=/home/root/
datacollector/data -helm3=true
```

```
./nfDataCapture.sh -n=ocnadd -u=username -p=password -s=5M -o=/home/root/
datacollector/data -helm3=true -b=false
```

```
./nfDataCapture.sh -n=ocnadd -u=username -p=password -s=5M -o=/home/root/
datacollector/data -helm3=true -d=true -b=false
```

> **Note**
>
> Default size of tarball generated will be 10M, if not provided, and default location of output will be tool working directory.
>
> Kafka Detailed Staus will be by default true and if we do not want to collect the details we have to pass it as false.
>
> By default, helm2 is used. Use proper argument in command to use helm3.

> **Note**
>
> If the database is not in same namespace , then the script should be run again for the namespace in which database is deployed to capture the database related logs.

- Only if the size of the tar [example: ocnadd.debugData.2023.01.16_09.15.01.tar.gz] generated is greater than "SIZE_OF_EACH_TARBALL" specified in the command ,tar is split into several tarball based on the size specified.
- After execution of command, tar-balls will be created based on size specified in the following format:
  `<namespace>.debugData.<timestamp>`

Example:

```
ocnadd.debugData.2023.01.16_09.15.01-part01
```

Each tarball can then be combined into one tarball with the following command:

```
cat ocnadd.debugData.2023.01.16_09.15.01-part* >
onadd.debugData.2023.01.16_09.15.01-combined.tar.gz
```

4 Capturing `tcpdump` from CNE

Perform the following steps to capture `tcpdump` in CNE:

1. Run the following command to identify the worker node for the running pod:

   ```
   $ kubectl get pods -n ocnadd -o wide
   ```

2. Login to the worker node and run the following command to search for the IP address of the pod:

   ```
   $ ip a
   ```

3. Run the following command to start tcpdump on the identified network interface:

   ```
   $ sudo tcpdump -n -s0 -i <interface> w <file-name>.pcap -Z <node-user-name>
   ```

4. Run the following command to change the file permissions:

   ```
   $ chmod 777 <file-name>.pcap
   ```

5. Exit the worker node and run the following command to `scp` the file from the bastion host:

   ```
   $ scp <user-name>@<worker node>:<path-in-workerNode-machine>
   ```

5
OCNADD Alerts

This section provides information on Oracle Communications Network Analytics Data Director (OCNADD) alerts and their configuration.

5.1 Configuring Alerts

This section describes how to configure alerts in OCNADD.

If OCNADD is installed in the CNE setup, all services are monitored by Prometheus by default. Therefore, there are no modifications required in the Helm chart. All the Prometheus alert rules present in helm chart are updated in the Prometheus Server.

> **Note**
>
> Here, the label used to update the Prometheus server is `role: cnc-alerting-rules`, which is added by default in helm charts.

If OCNADD is installed in the TANZU Setup, one of the files needs to be modified in Helm charts with the following parameters.

> **Note**
>
> Update the `release: prom-operator` label with `role: cnc-alerting-rules` in the `ocnadd-alerting-rules.yaml` file.

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

5.2 Alert Forwarding Using Simple Network Management Protocol (SNMP)

OCNADD forwards the Prometheus alerts as Simple Network Management Protocol (SNMP) traps to the southbound SNMP servers. OCNADD uses two SNMP MIB files to generate the traps. The alert manager configuration is modified by updating the *alertmanager.yaml* file. In the *alertmanager.yaml file*, the alerts can be grouped based on podname, alertname, severity, namespace, and so on. The Prometheus alert manager is integrated with Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) *snmp-notifier* service. The external SNMP servers are set up to receive the Prometheus alerts as SNMP traps. The

operator must update the MIB files along with the alert manager file to fetch the SNMP traps in their environment.

> **Note**
>
> - Only a user with *admin* privileges can perform the following procedures.

Alert Manager Configuration

- Run the following command to obtain the Alert Manager Secret configuration from the Bastion Host and save it to a file:

  ```
  $ kubectl get secret alertmanager-occne-kube-prom-stack-kube-alertmanager -
  o yaml -n occne-infra > alertmanager-secret-k8s.yaml
  ```

  Sample output:

  ```
  apiVersion: v1
  data:
    alertmanager.yaml:
  Z2xvYmFsOgogIHJlc29sdmVfdGltZW91dDogNW0KcmVjZWl2ZXJzOgotIG5hbWU6IGRlZmF1bHQ
  tcmVjZWl2ZXIKICB3ZWJob29rX2NvbmZpZ3M6CiAgLSB1cmw6IGh0dHA6Ly9vY2NuZS1zbm1wLW
  5vdGlmaWVyOjk0NjQvYWxlcnRzCnJvdXRlOgogIGdyb3VwX2J5OgogIC0gam9iCiAgZ3JvdXBfa
  W50ZXJ2YWw6IDVtCiAgZ3JvdXBfd2FpdDogMzBzCiAgcmVjZWl2ZXI6IGRlZmF1bHQtcmVjZWl2
  ZXIKICByZXBlYXRfaW50ZXJ2YWw6IDEyaAogIHJvdXRlczoKICAtIG1hdGNoOgogICAgbWF0Y2h
  ydG5hbWU6IFdhdGNoZG9nCiAgICByZWNlaXZlcjogZGVmYXVsdC1yZWNlaXZlcgp0ZW1wbGF0ZX
  M6Ci0gL2V0Yy9hbGVydG1hbmFnZXIvY29uZmlnLyoudG1wbA==
  kind: Secret
  metadata:
    annotations:
      meta.helm.sh/release-name: occne-kube-prom-stack
      meta.helm.sh/release-namespace: occne-infra
    creationTimestamp: "2022-01-24T22:46:34Z"
    labels:
      app: kube-prometheus-stack-alertmanager
      app.kubernetes.io/instance: occne-kube-prom-stack
      app.kubernetes.io/managed-by: Helm
      app.kubernetes.io/part-of: kube-prometheus-stack
      app.kubernetes.io/version: 18.0.1
      chart: kube-prometheus-stack-18.0.1
      heritage: Helm
      release: occne-kube-prom-stack
    name: alertmanager-occne-kube-prom-stack-kube-alertmanager
    namespace: occne-infra
    resourceVersion: "5175"
    uid: a38eb420-a4d0-4020-a375-ab87421defde
  type: Opaque
  ```

- Extract the Alert Manager configuration. The third line of the *alertmanager.yaml* file contains Alert Manager configuration encoded in base64 format. To extract the Alert Manager configuration, decode the *alertmanager.yaml* file. Run the following command:

  ```
  echo
  'Z2xvYmFsOgogIHJlc29sdmVfdGltZW91dDogNW0KcmVjZWl2ZXJzOgotIG5hbWU6IGRlZmF1bH
  QtcmVjZWl2ZXIKICB3ZWJob29rX2NvbmZpZ3M6CiAgLSB1cmw6IGh0dHA6Ly9vY2NuZS1zbm1wL
  W5vdGlmaWVyOjk0NjQvYWxlcnRzCnJvdXRlOgogIGdyb3VwX2J5OgogIC0gam9iCiAgZ3JvdXBf
  aW50ZXJ2YWw6IDVtCiAgZ3JvdXBfd2FpdDogMzBzCiAgcmVjZWl2ZXI6IGRlZmF1bHQtcmVjZWl
  2ZXIKICByZXBlYXRfaW50ZXJ2YWw6IDEyaAogIHJvdXRlczoKICAtIG1hdGNoOgogICAgICBhbG
  VydG5hbWU6IFdhdGNoZG9nCiAgICByZWNlaXZlcjogZGVmYXVsdC1yZWNlaXZlcgp0ZW1wbGF0Z
  XM6Ci0gL2V0Yy9hbGVydG1hbmFnZXIvY29uZmlnLyoudG1wbA==' | base64 --decode
  ```

  Sample output:

  ```
  global:
    resolve_timeout: 5m
  receivers:
  - name: default-receiver
    webhook_configs:
    - url: http://occne-snmp-notifier:9464/alerts
  route:
    group_by:
    - job
    group_interval: 5m
    group_wait: 30s
    receiver: default-receiver
    repeat_interval: 12h
    routes:
    - match:
        alertname: Watchdog
      receiver: default-receiver
  templates:
  - /etc/alertmanager/config/*.tmpl
  ```

- Update the *alertmanager.yaml* file, alerts can be grouped based on the following:
  - podname
  - alertname
  - severity
  - namespace

  Save the changes to *alertmanager.yaml file*.

  For example:

  ```
  route:
    group_by: [podname, alertname, severity, namespace]
    group_interval: 5m
    group_wait: 30s
    receiver: default-receiver
    repeat_interval: 12h
  ```

- Encode the updated *alertmanager.yaml* file, run the following command:

```
$ cat alertmanager.yaml | base64 -w0
Z2xvYmFsOgogIHJlc29sdmVfdGltZW91dDogNW0KcmVjZWl2ZXJzOgotIG5hbWU6IGRlZmF1bHQ
tcmVjZWl2ZXIKICB3ZWJob29rX2NvbmZpZ3M6CiAgLSB1cmw6IGh0dHA6Ly9vY2NuZS1zbm1wLW
5vdGlmaWVyOjk0NjQvYWxlcnRzCi0gbmFtZTogbmV3LXJlY2VpdmVyLTEKICB3ZWJob29rX2Nvb
mZpZ3M6CiAgLSB1cmw6IGh0dHA6Ly9vY2NuZS1zbm1wLW5vdGlmaWVyLTE6OTQ2NS9hbGVydHMK
cm91dGU6CiAgZ3JvdXBfYnk6CiAgLSBqb2IKICBncm91cF9pbnRlcnZhbDogNW0KICBncm91cF9
3YWl0OiAzMHMKICByZWNlaXZlcjogZGVmYXVsdC1yZWNlaXZlcgogIHJlcGVhdF9pbnRlcnZhD
ogMTJoCiAgcm91dGVzOgogIC0gcmVjZWl2ZXI6IGRlZmF1bHQtcmVjZWl2ZXIKICAgIGdyb3VwX
3dhaXQ6IDMwcwogICAgZ3JvdXBfaW50ZXJ2YWw6IDVtCiAgICByZXBlYXRfaW50ZXJ2YWw6IDEy
aAogIC0gcmVjZWl2ZXI6IG5ldy1yZWNlaXZlci0xCiAgICBncm91cF93YWl0OiAzMHMKICAgIGd
yb3VwX2ludGVydmFsOiA1bQogICAgcmVwZWF0X2ludGVydmFsOiAxMmgKICAtIG1hdGNoOgogIC
AgICBhbGVydG5hbWU6IFdhdGNoZG9nCiAgICByZWNlaXZlcjogZGVmYXVsdC1yZWNlaXZlcgogI
C0gbWF0Y2g6CiAgICAgIGFsZXJ0bmFtZTogV2F0Y2hkb2cKICAgIHJlY2VpdmVyOiBuZXctcmVj
ZWl2ZXItMQp0ZW1wbGF0ZXM6Ci0gL2V0Yy9hbGVydG1hbmFnZXIvY29uZmlnLyoudG1wbAo=
```

- Edit the *alertmanager-secret-k8s.yaml* file created in step 1. Replace the *alertmanager.yaml* encoded content with the output generated in the previous step. For example:

```
$ vi alertmanager-secret-k8s.yaml
apiVersion: v1
data:
  alertmanager.yaml: <paste here the encoded content of alertmanager.yaml>
kind: Secret
metadata:
  annotations:
    meta.helm.sh/release-name: occne-kube-prom-stack
    meta.helm.sh/release-namespace: occne-infra
  creationTimestamp: "2023-02-16T09:44:58Z"
  labels:
    app: kube-prometheus-stack-alertmanager
    app.kubernetes.io/instance: occne-kube-prom-stack
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/part-of: kube-prometheus-stack
    app.kubernetes.io/version: 36.2.0
    chart: kube-prometheus-stack-36.2.0
    heritage: Helm
    release: occne-kube-prom-stack
  name: alertmanager-occne-kube-prom-stack-kube-alertmanager
  namespace: occne-infra
  resourceVersion: "8211"
  uid: 9b499b32-6ad2-4754-8691-70665f9daab4
type: Opaque
```

- Run the following command:

```
$ kubectl apply -f alertmanager-secret-k8s.yaml -n occne-infra
```

Integrate the Alert Manager with snmp-notifier Service

- Update the SNMP client destination in the *occne-snmp-notifier* service with the SNMP destination client IP.

> **Note**
>
> For a persistent client configuration, edit the values of the *snmp-notifier* in Helm charts and perform a Helm upgrade.

Add "warn" in alert severity to receive warning alerts from OCNADD. Run the following command:

```
$ kubectl edit deployment -n occne-infra occne-snmp-notifier

1. update the field "--snmp.destination=<IP>:<port>" inside the args of
container and add the snmp-client destination IP.
   Example:

    spec:
      containers:
      - args:
        - --snmp.destination=10.20.30.40:162

2. "warn" should also be added to the severity list as some of the DD
alerts are raised with severity: warn.
   Exmaple:

    - --alert.severities=critical,major,minor,warning,info,clear,warn
```

- To verify the SNMP notification, see the new notifications in the pod logs of *occne snmp notifier*. Run the following command to see the logs:

  ```
  $ kubectl logs -n occne-infra <occne-snmp-notifier-pod-name>
  ```

  Sample output:

  ```
  10.20.30.50 - - [26/Mar/2023:13:58:14 +0000] "POST /alerts HTTP/1.1" 200 0
  10.20.30.60 - - [26/Mar/2023:14:02:51 +0000] "POST /alerts HTTP/1.1" 200 0
  10.20.30.70 - - [26/Mar/2023:14:03:14 +0000] "POST /alerts HTTP/1.1" 200 0
  10.20.30.80 - - [26/Mar/2023:14:07:51 +0000] "POST /alerts HTTP/1.1" 200 0
  10.20.30.90 - - [26/Mar/2023:14:08:14 +0000] "POST /alerts HTTP/1.1" 200 0
  ```

OCNADD MIB Files

Two OCNADD MIB files are used to generate the traps. The operator has to update the MIB files and the alert manager file to obtain the traps in their environment. The files are:

- *OCNADD-MIB-TC-23.4.0.mib*: This is a top level mib file, where the objects and their data types are defined.
- *OCNADD-MIB-23.4.0.mib*: This file fetches the objects from the top level mib file and based on the alert notification, the objects are selected for display.

> **Note**
>
> MIB files are packaged along with OCNADD Custom Templates. Download the files from MOS. See *Oracle Communications Cloud Native Core Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide* for more information.

5.3 List of Alerts

This section provides detailed information about the alert rules defined for OCNADD.

5.3.1 System Level Alerts

This section lists the system level alerts for OCNADD.

OCNADD_POD_CPU_USAGE_ALERT

Table 5-1 OCNADD_POD_CPU_USAGE_ALERT

Field	Details
Triggering Condition	POD CPU usage is above the set threshold (default 70%)
Severity	Major
Description	OCNADD Pod High CPU usage detected for a continuous period of 5min

Table 5-1 (Cont.) OCNADD_POD_CPU_USAGE_ALERT

Field	Details
Alert Details	**Summary:** 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: CPU usage is {{ "{{" }} $value | printf "%.2f" }} which is above threshold {{ .Values.global.cluster.cpu_threshold }} % ' **Expression**: expr: (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*aggregation.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*2) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*kafka.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*6) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*zookeeper.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*adapter.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*3) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*correlation.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*1) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*filter.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*1) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*configuration.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*1) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*admin.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*1) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*health.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*1) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*alarm.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*1) or

Table 5-1 (Cont.) OCNADD_POD_CPU_USAGE_ALERT

Field	Details
	(sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*ui.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*1)
OID	1.3.6.1.4.1.323.5.3.53.1.29.4002
Metric Used	container_cpu_usage_seconds_total **Note:** This is a Kubernetes metric used for instance availability monitoring. If the metric is not available, use the similar metric as exposed by the monitoring system.
Resolution	The alert gets cleared when the CPU utilization is below the critical threshold. **Note:** The threshold is configurable in the `ocnadd-custom-values-23.4.0.yaml` file. If guidance is required, contact My Oracle Support.

OCNADD_POD_MEMORY_USAGE_ALERT

Table 5-2 OCNADD_POD_MEMORY_USAGE_ALERT

Field	Details
Triggering Condition	POD Memory usage is above set threshold (default 70%)
Severity	Major
Description	OCNADD Pod High Memory usage detected for the continuous period of 5 min

Table 5-2 (Cont.) OCNADD_POD_MEMORY_USAGE_ALERT

Field	Details
Alert Details	**Summary:** 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Memory usage is {{ "{{" }} $value | printf "%.2f" }} which is above threshold {{ .Values.global.cluster.memory_threshold }} % ' **Expression**: expr: (sum(container_memory_working_set_bytes{image!="" , pod=~".*aggregation.*"}) by (pod,namespace) > 3*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*kafka.*"}) by (pod,namespace) > 64*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*zookeeper.*"}) by (pod,namespace) > 1*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*filter.*"}) by (pod,namespace) > 3*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*adapter.*"}) by (pod,namespace) > 4*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*correlation.*"}) by (pod,namespace) > 4*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*configuration.*"}) by (pod,namespace) > 4*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*admin.*"}) by (pod,namespace) > 2*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*health.*"}) by (pod,namespace) > 2*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*alarm.*"}) by (pod,namespace) >

Table 5-2 (Cont.) OCNADD_POD_MEMORY_USAGE_ALERT

Field	Details
	2*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*ui.*"}) by (pod,namespace) > 2*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100)
OID	1.3.6.1.4.1.323.5.3.53.1.29.4005
Metric Used	container_memory_working_set_bytes **Note**: This is a Kubernetes metric used for instance availability monitoring. If the metric is not available, use the similar metric as exposed by the monitoring system.
Resolution	The alert gets cleared when the memory utilization is below the critical threshold. **Note**: The threshold is configurable in the `ocnadd/values.yaml` file. If guidance is required, contact My Oracle Support .

OCNADD_POD_RESTARTED

Table 5-3 OCNADD_POD_RESTARTED

Field	Details
Triggering Condition	A POD has restarted
Severity	Minor
Description	A POD has restarted in last 2 min
Alert Details	**Summary:** 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: A Pod has restarted' **Expression**: expr: kube_pod_container_status_restarts_total{namespace="{{ .Values.global.cluster.nameSpace.name }}" } > 1
OID	1.3.6.1.4.1.323.5.3.51.29.5006
Metric Used	kube_pod_container_status_restarts_total **Note:** This is a Kubernetes metric used for instance availability monitoring. If the metric is not available, use the similar metric asexposed by the monitoring system.

Table 5-3 (Cont.) OCNADD_POD_RESTARTED

Field	Details
Resolution	The alert is cleared automatically if the specific pod is up. **Steps**: 1. Check the application logs. Check for database related failures such as connectivity, Kubernetes secrets, and so on. 2. Run the following command to check orchestration logs for liveness or readiness probe failures: kubectl get po -n <namespace> Note the full name of the pod that is not running, and use it in the following command: kubectl describe pod <desired full pod name> -n <namespace> 3. Check the database status. For more information, see "Oracle Communications Cloud Native Core DBTier User Guide". 4. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support, If guidance is required.

OCNADD_CORRELATION_SVC_DOWN

Table 5-4 OCNADD_CORRELATION_SVC_DOWN

Field	Details
Triggering Condition	The OCNADD Correlation service went down or not accessible
Severity	Critical
Description	OCNADD Correlation service not available for more than 2 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Correlation service is down' **Expression:** expr: up{service=~".*correlation.*"} != 1
OID	1.3.6.1.4.1.323.5.3.53.1.33.2002
Metric Used	'up' Note: This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use a similar metric as exposed by the monitoring system.

Table 5-4 (Cont.) OCNADD_CORRELATION_SVC_DOWN

Field	Details
Resolution	The alert is cleared automatically when the OCNADD Correlation service starts becoming available. Steps: 1. Check for service specific alerts which may be causing the issues with service exposure. 2. Run the following command to check if the pod's status is in the "Running" state: kubectl –n <namespace> get pod If it is not in a running state, capture the pod logs and events. Run the following command to fetch the events as follows: kubectl get events --sortby=.metadata.creationTimestamp -n <namespace> 3. Refer to the application logs and check for database related failures such as connectivity, invalid secrets, and so on. 4. Run the following command to check Helm status and make sure there are no errors: helm status <helm release name of data director> -n<namespace> If it is not in "STATUS: DEPLOYED", then again capture logs and events. 5. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support, If guidance is required.

5.3.2 Application Level Alerts

This section lists the application level alerts for OCNADD.

OCNADD_ADMIN_SVC_DOWN

Table 5-5 OCNADD_ADMIN_SVC_DOWN

Field	Details
Triggering Condition	The OCNADD Admin service is down or not accessible.
Severity	Critical
Description	OCNADD Admin service not available for more than 2 min

Table 5-5 (Cont.) OCNADD_ADMIN_SVC_DOWN

Field	Details
Alert Details	**Summary**: "namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddadminservice service is down' **Expression:** expr: up{service="ocnaddadminservice"} != 1
OID	1.3.6.1.4.1.323.5.3.51.30.2002
Metric Used	'up' Note: This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.
Resolution	The alert is cleared automatically when the OCNADD Admin service start becoming available. Steps: 1. Check for service specific alerts which may be causing the issues with service exposure. 2. Run the following command to check if the pod's status is in "Running" state: kubectl –n <namespace> get pod If it is not in running state, capture the pod logs and events. Run the following command to fetch the events as follows: kubectl get events --sortby=.metadata.creationTimestamp -n <namespace> 3. Refer to the application logs and check for database related failures such as connectivity, invalid secrets, and so on. 4. Run the following command to check Helm status and make sure there are no errors: helm status <helm release name of data director> -n<namespace> If it is not in "STATUS: DEPLOYED", then again capture logs and events. 5. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support, if guidance is required.

OCNADD_ALARM_SVC_DOWN

Table 5-6 OCNADD_ALARM_SVC_DOWN

Field	Details
Triggering Condition	The alarm service is down or not accessible.
Severity	Critical

Table 5-6 (Cont.) OCNADD_ALARM_SVC_DOWN

Field	Details
Description	OCNADD Alarm service not available for more than 2 min
Alert Details	**Summary**: namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddalarm service is down' **Expression:** expr: up{service="ocnaddalarm"} != 1
OID	1.3.6.1.4.1.323.5.3.51.24.2002
Metric Used	'up' **Note** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-6 (Cont.) OCNADD_ALARM_SVC_DOWN

Field	Details
Resolution	The alert is cleared automatically when the OCNADD Alarm service start becoming available. Steps: 1. Check for service specific alerts which may be causing the issues with service exposure. 2. Run the following command to check if the pod's status is in "Running" state: kubectl –n <namespace> get pod If it is not in running state, capture the pod logs and events. Run the following command to fetch the events as follows: kubectl get events --sortby=.metadata.creationTimestamp -n <namespace> 3. Refer to the application logs and check for database related failures such as connectivity, invalid secrets, and so on. 4. Run the following command to check Helm status and make sure there are no errors: helm status <helm release name of data director> -n<namespace> If it is not in "STATUS: DEPLOYED", then again capture logs and events. 5. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support, if guidance is required.

OCNADD_CONFIG_SVC_DOWN

Table 5-7 OCNADD_CONFIG_SVC_DOWN

Field	Details
Triggering Condition	The configuration service is down or not accessible.
Severity	Critical
Description	OCNADD Configuration service not available for more than 2 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddconfiguration service is down' **Expression:** expr: up{service="ocnaddconfiguration"} != 1
OID	1.3.6.1.4.1.323.5.3.51.20.2002

Table 5-7 (Cont.) OCNADD_CONFIG_SVC_DOWN

Field	Details
Metric Used	'up' Note: This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.
Resolution	The alert is cleared automatically when the OCNADD Configuration service start becoming available. Steps: 1. Check for service specific alerts which may be causing the issues with service exposure. 2. Run the following command to check if the pod's status is in "Running" state: kubectl –n <namespace> get pod If it is not in running state, capture the pod logs and events. Run the following command to fetch the events as follows: kubectl get events --sortby=.metadata.creationTimestamp -n <namespace> 3. Refer to the application logs and check for database related failures such as connectivity, invalid secrets, and so on. 4. Run the following command to check Helm status and make sure there are no errors: helm status <helm release name of data director> -n<namespace> If it is not in "STATUS: DEPLOYED", then again capture logs and events. 5. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support, if guidance is required.

OCNADD_CONSUMER_ADAPTER_SVC_DOWN

Table 5-8 OCNADD_CONSUMER_ADAPTER_SVC_DOWN

Field	Details
Triggering Condition	The OCNADD Consumer Adapter service is down or not accessible
Severity	Critical
Description	OCNADD Consumer Adapter service not available for more than 2 min

Table 5-8 (Cont.) OCNADD_CONSUMER_ADAPTER_SVC_DOWN

Field	Details
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Consumer Adapter service is down' **Expression:** expr: up{service=~".*adapter.*"} != 1
OID	1.3.6.1.4.1.323.5.3.51.25.2002
Metric Used	'up' Note: This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.
Resolution	The alert is cleared automatically when the OCNADD Consumer Adapter service start becoming available. Steps: 1. Check for service specific alerts which may be causing the issues with service exposure. 2. Run the following command to check if the pod's status is in "Running" state: kubectl –n <namespace> get pod If it is not in running state, capture the pod logs and events. Run the following command to fetch the events as follows: kubectl get events --sortby=.metadata.creationTimestamp -n <namespace> 3. Refer to the application logs and check for database related failures such as connectivity, invalid secrets, and so on. 4. Run the following command to check Helm status and make sure there are no errors: helm status <helm release name of data director> -n<namespace> If it is not in "STATUS: DEPLOYED", then again capture logs and events. 5. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support, If guidance is required.

OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_0.1PERCENT

Table 5-9 OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_0.1PERCENT

Field	Details
Triggering Condition	The Egress adapter failure rate towards the third-party application is above the configured threshold of 0.1% of the total supported MPS
Severity	Info
Description	Egress external connection failure rate towards third-party application is crossing info threshold of 0.1% in the period of 5 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }} $labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Egress external connection Failure Rate detected above 0.1 Percent of Total Egress external connections' **Expression**: expr: (sum(rate(ocnadd_egress_failed_request_total[5m])) by (namespace))/ (sum(rate(ocnadd_egress_requests_total[5m])) by (namespace)) *100 >= 0.1 < 10
OID	1.3.6.1.4.1.323.5.3.53.1.29.5022
Metric Used	ocnadd_egress_failed_request_total, ocnadd_egress_requests_total
Resolution	The alert is cleared automatically when the failure rate towards third-party consumers goes below the threshold (0.1%) alert level of supported MPS.

OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_10PERCENT

Table 5-10 OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_10PERCENT

Field	Details
Triggering Condition	The Egress adapter failure rate towards the third party application is above the configured threshold of 10% of total supported MPS.
Severity	Minor
Description	Egress external connection failure rate towards third-party application is crossing minor threshold of 10% in the period of 5 min

Table 5-10 (Cont.) OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_10PERCENT

Field	Details
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }} $labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Egress external connection Failure Rate detected above 10 Percent of Total Egress external connections' **Expression:** expr: (sum(rate(ocnadd_egress_failed_request_total[5m])) by (namespace))/ (sum(rate(ocnadd_egress_requests_total[5m])) by (namespace)) *100 >= 10 < 25
OID	1.3.6.1.4.1.323.5.3.53.1.29.5024
Metric Used	ocnadd_egress_failed_request_total, ocnadd_egress_requests_total
Resolution	The alert is cleared automatically when the failure rate towards third party consumer goes below the threshold (10%) alert level of supported MPS.

OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_1PERCENT

Table 5-11 OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_1PERCENT

Field	Details
Triggering Condition	The Egress adapter failure rate towards the third-party application is above the configured threshold of 1% of the total supported MPS.
Severity	Warn
Description	Egress external connection failure rate towards third-party application is crossing the warning threshold of 1% in the period of 5 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{"{{" }} $labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Egress external connection Failure Rate detected above 1 Percent of Total Egress external connections' **Expression:** expr: (sum(rate(ocnadd_egress_failed_request_total[5m])) by (namespace))/ (sum(rate(ocnadd_egress_requests_total[5m])) by (namespace)) *100 >= 1 < 10
OID	1.3.6.1.4.1.323.5.3.53.1.29.5023
Metric Used	ocnadd_egress_failed_request_total, ocnadd_egress_requests_total

Table 5-11 (Cont.) OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_1PERCENT

Field	Details
Resolution	The alert is cleared automatically when the failure rate towards third party consumer goes below the threshold (1%) alert level of supported MPS.

OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_25PERCENT

Table 5-12 OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_25PERCENT

Field	Details
Triggering Condition	The Egress adapter failure rate towards the third party application is above the configured threshold of 25% of total supported MPS.
Severity	Major
Description	Egress external connection failure rate towards third-party application is crossing major threshold of 25% in the period of 5 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }} $labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Egress external connection Failure Rate detected above 25 Percent of Total Egress external connections' **Expression:** expr: (sum(rate(ocnadd_egress_failed_request_total[5m])) by (namespace))/ (sum(rate(ocnadd_egress_requests_total[5m])) by (namespace)) *100 >= 25 < 50
OID	1.3.6.1.4.1.323.5.3.53.1.29.5025
Metric Used	ocnadd_egress_failed_request_total, ocnadd_egress_requests_total
Resolution	The alert is cleared automatically when the failure rate towards third party consumer goes below the threshold (25%) alert level of supported MPS.

OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_50PERCENT

Table 5-13 OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_50PERCENT

Field	Details
Triggering Condition	The Egress adapter failure rate towards the third-party application is above the configured threshold of 50% of total supported MPS.
Severity	Critical
Description	Egress external connection failure rate towards third-party application is crossing critical threshold of 50% in the period of 5 min.

Table 5-13 (Cont.) OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_50PERCENT

Field	Details
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Egress external connection Failure Rate detected above 50 Percent of Total Egress external connections' **Expression:** expr: (sum(rate(ocnadd_egress_failed_request_total[5m])) by (namespace))/ (sum(rate(ocnadd_egress_requests_total[5m])) by (namespace)) *100 >= 50
OID	1.3.6.1.4.1.323.5.3.53.1.29.5026
Metric Used	ocnadd_egress_failed_request_total, ocnadd_egress_requests_total
Resolution	The alert is cleared automatically when the failure rate towards third-party consumer goes below the threshold (50%) alert level of supported MPS.

OCNADD_HEALTH_MONITORING_SVC_DOWN

Table 5-14 OCNADD_HEALTH_MONITORING_SVC_DOWN

Field	Details
Triggering Condition	The health monitoring service is down or not accessible.
Severity	Critical
Description	OCNADD Health monitoring service not available for more than 2 min.
Alert Details	**Summary**: summary: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddhealthmonitoring service is down' **Expression:** expr: up{service="ocnaddhealthmonitoring"} != 1
OID	1.3.6.1.4.1.323.5.3.51.28.2002
Metric Used	'up' Note: This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-14 (Cont.) OCNADD_HEALTH_MONITORING_SVC_DOWN

Field	Details
Resolution	The alert is cleared automatically when the OCNADD Health monitoring service start becoming available. Steps: 1. Check for service specific alerts which may be causing the issues with service exposure. 2. Run the following command to check if the pod's status is in "Running" state: kubectl –n <namespace> get pod If it is not in running state, capture the pod logs and events. Run the following command to fetch the events as follows: kubectl get events --sortby=.metadata.creationTimestamp -n <namespace> 3. Refer to the application logs and check for database related failures such as connectivity, invalid secrets, and so on. 4. Run the following command to check Helm status and make sure there are no errors: helm status <helm release name of data director> -n<namespace> If it is not in "STATUS: DEPLOYED", then again capture logs and events. 5. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support if guidance is required.

OCNADD_INGRESS_TRAFFIC_RATE_DECREASE_SPIKE_10PERCENT

Table 5-15 OCNADD_INGRESS_TRAFFIC_RATE_DECREASE_SPIKE_10PERCENT

Field	Details
Triggering Condition	The ingress traffic decrease is more than 10% of the supported MPS.
Severity	Major
Description	The ingress traffic decrease is more than 10% of the supported MPS in last 5 min.

Table 5-15 (Cont.)
OCNADD_INGRESS_TRAFFIC_RATE_DECREASE_SPIKE_10PERCENT

Field	Details
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Ingress MPS decrease is more than 10 Percent of current supported MPS' **Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace)/ sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m] offset 5m)) by (namespace) <= 0.9
OID	1.3.6.1.4.1.323.5.3.51.29.5013
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the decrease in MPS comes back to lower than 10% of the supported MPS.

OCNADD_INGRESS_TRAFFIC_RATE_INCREASE_SPIKE_10PERCENT

Table 5-16 OCNADD_INGRESS_TRAFFIC_RATE_INCREASE_SPIKE_10PERCENT

Field	Details
Triggering Condition	The ingress traffic increase is more than 10% of the supported MPS.
Severity	Major
Description	The ingress traffic increase is more than 10% of the supported MPS in last 5 min.
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Ingress MPS increase is more than 10 Percent of current supported MPS' **Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace)/ sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m] offset 5m)) by (namespace) >= 1.1
OID	1.3.6.1.4.1.323.5.3.51.29.5013
Metric Used	kafka_stream_processor_node_process_total

Table 5-16 (Cont.)
OCNADD_INGRESS_TRAFFIC_RATE_INCREASE_SPIKE_10PERCENT

Field	Details
Resolution	The alert is cleared automatically when the increase in MPS comes back to lower than 10% of the supported MPS.

OCNADD_KAFKA_PACKET_DROP_THRESHOLD_10PERCENT_MPS

Table 5-17 OCNADD_KAFKA_PACKET_DROP_THRESHOLD_10PERCENT_MPS

Field	Details
Triggering Condition	The packet drop rate in Kafka streams is above the configured critical threshold of 10% of total supported MPS.
Severity	Critical
Description	The packet drop rate in Kafka streams is above the configured critical threshold of 10% of total supported MPS in the period of 5 min.
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Packet Drop rate is above 10% thereshold of Max messages per second:{{ .Values.global.cluster.mps }}' **Expression:** expr: sum(rate(kafka_stream_task_dropped_records_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.1*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.51.29.5011
Metric Used	kafka_stream_task_dropped_records_total
Resolution	The alert is cleared automatically when the packet drop rate goes below the critical threshold (10%) alert level of supported MPS.

OCNADD_KAFKA_PACKET_DROP_THRESHOLD_1PERCENT_MPS

Table 5-18 OCNADD_KAFKA_PACKET_DROP_THRESHOLD_1PERCENT_MPS

Field	Details
Triggering Condition	The packet drop rate in Kafka streams is above the configured major threshold of 1% of total supported MPS.
Severity	Major
Description	The packet drop rate in Kafka streams is above the configured major threshold of 1% of total supported MPS in the period of 5 min.

Table 5-18 (Cont.) OCNADD_KAFKA_PACKET_DROP_THRESHOLD_1PERCENT_MPS

Field	Details
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Packet Drop rate is above 1% thereshold of Max messages per second:{{ .Values.global.cluster.mps }}' **Expression:** expr: sum(rate(kafka_stream_task_dropped_records_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.01*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.51.29.5011
Metric Used	kafka_stream_task_dropped_records_total
Resolution	The alert is cleared automatically when the packet drop rate goes below the major threshold (1%) alert level of supported MPS.

OCNADD_MPS_CRITICAL_INGRESS_THRESHOLD_CROSSED

Table 5-19 OCNADD_MPS_CRITICAL_INGRESS_THRESHOLD_CROSSED

Field	Details
Triggering Condition	The total ingress MPS crossed the critical threshold alert level of 100% of the supported MPS.
Severity	Critical
Description	Total Ingress Message Rate is above configured critical threshold alert (100%) for the period of 5 min.
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Message Rate is above the supported Max messages per second:{{ .Values.global.cluster.mps }}' **Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace) > 1.0*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.51.29.5007
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the the MPS rate goes below the critical threshold alert level of 100% of supported MPS.

OCNADD_MPS_MAJOR_INGRESS_THRESHOLD_CROSSED

Table 5-20 OCNADD_MPS_MAJOR_INGRESS_THRESHOLD_CROSSED

Field	Details
Triggering Condition	The total ingress MPS crossed the major threshold alert level of 95% of the supported MPS.
Severity	Major
Description	Total Ingress Message Rate is above configured major threshold alert (95%) for the period of 5 min.
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Message Rate is above 95 Percent of Max messages per second:{{ .Values.global.cluster.mps }}' **Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.95*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.51.29.5007
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the the MPS rate goes below the major threshold alert level of 95%.

OCNADD_MPS_MINOR_INGRESS_THRESHOLD_CROSSED

Table 5-21 OCNADD_MPS_MINOR_INGRESS_THRESHOLD_CROSSED

Field	Details
Triggering Condition	The total ingress MPS crossed the minor threshold alert level of 90% of the supported MPS.
Severity	Minor
Description	Total Ingress Message Rate is above configured minor threshold alert (90%) for the period of 5 min.
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Message Rate is above 90 Percent of Max messages per second:{{ .Values.global.cluster.mps }}' **Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.9*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.51.29.5007

Table 5-21 (Cont.) OCNADD_MPS_MINOR_INGRESS_THRESHOLD_CROSSED

Field	Details
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the the MPS rate goes below the minor threshold alert level of 90%.

OCNADD_MPS_WARNING_INGRESS_THRESHOLD_CROSSED

Table 5-22 OCNADD_MPS_WARNING_INGRESS_THRESHOLD_CROSSED

Field	Details
Triggering Condition	The total ingress MPS crossed the warning threshold of 80% of the supported MPS.
Severity	Warn
Description	Total Ingress Message Rate is above configured warning threshold (80%) for the period of 5 min.
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Message Rate is above 80 Percent of Max messages per second:{{ .Values.global.cluster.mps }}' **Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.8*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.51.29.5007
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the the MPS rate goes below the warning threshold level of 80%.

OCNADD_NRF_AGGREGATION_SVC_DOWN

Table 5-23 OCNADD_NRF_AGGREGATION_SVC_DOWN

Field	Details
Triggering Condition	The NRF Aggregation service is down or not accessible
Severity	Critical
Description	OCNADD NRF Aggregation service not available for more than 2 min

Table 5-23 (Cont.) OCNADD_NRF_AGGREGATION_SVC_DOWN

Field	Details
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddnrfaggregation service is down' **Expression:** expr: up{service="ocnaddnrfaggregation"} != 1
OID	1.3.6.1.4.1.323.5.3.53.1.31.2002
Metric Used	'up' Note: This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.
Resolution	The alert is cleared automatically when the OCNADD NRF Aggregation service start becoming available. Steps: 1. Check for service specific alerts which may be causing the issues with service exposure. 2. Run the following command to check if the pod's status is in "Running" state: kubectl –n <namespace> get pod If it is not in running state, capture the pod logs and events. Run the following command to fetch the events as follows: kubectl get events --sortby=.metadata.creationTimestamp -n <namespace> 3. Refer to the application logs and check for database related failures such as connectivity, invalid secrets, and so on. 4. Run the following command to check Helm status and make sure there are no errors: helm status <helm release name of data director> -n<namespace> If it is not in "STATUS: DEPLOYED", then again capture logs and events. 5. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support, If guidance is required.

OCNADD_SCP_AGGREGATION_SVC_DOWN

Table 5-24 OCNADD_SCP_AGGREGATION_SVC_DOWN

Field	Details
Triggering Condition	The SCP Aggregation service is down or not accessible

Table 5-24 (Cont.) OCNADD_SCP_AGGREGATION_SVC_DOWN

Field	Details
Severity	Critical
Description	OCNADD SCP Aggregation service not available for more than 2 min.
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddscpaggregation service is down' **Expression:** expr: up{service="ocnaddscpaggregation"} != 1
OID	1.3.6.1.4.1.323.5.3.51.22.2002
Metric Used	'up' Note: This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.
Resolution	The alert is cleared automatically when the OCNADD SCP Aggregation service start becoming available. Steps: 1. Check for service specific alerts which may be causing the issues with service exposure. 2. Run the following command to check if the pod's status is in "Running" state: kubectl –n <namespace> get pod If it is not in running state, capture the pod logs and events. Run the following command to fetch the events as follows: kubectl get events --sortby=.metadata.creationTimestamp -n <namespace> 3. Refer to the application logs and check for database related failures such as connectivity, invalid secrets, and so on. 4. Run the following command to check Helm status and make sure there are no errors: helm status <helm release name of data director> -n<namespace> If it is not in "STATUS: DEPLOYED", then again capture logs and events. 5. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support, If guidance is required.

OCNADD_SEPP_AGGREGATION_SVC_DOWN

Table 5-25 OCNADD_SEPP_AGGREGATION_SVC_DOWN

Field	Details
Triggering Condition	The SEPP Aggregation service is down or not accessible
Severity	Critical
Description	OCNADD SEPP Aggregation service not available for more than 2 min.
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddseppaggregation service is down' **Expression:** expr: up{service="ocnaddseppaggregation"} != 1
OID	1.3.6.1.4.1.323.5.3.53.1.32.2002
Metric Used	'up' Note: This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.
Resolution	The alert is cleared automatically when the OCNADD SEPP Aggregation service start becoming available. Steps: 1. Check for service specific alerts which may be causing the issues with service exposure. 2. Run the following command to check if the pod's status is in "Running" state: kubectl –n <namespace> get pod If it is not in running state, capture the pod logs and events. Run the following command to fetch the events as follows: kubectl get events --sortby=.metadata.creationTimestamp -n <namespace> 3. Refer to the application logs and check for database related failures such as connectivity, invalid secrets, and so on. 4. Run the following command to check Helm status and make sure there are no errors: helm status <helm release name of data director> -n<namespace> If it is not in "STATUS: DEPLOYED", then again capture logs and events. 5. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support, If guidance is required.

OCNADD_MPS_WARNING_EGRESS_THRESHOLD_CROSSED

Table 5-26 OCNADD_MPS_WARNING_EGRESS_THRESHOLD_CROSSED

Field	Description
Triggering Condition	The total egress MPS crossed the warning threshold alert level of 80% of the supported MPS
Severity	Warn
Description	The total Egress Message Rate is above the configured warning threshold alert (80%) for the period of 5 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Message Rate is above 80% of Max messages per second: {{ .Values.global.cluster.mps }}' **Expression:** expr: sum (irate(ocnadd_egress_requests_total[5m])) by (namespace) > 0.80*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5011
Metric Used	ocnadd_egress_requests_total
Resolution	The alert is cleared automatically when the MPS rate goes below the warning threshold alert level of 80% of supported MPS

OCNADD_MPS_MINOR_EGRESS_THRESHOLD_CROSSED

Table 5-27 OCNADD_MPS_MINOR_EGRESS_THRESHOLD_CROSSED

Field	Description
Triggering Condition	The total egress MPS crossed the minor threshold alert level of 90% of the supported MPS
Severity	Minor
Description	The total Egress Message Rate is above the configured minor threshold alert (90%) for the period of 5 min

Table 5-27 (Cont.) OCNADD_MPS_MINOR_EGRESS_THRESHOLD_CROSSED

Field	Description
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Message Rate is above 90% of Max messages per second: {{ .Values.global.cluster.mps }}' **Expression:** expr: sum(irate(ocnadd_egress_requests_total[5m])) by (namespace) > 0.90*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5012
Metric Used	ocnadd_egress_requests_total
Resolution	The alert is cleared automatically when the MPS rate goes below the minor threshold alert level of 90% of supported MPS

OCNADD_MPS_MAJOR_EGRESS_THRESHOLD_CROSSED

Table 5-28 OCNADD_MPS_MAJOR_EGRESS_THRESHOLD_CROSSED

Field	Description
Triggering Condition	The total egress MPS crossed the major threshold alert level of 95% of the supported MPS
Severity	Major
Description	The total Egress Message Rate is above the configured major threshold alert (95%) for the period of 5 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Message Rate is above 95% of Max messages per second: {{ .Values.global.cluster.mps }}' **Expression:** expr: sum (irate(ocnadd_egress_requests_total[5m])) by (namespace) > 0.95*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5013
Metric Used	ocnadd_egress_requests_total
Resolution	The alert is cleared automatically when the MPS rate goes below the major threshold alert level of 95% of supported MPS

OCNADD_MPS_CRITICAL_EGRESS_THRESHOLD_CROSSED

Table 5-29 OCNADD_MPS_CRITICAL_EGRESS_THRESHOLD_CROSSED

Field	Description
Triggering Condition	The total egress MPS crossed the critical threshold alert level of 100% of the supported MPS
Severity	Critical
Description	The total Egress Message Rate is above the configured critical threshold alert (100%) for the period of 5 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Message Rate is above supported Max messages per second:{{ .Values.global.cluster.mps }}' **Expression:** expr: sum (irate(ocnadd_egress_requests_total[5m])) by (namespace) > 1.0*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5014
Metric Used	ocnadd_egress_requests_total
Resolution	The alert is cleared automatically when the MPS rate goes below the critical threshold alert level of 100% of supported MPS

OCNADD_MPS_CRITICAL_EGRESS_THRESHOLD_CROSSED_FOR_A_CONSUMER

Table 5-30 OCNADD_MPS_CRITICAL_EGRESS_THRESHOLD_CROSSED_FOR_A_CONSUMER

Field	Description
Triggering Condition	The total egress MPS crossed the critical threshold alert level of 100% of the supported MPS for a consumer
Severity	Critical
Description	The total Egress Message Rate is above the configured critical threshold alert (100%) for the period of 5 min

Table 5-30 (Cont.)
OCNADD_MPS_CRITICAL_EGRESS_THRESHOLD_CROSSED_FOR_A_CONSUMER

Field	Description
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Message Rate is above supported Max messages per second:{{ .Values.global.cluster.mps }}' **Expression:** expr: sum (rate(ocnadd_egress_requests_total[5m])) by (namespace, instance_identifier) > 1.0*{{ .Values.global.cluster.mps }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5015
Metric Used	ocnadd_egress_requests_total
Resolution	The alert is cleared automatically when the MPS rate goes below the critical threshold alert level of 100% of supported MPS

OCNADD_E2E_AVG_RECORD_LATENCY_WARNING_THRESHOLD_CROSSED

Table 5-31 OCNADD_E2E_AVG_RECORD_LATENCY_WARNING_THRESHOLD_CROSSED

Field	Description
Triggering Condition	The total observed latency is above the configured warning threshold alert level of 80%
Severity	Warn
Description	Average E2E Latency is above the configured warning threshold alert level (80%) for the period of 5 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: E2E Latency is above 80% of Maximum permissable latency {{ .Values.global.cluster.max_latency }} ms' **Expression:** expr: (sum (irate(ocnadd_egress_e2e_request_processing_latency_seconds_sum[5m])) by (namespace)) /(sum (irate(ocnadd_egress_e2e_request_processing_latency_seconds_count[5m])) by (namespace)) > .80*{{ .Values.global.cluster.max_latency }} <= .90*{{ .Values.global.cluster.max_latency }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5016

Table 5-31 (Cont.)
OCNADD_E2E_AVG_RECORD_LATENCY_WARNING_THRESHOLD_CROSSED

Field	Description
Metric Used	ocnadd_egress_e2e_request_processing_latency_seconds_sum, ocnadd_egress_e2e_request_processing_latency_seconds_count
Resolution	The alert is cleared automatically when the average latency goes below the warning threshold alert level of 80% of permissible latency

OCNADD_E2E_AVG_RECORD_LATENCY_MINOR_THRESHOLD_CROSSED

Table 5-32 OCNADD_E2E_AVG_RECORD_LATENCY_MINOR_THRESHOLD_CROSSED

Field	Description
Triggering Condition	The total observed latency is above the configured minor threshold alert level of 90%
Severity	Minor
Description	Average E2E Latency is above the configured minor threshold alert level (90%) for the period of 5 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: E2E Latency is above 90% of Maximum permissable latency {{ .Values.global.cluster.max_latency }} ms' **Expression:** expr: (sum (irate(ocnadd_egress_e2e_request_processing_latency_seconds_sum[5m])) by (namespace)) /(sum (irate(ocnadd_egress_e2e_request_processing_latency_seconds_count[5m])) by (namespace)) > .90*{{ .Values.global.cluster.max_latency }} <= 0.95*{{ .Values.global.cluster.max_latency }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5017
Metric Used	ocnadd_egress_e2e_request_processing_latency_seconds_sum, ocnadd_egress_e2e_request_processing_latency_seconds_count
Resolution	The alert is cleared automatically when the average latency goes below the minor threshold alert level of 90% of permissible latency

OCNADD_E2E_AVG_RECORD_LATENCY_MAJOR_THRESHOLD_CROSSED

Table 5-33 OCNADD_E2E_AVG_RECORD_LATENCY_MAJOR_THRESHOLD_CROSSED

Field	Description
Triggering Condition	The total observed latency is above the configured major threshold alert level of 95%
Severity	Major
Description	Average E2E Latency is above the configured minor threshold alert level (95%) for the period of 5 min
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: E2E Latency is above 95% of Maximum permissable latency {{ .Values.global.cluster.max_latency }} ms' **Expression:** expr: (sum (irate(ocnadd_egress_e2e_request_processing_latency_seconds_sum[5m])) by (namespace)) /(sum (irate(ocnadd_egress_e2e_request_processing_latency_seconds_count[5m])) by (namespace)) > .95*{{ .Values.global.cluster.max_latency }} <= 1.0*{{ .Values.global.cluster.max_latency }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5018
Metric Used	ocnadd_egress_e2e_request_processing_latency_seconds_sum, ocnadd_egress_e2e_request_processing_latency_seconds_count
Resolution	The alert is cleared automatically when the average latency goes below the major threshold alert level of 95% of permissible latency

OCNADD_E2E_AVG_RECORD_LATENCY_CRITICAL_THRESHOLD_CROSSED

Table 5-34 OCNADD_E2E_AVG_RECORD_LATENCY_CRITICAL_THRESHOLD_CROSSED

Field	Description
Triggering Condition	The total observed latency is above the configured critical threshold alert level of 100%
Severity	Critical
Description	Average E2E Latency is above the configured critical threshold alert level (100%) for the period of 5 min

Table 5-34 (Cont.)
OCNADD_E2E_AVG_RECORD_LATENCY_CRITICAL_THRESHOLD_CROSSED

Field	Description
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: E2E Latency is above 100% of Maximum permissable latency {{ .Values.global.cluster.max_latency }} ms' **Expression:** expr: (sum (irate(ocnadd_egress_e2e_request_processing_latency_seconds_sum[5m])) by (namespace)) /(sum (irate(ocnadd_egress_e2e_request_processing_latency_seconds_count[5m])) by (namespace)) > 1.0*{{ .Values.global.cluster.max_latency }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5019
Metric Used	ocnadd_egress_e2e_request_processing_latency_seconds_sum, ocnadd_egress_e2e_request_processing_latency_seconds_count
Resolution	The alert is cleared automatically when the average latency goes below the critical threshold alert level of permissible latency

OCNADD_TRANSACTION_SUCCESS_CRITICAL_THRESHOLD_DROPPED

Table 5-35 OCNADD_TRANSACTION_SUCCESS_CRITICAL_THRESHOLD_DROPPED

Field	Description
Triggering Condition	The total transaction success xDRs rate has dropped the critical threshold alert level of 90%
Severity	Critical
Description	The total transaction success xDRs rate has dropped the critical threshold alert level of 90% for the period of 5min

Table 5-35 (Cont.)
OCNADD_TRANSACTION_SUCCESS_CRITICAL_THRESHOLD_DROPPED

Field	Description
Alert Details	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, workergroup: {{ "{{" }}$labels.worker_group }}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Transaction Success Rate is below 90% per hour: {{ .Values.global.cluster.mps }}' **Expression:** expr: sum(irate(ocnadd_total_transactions_total{service=~".*correlation.*",status="SUCCESS"}[5m]))by (namespace,service) / sum(irate(ocnadd_total_transactions_total{service=~".*correlation.*"}[5m]))by (namespace,service) *100 < 90
OID	**1.3.6.1.4.1.323.5.3**.53.1.33.5029
Metric Used	ocnadd_total_transactions_total
Resolution	The alert is cleared automatically when the transaction success rate goes above the critical threshold alert level of 90%