Oracle® Communications

Network Analytics Data Director Troubleshooting Guide

Release 25.2.100
G41330-01
October 2025

Oracle Communications Network Analytics Data Director Troubleshooting Guide, Release 25.2.100

G41330-01

Contents

Preface

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

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

Conventions

The following text conventions are used in this document:

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

My Oracle Support (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 *25.2.1xx*.

Release 25.2.100 - G41330-01, October 2025

- Added Extended Storage with Druid Troubleshooting Scenarios.

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*.

> **Note**
>
> The performance and capacity of the OCNADD system may vary based on the call model, Feature/Interface configuration, and underlying CNE and hardware environment.

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 Core, 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://ocnaddconfiguration:12590/ocnadd-configuration/v1/
  {workerGroup}/topic'
  ```

  If intraTLS is enabled or intraTls & mTLS enabled, then use the following command:

  ```
  curl -k --location  --cert-type P12 --cert /var/securityfiles/keystore/
  clientKeyStore.p12:$OCNADD_SERVER_KS_PASSWORD "https://
  ocnaddconfiguration:12590/ocnadd-configuration/v1/{workerGroup}/topic"
  ```

  > **Note**
  >
  > These commands are in their simple format and display the logs only if ocnaddconfiguration and ocnaddadmin 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-configuration/v1/{workerGroup}/topic
- <ip>:<port>/ocnadd-configuration/v1/{workerGroup}/topic/<topicName>
- <ip>:<port>/ocnadd-configuration/v1/{workerGroup}/broker/bootstrap-server

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 Kraft 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 Kraft 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> kraft-broker-security-kraft-controller-0`

v. `kubectl delete pvc -n <namespace> kraft-broker-security-kraft-controller-1`

vi. `kubectl delete pvc -n <namespace> kraft-broker-security-kraft-controller-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/Kraft 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 --cert-type P12 --cert /var/securityfiles/keystore/
clientKeyStore.p12:$KEYSTORE_PASS --request GET 'https://
ocnaddadminservice:9181/ocnadd-admin/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 --cert-type P12 --cert /var/securityfiles/keystore/
clientKeyStore.p12:$KEYSTORE_PASS --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 --cert-type P12 --cert /var/securityfiles/keystore/
clientKeyStore.p12:$KEYSTORE_PASS --request GET 'https://
ocnaddadminservice:9181/ocnadd-admin/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 --cert-type P12 --cert /var/securityfiles/keystore/
      clientKeyStore.p12:$KEYSTORE_PASS --request GET 'https://
      ocnaddadminservice:9181/ocnadd-admin/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 --cert-type P12 --cert /var/securityfiles/
      keystore/clientKeyStore.p12:$KEYSTORE_PASS --request GET 'https://
      ocnaddadminservice:9181/ocnadd-admin/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 --cert-type P12 --cert /var/securityfiles/
keystore/clientKeyStore.p12:$KEYSTORE_PASS --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 --cert-type P12 --cert /var/securityfiles/
keystore/clientKeyStore.p12:$KEYSTORE_PASS --request GET 'https://
ocnaddadminservice:9181/ocnadd-admin/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 --cert-type P12 --cert /var/securityfiles/
keystore/clientKeyStore.p12:$KEYSTORE_PASS --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 *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*.

2.2.6 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 kafka-broker3 kafka-broker4`.

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 Kraft, 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 kafka-volume-kafka-broker-0 kafka-volume-
kafka-broker-1 kafka-volume-kafka-broker-2 kraft-broker-security-kraft-
controller-0 kraft-broker-security-kraft-controller-1 kraft-broker-security-
kraft-controller-2
```

2.2.7 Kafka Brokers Continuously Restart After the Disk is Full

Problem

While there is no disk space left on the broker or Kraft controller 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-broker kraft-controller
      ```

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 Kraft 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 kafka-volume-kafka-broker-0 kafka-volume-kafka-broker-1 kafka-volume-kafka-broker-2 kraft-broker-security-kraft-controller-0 kraft-broker-security-kraft-controller-1 kraft-broker-security-kraft-controller-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.

2.2.8 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 Kraft controllers to come online before they are started. If the Kraft controller cluster does not come online within the given interval, the broker will start before the Kraft controller and will error out as it does not have access to the Kraft controller. This may Kraft controller may start after the 3 interval as the node may take more time to pull the images due to network issues. Therefore, when the Kraft controller does not come online within the given time this issue may be observed.

2.2.9 Kafka Broker Pod Stuck in Prolonged `Init` State During Rollback

Problem

During the rollback procedure, a Kafka broker instance may get stuck in a prolonged `Init` state, resulting in a significant delay in rollback completion.

Pod Status

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

```
...
http-adapter-6465f66cdf-tzzgx                1/1     Running    0
109m
kafka-broker-0                               2/2     Running    0
111m
kafka-broker-1                               2/2     Running    0
112m
kafka-broker-2                               0/2     Init:0/1   0
2m51s
kafka-broker-3                               2/2     Running    0
3m42s
...
```

Logs in Kafka Broker `Init` Container

```
kubectl logs -n <namespace> kafka-broker-2 -c ocnaddinitcontainer
```

```
1
2
3
4
5
6
...
...
```

Solution

Perform the following steps to resolve the issue:

1. Edit the Kafka broker StatefulSet (STS) using the command below:

   ```
   kubectl edit sts kafka-broker -n <namespace>
   ```

2. Navigate to the `args` section under the `initContainers` block and locate the script responsible for health checks. The original block will look like this:

   ```
   ...
   ...
   208       initContainers:
   209       - args:
   ```

```
210         - -c
211         - "limit=0; while true; do\n  sleep 10;\n  if [[ \"false\" ==
\"true\" ]];
212           then\n  upInstance=$(curl --cert  $CLIENT_CERT_FILE --
key $CLIENT_KEY_FILE
213           -k --silent https://
ocnaddhealthmonitoring.$OCNADD_MGMT_NS:12591/ocnadd-health/v1/metrics
214           2>/dev/null |\n        grep -o
'\"serviceType\":\"ZOOKEEPER\",\"numberOfInstances\":[0-9]*,\"upState\":
[0-9]*'
215           |\n       sed -E 's/.*\"upState\":([0-9]+).*/\\1/');\n
else\n  if [[ \"false\"
216           == \"true\" ]]; then\n  upInstance=$(curl -k https://
ocnaddhealthmonitoring.$OCNADD_MGMT_NS:12591/ocnadd-health/v1/metrics
217           2>/dev/null |\n        grep -o
'\"serviceType\":\"ZOOKEEPER\",\"numberOfInstances\":[0-9]*,\"upState\":
[0-9]*'
218           |\n       sed -E 's/.*\"upState\":([0-9]+).*/\\1/');\n
else\n  upInstance=$(curl
219           http://ocnaddhealthmonitoring.$OCNADD_MGMT_NS:12591/ocnadd-
health/v1/metrics
220           2>/dev/null |\n        grep -o
'\"serviceType\":\"ZOOKEEPER\",\"numberOfInstances\":[0-9]*,\"upState\":
[0-9]*'
221           |\n       sed -E 's/.*\"upState\":([0-9]+).*/\\1/');\n
fi\n  fi   \n  if
222           [[ $upInstance == $ZK_REPLICA ]]; then\n     echo \"All
zookeeper instances
223           up. Kafka pods can start now...\"\n     break;\n  else\n
limit=$((limit
224           + 1));\n     echo $limit;\n  fi;\ndone; openssl pkcs12 -
export -inkey $SERVER_KEY_FILE
225           -in $SERVER_CERT_FILE -out /var/securityfiles/keystore/
keyStore.p12 -password
226           pass:$KS_PASS && openssl pkcs12 -export -
inkey $CLIENT_KEY_FILE -in $CLIENT_CERT_FILE
227           -out $CLIENT_KEY_STORE -password pass:$KS_PASS && keytool -
importcert -file
228           $CA_CERT_FILE -alias ocnaddcacert -keystore $TRUST_STORE -
storetype PKCS12
229           -storepass $TS_PASS -noprompt && keytool -importkeystore -
srckeystore $JAVA_HOME/lib/security/cacerts
230           -srcstoretype JKS -srcstorepass changeit -
destkeystore $TRUST_STORE -deststoretype
231           PKCS12 -deststorepass $TS_PASS -noprompt  \n"
232         command:
233         - /bin/sh
...
...
```

3. Update all occurrences of the `ocnaddhealthmonitoring` endpoint in the command with the following pattern:

   ```
   https://ocnaddhealthmonitoring.$OCNADD_MGMT_NS:12591/ocnadd-health/v1/
   metrics?workerGroup=<namespace>:<site>
   ```

Where:

a. <namespace> is the namespace where the Kafka broker pod is stuck in the Init state

b. <site> is the cluster name where the OCNADD worker group is deployed

Example:

If the namespace is test-wg and the site name is pv-l2, the endpoint should be updated as:

```
...
...
initContainers:
209       - args:
210         - -c
211         - "limit=0; while true; do\n  sleep 10;\n  if [[ \"false\" ==
\"true\" ]];
212           then\n  upInstance=$(curl --cert  $CLIENT_CERT_FILE --
key $CLIENT_KEY_FILE
213           -k --silent https://
ocnaddhealthmonitoring.$OCNADD_MGMT_NS:12591/ocnadd-health/v1/metrics?
workerGroup=test-wg:pv-l2
214           2>/dev/null |\n        grep -o
'\"serviceType\":\"ZOOKEEPER\",\"numberOfInstances\":[0-9]*,\"upState\":
[0-9]*'
215           |\n       sed -E 's/.*\"upState\":([0-9]+).*/\\1/');\n
else\n  if [[ \"false\"
216           == \"true\" ]]; then\n  upInstance=$(curl -k https://
ocnaddhealthmonitoring.$OCNADD_MGMT_NS:12591/ocnadd-health/v1/metrics?
workerGroup=test-wg:pv-l2
217           2>/dev/null |\n        grep -o
'\"serviceType\":\"ZOOKEEPER\",\"numberOfInstances\":[0-9]*,\"upState\":
[0-9]*'
218           |\n       sed -E 's/.*\"upState\":([0-9]+).*/\\1/');\n
else\n  upInstance=$(curl
219           http://ocnaddhealthmonitoring.$OCNADD_MGMT_NS:12591/ocnadd-
health/v1/metrics?workerGroup=test-wg:pv-l2
220           2>/dev/null |\n        grep -o
'\"serviceType\":\"ZOOKEEPER\",\"numberOfInstances\":[0-9]*,\"upState\":
[0-9]*'
221           |\n       sed -E 's/.*\"upState\":([0-9]+).*/\\1/');\n
fi\n  fi   \n  if
222           [[ $upInstance == $ZK_REPLICA ]]; then\n     echo \"All
zookeeper instances
223           up. Kafka pods can start now...\"\n     break;\n  else\n
limit=$((limit
224           + 1));\n     echo $limit;\n  fi;\ndone; openssl pkcs12 -
export -inkey $SERVER_KEY_FILE
225           -in $SERVER_CERT_FILE -out /var/securityfiles/keystore/
keyStore.p12 -password
226           pass:$KS_PASS && openssl pkcs12 -export -
inkey $CLIENT_KEY_FILE -in $CLIENT_CERT_FILE
227           -out $CLIENT_KEY_STORE -password pass:$KS_PASS && keytool -
importcert -file
228           $CA_CERT_FILE -alias ocnaddcacert -keystore $TRUST_STORE -
storetype PKCS12
229           -storepass $TS_PASS -noprompt && keytool -importkeystore -
```

```
srckeystore $JAVA_HOME/lib/security/cacerts
230           -srcstoretype JKS -srcstorepass changeit -
destkeystore $TRUST_STORE -deststoretype
231           PKCS12 -deststorepass $TS_PASS -noprompt  \n"
232         command:
233         - /bin/sh
...
...
```

4. Restart the Kafka broker pod that is stuck in the `Init` state:

   ```
   kubectl delete po <kafka_broker_pod_name> -n <namespace>
   ```

5. Verify the status of all Kafka broker pods to ensure they are in the `Running` state.

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 Data Feeds Do Not delete after Rollback

Scenario:

During an upgrade from one version to another, if any data feeds (Adapter/Correlation) are spawned in the new version, the newly created resources may not be deleted even after a rollback.

Problem:

Upon rolling back to an older version, all resources in OCNADD should revert to their previous states. Consequently, any new resources generated in the upgraded version should be deleted since they did not exist in older versions.

Solution:

If the resources fail to be deleted automatically, use the following command to manually delete them:

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

2.2.18 Data Feeds Do Not Restart after Rollback

Scenario:

When Data Feeds (Adapters/Correlation) are created in the source release, and an upgrade is performed to the latest release, later, if a rollback to the previous release is executed, the restart of respective data feed pods is expected.

Problem:

The Data feed pods do not restart, and the Admin Service reports 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 data feed pods to revert them to the older version.

2.2.19 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
```

2.2.20 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.

2.2.21 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.

2.2.22 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.

2.2.23 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.

2.2.24 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.

2.2.25 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.

2.2.26 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 Kraft controller or Kafka use below commands to check the logs:

```
kubectl exec -it -n <namespace> kafka-broker-0 -- bash [ocnadd@kafka-broker-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.

2.2.27 Alarm for Unable to Transfer File to SFTP Server for Export Configuration

Scenario:

While performing the Export Configuration, the user encounters the following alarm:

Alarm: `OCNADD02009: SFTP service is unreachable`

Solution:

Verify the following points to manage this alarm:

1. **Check the SFTP server IP:** Ensure the SFTP server IP is correct. Verify that the provided IP is neither IPv6 nor FQDN.
2. **Check the Storage Path in the File Location:** Ensure the file path is correct. The file path should be relative; absolute paths are not supported.

2.2.28 OCNADD Two-Site Redundancy Troubleshooting Scenarios

General Troubleshooting

This section provides information to address troubleshooting issues faced post two-site redundancy configuration. The issues and fixes mentioned in this section are related to alarms. After the site configuration is done, the user should move to the Alarm Dashboard and verify if any newly generated alarms are raised for two-site redundancy.

OCNADD02006: Mate Site Down

TYPE: COMMUNICATION

SEVERITY: MAJOR

Scenario:

When either CNE cluster of the mate site or mate site worker group is down or not accessible.

Problem Description:

The CNE cluster or worker group in the mate site is down or inaccessible.

Troubleshooting Steps:

1. Check the status of worker group services in the mate site.
2. Check if the CNE cluster is accessible or not.
3. Verify if nodes are in an isolated zone or not.
4. Run sync after the issue has been fixed.

OCNADD02000: Loss of Connection

TYPE: COMMUNICATION

SEVERITY: MAJOR

Scenario:

Initial connection with Redundancy Agent of mate site is not established.

Problem Description:

Connection could not be established with the mate site Redundancy Agent service due to egress annotation not enabled or cluster communication not supported.

Troubleshooting Steps:

1. Check if LoadBalancer IP of mated site is able to communicate with each other.
   a. Run Exec command in the mate site Redundancy Agent pod:

      ```
      kubectl exec -ti -n <namespace> <mate_redundancyagent_pod_name> -- bash
      ```

   b. Run the following cURL command:

      ```
      curl -k --cert-type P12 --cert /var/securityfiles/keystore/
      serverKeyStore.p12:$OCNADD_SSL_KEY_STORE_PASSWORD --location -X GET
      https://<mate_redundancy_agent_external_ip>:13000
      ```

2. Verify if CNE version requires egress annotation or not.
3. Check the Redundancy Agent service pod logs in the mate site.
4. If there are error logs in Redundancy Agent, then:
   a. Go to custom values of management charts.
   b. Set the value of `global.ocnaddredundancyagent.egress` to `true`.
   c. Run Helm chart upgrade:

      ```
      helm upgrade <release_name> <management_chart_path> -f
      <custom_values_path> -n <management_namespace>
      ```

5. Post-upgrade error logs should not recur.

OCNADD02001: Loss of Heartbeat

TYPE: COMMUNICATION

SEVERITY: MINOR

Scenario:

When heartbeat from mate site Redundancy Agent is not sent to the primary site Redundancy Agent.

Problem Description:

When there is no heartbeat sent from mate site Redundancy Agent to primary site Redundancy Agent, then Loss of Heartbeat is detected.

Troubleshooting Steps:

1. Run curl request from primary site to secondary site Redundancy Agent to verify if calls are received at the Secondary Site:
   a. Run Exec command in the mate site Redundancy Agent pod

      ```
      kubectl exec -ti -n <namespace> <mate_redundancyagent_pod_name> -- bash
      ```

   b. Run the following cURL command:

      ```
      curl -k --cert-type P12 --cert /var/securityfiles/keystore/
      serverKeyStore.p12:$OCNADD_SSL_KEY_STORE_PASSWORD --location -X GET
      https://<mate_redundancy_agent_external_ip>:13000
      ```

2. This should return 4xx error; if 500 response is received, then the connection between two sites is lost.
3. If communication fails, then verify network communication between both setups.
4. Run sync after the issue has been fixed.

OCNADD050018: Consumer Feed Configuration Sync Discrepancy

TYPE: OPERATIONAL_ALARM

SEVERITY: MAJOR

Scenario:

When the mate site configuration is created, both the sites have Consumer Feed Configuration with the same name, but Data Transfer Object (DTO) parameters are different.

Problem Description:

The consumer feed configuration is mismatched between mated worker group pair.

Troubleshooting Steps:

1. Check mate site consumer feed configuration.
2. Update the configuration with the mated site.
3. Run sync option to clear the alarm.

OCNADD050019: Kafka Feed Configuration Sync Discrepancy

TYPE: OPERATIONAL_ALARM

SEVERITY: MAJOR

Scenario:

When the mate site configuration is created, both the sites have Kafka feed configuration with the same name, but DTO parameters are different.

Problem Description:

The Kafka feed configuration is mismatched between mated worker group pair.

Troubleshooting Steps:

1. Check mate site Kafka feed configuration.
2. Update the Kafka feed configuration with the mated site.
3. Run sync option to clear the alarm.

OCNADD050020: Filter Configuration Sync Discrepancy

TYPE: OPERATIONAL_ALARM

SEVERITY: MAJOR

Scenario:

When the mate site configuration is created, both the sites have filter configuration with the same name, but DTO parameters are different.

Problem Description:

The filter configuration is mismatched between mated worker group pair.

Troubleshooting Steps:

1. Check mate site filter configuration.
2. Update the filter configuration with the mated site.
3. Run sync option to clear the alarm.

OCNADD050021: Correlation Configuration Sync Discrepancy

TYPE: OPERATIONAL_ALARM

SEVERITY: MAJOR

Scenario:

When mate site configuration is created, both the sites have Correlation feed configuration with the same name, but DTO parameters are different.

Problem Description:

The correlation configuration is mismatched between mated worker group pair.

Troubleshooting Steps:

1. Check mate site correlation configuration.
2. Update the correlation configuration with the mated site.

3. Run sync option to clear the alarm.

Unable to Sync Consumer Feed/Filter/Correlation/Kafka Feed

Scenario:

Unable to Sync Consumer Feed/Filter/Correlation/Kafka Feed

Problem Description:

Consumer Feed/Filter/Correlation/Kafka Feed configuration is not getting synced and is missing in the local site even after triggering manual sync. This issue persists despite resolving discrepancies encountered during the initial configuration.

Solution:

1. Delete the two-site redundancy configuration and create again.
2. Verify on secondary site if all the Consumer Feed/Filter/Correlation/Kafka Feed has been synced or not.

One of the mated sites is unavailable and user updates Consumer Feed/Filter/Correlation in the available Site

Scenario:

One of the Mated Site is unavailable, and the user updates Consumer Feed/Filter/Correlation in the available Site.

Problem Description:

After creating mate site configuration, the user updates the Consumer Feed/Filter/Correlation when one of the mated sites is unavailable. On updating the Consumer Feed/Filter/Correlation, the change will be reflected only on the available site, and when the other mated site becomes available again, discrepancy alarms will be raised.

Solution:

1. Repeat the Consumer Feed/Filter/Correlation update operation in the primary site.
2. Clear the discrepancy alarm that was raised previously.

One of the mated sites is unavailable and user deletes Consumer Feed/Filter/Correlation in the available Site

Scenario:

One of the mated site is unavailable, and the user deletes Consumer Feed/Filter/Correlation in the available Site.

Problem Description:

After creating mate site configuration, the user deletes the Consumer Feed/Filter/Correlation when one of the mated sites is unavailable. On deleting the Consumer Feed/Filter/Correlation, the change will be reflected only on the available site, and when the other mated site becomes available again, discrepancy alarms will be raised.

Solution:

1. Delete the mate site configuration.
2. Delete the Consumer Feed/Filter/Correlation in both sites if present.
3. Create the mate site configuration again.

2.2.29 Resource Allocation Challenges During DD Installation on OCI

Scenario:

When DD is installed in OCI, some of the services are stuck in a pending state for a long time.

Problem:

Describing pods of pending services using the below command shows insufficient resources available to start the services:

```
kubectl describe po <pod_name> -n <namespace>
```

Solution:

When encountering OKE cluster nodes CPU and memory issues during DD installation, it's essential to address them promptly. Follow these steps:

1. Begin by assessing the utilization of resources on the affected nodes. You can do this by executing the command:

   ```
   kubectl get nodes
   kubectl describe node <node_name>
   ```

2. If the resource utilization exceeds 90%, it indicates potential congestion. To alleviate this, proceed with increasing the node's resources using the following steps:
   a. Manually adjust the CPU and memory allocation for the instance node. Access the Oracle Cloud Infrastructure Console, navigate to "**Instances**," and select "**Instance Details**."
   b. Under "**More Actions**," choose "**Edit**" and then "**Edit Shape**."
   c. Expand the shape settings and adjust the CPU and memory allocation according to the specific requirements.
   d. After making the changes, reboot the instance and wait until it transitions to the "**Running**" state.

> **Note**
>
> - One OCPU (Oracle Compute Unit) is equivalent to two vCPUs.
> - Ensure that the allocated resources remain within the maximum limit of the instance shape.
> - Increasing CPU and memory resources is subject to the limitations of the tenancy's assigned resource allocation.

2.2.30 Transaction Filter Update Takes Few Seconds to Reflect Changes in Traffic Processing

When a user updates a configuration for transaction filters, the DD services (filter service and consumer adapter) promptly receive a notification for the filter condition update; however, it may take a few seconds for the changes to be applied to the actual traffic processing.

2.2.31 Connection Timeout Errors in Configuration Service Logs

Scenario:

The "Connection Timeout Errors" are occasionally logged by the Configuration service when an instance of the service is deleted or goes down during pod restart and sends an unsubscribe request. However, the unsubscribe request to the configuration service may not always be successful, potentially due to network glitches. If the IP reported in these error logs does not belong to any pod, then these entries are treated as stale entries.

Error Message:

Following error logs will be reported in the configuration service logs:

```
reactor.core.Exceptions$ErrorCallbackNotImplemented:
org.springframework.web.reactive.function.client.WebClientRequestException:
connection timed out after 30000 ms: /10.233.92.24:9664
Caused by:
org.springframework.web.reactive.function.client.WebClientRequestException:
connection timed out after 30000 ms: /10.233.92.24:9664
        at
org.springframework.web.reactive.function.client.ExchangeFunctions$DefaultExch
angeFunction.lambda$wrapException$9(ExchangeFunctions.java:136) ~[spring-
webflux-6.1.2.jar!/:6.1.2]
...
...
Caused by: io.netty.channel.ConnectTimeoutException: connection timed out
after 30000 ms: /10.233.92.24:9664
...
```

Solution:

These error messages will not have any impact on the service processing the request. It can be ignored.

2.2.32 Correlation Service Pods Created in 24.3.0 are Not Deleting after Rolling Back to n-1 and n-2 Releases

Scenario:

When upgrading from 24.1.0 or 24.2.0 to 24.3.0, a new correlation service pods are created. Ideally, the pods (along with any storage adapter pods) created in 24.3.0 should delete themselves after the rollback.

Problem:

After a rollback, the correlation pods remain and are not deleted automatically.

Solution:

Manually delete the service, deployment, and HPA of the correlation service and storage adapter service using "`kubectl delete`".

These error messages will not impact the service that is processing the request, and they can be ignored.

2.2.33 High Latency in Adapter Feed Due to High Disk Latency

Scenario:

The data director adapter feed reports high latency.

Problem:

- The Fluentd pod reports multiple restarts and frequently gets killed by `OOMKilled`.
- This results in high disk usage by the Fluentd pods, impacting the Kafka disk write access. The Ceph storage class is shared by all the pods deployed in the cluster.

```
occne-fluentd-opensearch-2hv5p                    1/1     Running   2 (22d ago)          30d
occne-fluentd-opensearch-2kq5v                    1/1     Running   2 (37h ago)          30d
occne-fluentd-opensearch-2lcrx                    1/1     Running   0                    30d
occne-fluentd-opensearch-2zvwg                    1/1     Running   0                    30d
occne-fluentd-opensearch-5wmnx                    1/1     Running   2                    30d
occne-fluentd-opensearch-65c85                    1/1     Running   9 (3d16h ago)        30d
occne-fluentd-opensearch-6xjqq                    1/1     Running   0                    30d
occne-fluentd-opensearch-742wl                    1/1     Running   31 (3h55m ago)       30d
occne-fluentd-opensearch-774s4                    1/1     Running   6 (10d ago)          30d
occne-fluentd-opensearch-7ftvq                    1/1     Running   6 (10d ago)          30d
occne-fluentd-opensearch-7thmq                    1/1     Running   1 (5d13h ago)        30d
occne-fluentd-opensearch-85dfm                    1/1     Running   11 (44h ago)         30d
occne-fluentd-opensearch-879nv                    1/1     Running   3 (7d17h ago)        30d
occne-fluentd-opensearch-8d2mh                    1/1     Running   1 (3d5h ago)         30d
occne-fluentd-opensearch-8j89p                    1/1     Running   4 (3d ago)           30d
occne-fluentd-opensearch-9fhm9                    1/1     Running   1 (14d ago)          30d
occne-fluentd-opensearch-9ll85                    1/1     Running   0                    30d
occne-fluentd-opensearch-9sq65                    1/1     Running   5 (2d8h ago)         30d
occne-fluentd-opensearch-bxdrt                    1/1     Running   2 (20d ago)          30d
occne-fluentd-opensearch-chl9h                    1/1     Running   0                    30d
occne-fluentd-opensearch-cwh7n                    1/1     Running   3322 (25m ago)       30d
occne-fluentd-opensearch-cxnz4                    1/1     Running   0                    30d
occne-fluentd-opensearch-djt7f                    1/1     Running   4369 (4m38s ago)     30d
occne-fluentd-opensearch-fjxk9                    1/1     Running   2 (10d ago)          30d
occne-fluentd-opensearch-gw8wd                    1/1     Running   5 (2d8h ago)         30d
occne-fluentd-opensearch-gxdkd                    1/1     Running   1 (10d ago)          30d
occne-fluentd-opensearch-h6mps                    1/1     Running   0                    30d
occne-fluentd-opensearch-jqtxd                    1/1     Running   54 (12h ago)         30d
occne-fluentd-opensearch-l8lfs                    1/1     Running   0                    30d
occne-fluentd-opensearch-p4fn2                    1/1     Running   1 (27d ago)          30d
occne-fluentd-opensearch-phjwn                    1/1     Running   1 (30d ago)          30d
occne-fluentd-opensearch-pzct5                    1/1     Running   0                    30d
occne-fluentd-opensearch-qfbrd                    1/1     Running   4 (17d ago)          30d
occne-fluentd-opensearch-qx29v                    1/1     Running   0                    30d
occne-fluentd-opensearch-qx4zh                    1/1     Running   28 (17h ago)         30d
occne-fluentd-opensearch-tn772                    1/1     Running   1 (25d ago)          30d
occne-fluentd-opensearch-v7ql4                    1/1     Running   2 (14d ago)          30d
occne-fluentd-opensearch-wt5pf                    1/1     Running   1 (9d ago)           30d
occne-fluentd-opensearch-wx7qd                    1/1     Running   5 (6d18h ago)        30d
occne-fluentd-opensearch-x8c29                    1/1     Running   0                    30d
occne-fluentd-opensearch-xlt9q                    1/1     Running   2 (7d12h ago)        30d
occne-fluentd-opensearch-zljpf                    1/1     Running   23 (22m ago)         30d
occne-fluentd-opensearch-zm456                    1/1     Running   1 (23d ago)          30d
occne-fluentd-opensearch-zps5n                    1/1     Running   1 (21d ago)          30d
```

Details:

```
Containers:
  fluentd:
    Container ID:   containerd://
```

```
a586d8342a087b5f910134c0ed796e1acc6c5ababe1c6fcefc3516ac2fea7bb9
    Image:          occne-repo-host:5000/docker.io/fluent/fluentd-kubernetes-
daemonset:v1.16.2-debian-opensearch-1.0
    Image ID:       occne-repo-host:5000/docker.io/fluent/fluentd-kubernetes-
daemonset@sha256:02113369069664997020 4ed525f3395c17601b63dcf1fe9c269378dcd0d80
ae1
    Port:           24231/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 23 May 2024 10:42:00 +0000
    Last State:     Terminated
      Reason:       OOMKilled
```

Solution:

- Increase the memory allocation for the Fluentd pods to resolve the `OOMKilled` issue.
- Allow all the Fluentd pods to restart after the memory increase.

2.2.34 Kafka Broker Goes into "CrashLoopBackOff" State During Very Low-Throughput Traffic

Scenario

The Kafka broker enters a "CrashLoopBackOff" state during very low-throughput traffic due to the "No space left on device" error. This occurs when the broker's Persistent Volume Claim (PVC) is full, preventing the broker from starting its process.

Error Code or Error Message

```
[2024-05-20 07:04:11,964] ERROR Error while loading log dir /kafka/logdir/
kafka-logs (kafka.log.LogManager)
java.io.IOException: No space left on device
    at java.base/sun.nio.ch.FileDispatcherImpl.write0(Native Method)
    at java.base/
sun.nio.ch.FileDispatcherImpl.write(FileDispatcherImpl.java:62)
    at java.base/sun.nio.ch.IOUtil.writeFromNativeBuffer(IOUtil.java:132)
    at java.base/sun.nio.ch.IOUtil.write(IOUtil.java:97)
    at java.base/sun.nio.ch.IOUtil.write(IOUtil.java:67)
    at java.base/sun.nio.ch.FileChannelImpl.write(FileChannelImpl.java:288)
    at
org.apache.kafka.storage.internals.log.ProducerStateManager.writeSnapshot(Prod
ucerStateManager.java:683)
    at
org.apache.kafka.storage.internals.log.ProducerStateManager.takeSnapshot(Produ
cerStateManager.java:437)
    at kafka.log.LogLoader.recoverSegment(LogLoader.scala:373)
    at kafka.log.LogLoader.recoverLog(LogLoader.scala:425)
    at kafka.log.LogLoader.load(LogLoader.scala:163)
    at kafka.log.UnifiedLog$.apply(UnifiedLog.scala:1804)
    at kafka.log.LogManager.loadLog(LogManager.scala:278)
    at kafka.log.LogManager.$anonfun$loadLogs$15(LogManager.scala:421)
    at java.base/
java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:539)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at java.base/
```

```
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136
)
    at java.base/
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635
)
    at java.base/java.lang.Thread.run(Thread.java:842)
```

Solution

This issue can be resolved by following the steps below:

1. **Increase the PVC size of your Kafka broker** so that the Kafka process can be started normally. For more details on how to increase the size of the PVC, see the "Steps to Increase the Size of Existing PVC" section in the *Oracle Communications Network Analytics Data Director User Guide*.
   For example, if you have configured a small PVC size for your Kafka broker, say 8Gi, you can increase the size to 11Gi for the Kafka broker process to start normally.
2. **Run the following command to exec into the Kafka broker pod:**

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

3. Navigate to the "kafka/bin" folder and apply the configuration "segment.ms" for your Kafka topics by running the following command:
   If ACL is not enabled, then run the following command:

   ```
   ./kafka-configs.sh --bootstrap-server localhost:9092 --alter --entity-type
   topics --entity-name <TOPIC_NAME> --add-config segment.ms=<Time in ms>
   ```

   If ACL is enabled, then run the following command:

   ```
   /kafka-configs.sh --bootstrap-server kafka-broker:9094 --alter --entity-
   type topics --entity-name <TOPIC_NAME> --add-config segment.ms=<Time in
   ms> --command-config ../../admin.properties
   ```

> **Note**
>
> - The configuration has to be applied to all the OCNADD Kafka topics.
> - "Time in ms" can be any small value, such as 600,000 (10 minutes) or 720,000 (12 minutes).

2.2.35 Database Space Full Preventing Configuration and Subscription from Being Saved in Database

Scenario:

The Data Director has separate schemas for configuration, alarms, health monitoring, and xDRs. However, all these schemas share a common

```
cnDBTier
```

instance. The database space can become full in the following scenarios:

1. **Alarm Flooding**: Too many alarms are raised due to repetitive failure scenarios, such as network communication failure.
2. **Excessive xDR Generation**: Too many xDRs are generated either due to a traffic spike or network misconfiguration, leading to an increased number of xDRs.

Problem:

When this issue occurs, new entries in database tables within various schemas cannot be created. The following error might be seen in the logs of the Data Director export service:

```
SQL Error: 1114, SQLState: HY000 The table 'SUBSCRIPTION' is full
```

Solution:

- Free up space in the `AlarmDB` by deleting older alarm entries.
- Free up space in the `xDRDB` by deleting older xDRs.

Steps to Resolve:

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

> **Note**
>
> Use the namespace in which the `cnDBTier` is deployed. For example, the `occne-cndbtier` namespace is used. The default container name is
>
> `ndbmysqld-0`.

```
kubectl -n occne-cndbtier exec -it ndbmysqld-0 -- bash
```

Step 2: Run the following command to log in to the MySQL server using the MySQL client

```
$ mysql -h 127.0.0.1 -uocdd -p
$ Enter password:
```

Step 3: Free up space in the Alarm DB by deleting older alarms and event entries.

1. Fetch the required events from the `EVENTS` table:

   ```
   SELECT * FROM <alarm_db_name>.EVENTS
   WHERE ALARM_ID IN (
       SELECT ALARM_ID FROM <alarm_db_name>.ALARM
       WHERE (ALARM_SEVERITY = 'INFO')
       OR (ALARM_STATUS = 'CLEARED' AND RAISED_TIME <= NOW() - INTERVAL 7 DAY)
   )
   LIMIT 5000;
   ```

2. Delete the required events from the `EVENTS` table:

   ```
   DELETE FROM <alarm_db_name>.EVENTS
   WHERE ALARM_ID IN (
       SELECT ALARM_ID FROM <alarm_db_name>.ALARM
   ```

```
    WHERE (ALARM_SEVERITY = 'INFO')
    OR (ALARM_STATUS = 'CLEARED' AND RAISED_TIME <= NOW() - INTERVAL 7 DAY)
)
LIMIT 5000;
```

3. Fetch the required alarms from the `ALARM` table:

```
SELECT * FROM <alarm_db_name>.ALARM
WHERE (ALARM_SEVERITY = 'INFO')
OR (ALARM_STATUS = 'CLEARED' AND RAISED_TIME <= NOW() - INTERVAL 7 DAY)
LIMIT 5000;
```

4. Delete the required alarms from the `ALARM` table:

```
DELETE FROM <alarm_db_name>.ALARM
WHERE (ALARM_SEVERITY = 'INFO')
OR (ALARM_STATUS = 'CLEARED' AND RAISED_TIME <= NOW() - INTERVAL 7 DAY)
LIMIT 5000;
```

5. Commit the changes:

```
commit;
```

Step 4: Free up space in the xDR DB by deleting older xDRs. The xDR table name is the same as the Correlation Configuration name (for example, `<correlationname>XDR`). Replace `<xDR_table_name>` with `<correlationname>XDR` in the steps below:

1. Fetch the required xDRs from the corresponding xDR table:

```
SELECT * FROM <storageadapter_schema>.<correlationname>XDR
WHERE beginTime <= NOW() - INTERVAL 6 HOUR
LIMIT 5000;
```

2. Delete the required xDRs from the corresponding xDR table:

```
DELETE FROM <storageadapter_schema>.<correlationname>XDR
WHERE beginTime <= NOW() - INTERVAL 6 HOUR
LIMIT 5000;
```

3. Commit the changes:

```
commit;
```

> **Note**
>
> The limit in the above queries can be increased from '5000' as needed.

2.2.36 Invalid Subscription Entry in the Subscription Table

Scenario:

OCNADD supports a subscription mechanism for the configuration information exchange within Data Director services. OCNADD configuration service maintains the subscription table,

and all the other services like correlation, filter, export, consumer adapter, etc., subscribe to the configuration service on startup and receive configuration updates via notifications from the configuration service. The configuration service manages the subscription entry cleanup from the database and its local cache. It deletes the subscription entry of the service which is not available or which has been unregistered or unsubscribed with the subscription service.

In the older version of OCNADD, for example, 24.2.x/24.3.x, it is possible that the older subscription entry may remain in the subscription table as the cleanup mechanism was not available or the clean unsubscription could not happen. This results in the older entries remaining in the subscription table. These older entries may result in delayed configuration notifications sent to the services that have subscribed to the configuration service when the configuration service is restarted during an upgrade/rollback. The other services, like correlation, export, or adapter, may not receive the notification in a timely manner and could not perform their functionality for a prolonged time duration.

Problem:

When this issue occurs, older entries in the subscription table can be observed. This is typically expected in rollback scenarios to older releases (e.g., from release 25.1.0 to 24.2.x).

For example, the following is a stale subscription entry for the export service in the subscription table:

```
| 6c0b06a8-06e1-47f3-91b3-6e3828284cca | https://10.233.82.205:12595/ocnadd-
export/v1/export/notification | EXPORTSERVICE-1 | EXPORT_SERVICE | EXPORT-1 |
kp-mgmt:ocnadd-vcne3 |


mysql> select * from SUBSCRIPTION;
+--------------------------------------
+-------------------------------------------------------------------
+-------------------------+-------------------------+--------------------
+----------------------+
| SUBSCRIPTION_ID                      |
NOTIFICATION_URI                                                    |
SERVICE_ID              | SERVICE_TYPE             | TARGET_CONSUMERNAME |
WORKER_GROUP         |
+--------------------------------------
+-------------------------------------------------------------------
+-------------------------+-------------------------+--------------------
+----------------------+
| 980705eb-54ac-4148-9962-e587e80f4328 | https://10.233.113.164:9664/ocnadd-
storageadapter/v1/notification  | feed1-storage-adapter-1 |
STORAGE_ADAPTER_SERVICE  | feed1               | kp-mgmt:ocnadd-vcne3 |
| 012852ec-97f2-483c-a625-02d3c7f2c32d | https://10.233.96.131:12585/ocnadd-
egress-filter/v1/notifications  | FILTER_SERVICE-1        |
FILTER_SERVICE           | egress-filter       | kp-mgmt:ocnadd-vcne3 |
| 467ef3b8-f26f-443e-a1e0-11b0a308e33a | https://10.233.75.161:9664/ocnadd-
correlation/v1/notification      | feed1-correlation-1     |
CORRELATION_SERVICE      | feed1               | kp-mgmt:ocnadd-vcne3 |
| d75fc1f7-82fe-442c-a528-83b7b35d6078 | https://10.233.109.96:9664/ocnadd-
correlation/v1/notification      | feed1-correlation-1     |
CORRELATION_SERVICE      | feed1               | kp-mgmt:ocnadd-vcne3 |
| 3c353e06-0635-4c4a-8c32-d13ac18069f9 | https://10.233.97.134:9182/ocnadd-
consumeradapter/v5/notifications | A-ORA-ADAPTER-1         |
CONSUMER_ADAPTER_SERVICE | a-ora               | kp-mgmt:ocnadd-vcne3 |
| c1f0541f-ae36-46a2-b41f-09e3fb247505 | https://10.233.89.232:9182/ocnadd-
```

```
consumeradapter/v5/notifications | A-ORA-ADAPTER-2         |
CONSUMER_ADAPTER_SERVICE | a-ora               | kp-mgmt:ocnadd-vcne3 |
| 6c0b06a8-06e1-47f3-91b3-6e3828284cca | https://10.233.82.205:12595/ocnadd-
export/v1/export/notification  | EXPORTSERVICE-1         |
EXPORT_SERVICE           | EXPORT-1            | kp-mgmt:ocnadd-vcne3 |
| a38e52ef-fd69-4ffe-b4c0-b70ab05c9301 | https://10.233.109.77:9664/ocnadd-
correlation/v1/notification      | feed1-correlation-1     |
CORRELATION_SERVICE      | feed1               | kp-mgmt:ocnadd-vcne3 |
+--------------------------------------
+-------------------------------------------------------------------
+-------------------------+-------------------------+--------------------
+---------------------+
8 rows in set (0.01 sec)
```

The following logs could be seen in the configuration service:

```
OCL [36m2024-12-04 07:31:03.237[0;39m [1;30m[parallel-1][0;39m [31mWARN
[0;39m [35mc.o.c.c.o.c.e.s.n.ExportNotificationService[0;39m - Sending Export
Config notification to uri: https://10.233.82.205:12595/ocnadd-export/v1/
export/notification is not successful. Retry number 1

OCL [36m2024-12-04 07:31:33.234[0;39m [1;30m[parallel-1][0;39m [31mWARN
[0;39m [35mc.o.c.c.o.c.e.s.n.ExportNotificationService[0;39m - Sending Export
Config notification to uri: https://10.233.82.205:12595/ocnadd-export/v1/
export/notification is not successful. Retry number 2

OCL [36m2024-12-04 07:32:03.240[0;39m [1;30m[parallel-1][0;39m [31mWARN
[0;39m [35mc.o.c.c.o.c.e.s.n.ExportNotificationService[0;39m - Sending Export
Config notification to uri: https://10.233.82.205:12595/ocnadd-export/v1/
export/notification is not successful. Retry number 3
```

The configuration service is not able to create the connection with the IP of the POD corresponding to the older subscription entry as that POD no more exists, the following connection timeout can also be seen in the configuration service log:

```
OCL [36m2024-12-04 07:33:02.252[0;39m [1;30m[default-nioEventLoopGroup-1-2]
[0;39m [1;31mERROR[0;39m [35mclient-logger[0;39m - Failed to connect to
remote io.netty.channel.ConnectTimeoutException: connection timed out after
30000 ms: /10.233.82.205:12595 at
io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe$1.run(AbstractNioCha
nnel.java:263) at
io.netty.util.concurrent.PromiseTask.runTask(PromiseTask.java:98) at
io.netty.util.concurrent.ScheduledFutureTask.run(ScheduledFutureTask.java:153)
 at
io.netty.util.concurrent.AbstractEventExecutor.runTask(AbstractEventExecutor.j
ava:173) at
io.netty.util.concurrent.AbstractEventExecutor.safeExecute(AbstractEventExecut
or.java:166) at
io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEve
ntExecutor.java:469) at
io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:569) at
io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExec
utor.java:994) at
io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74) at
```

```
io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.j
ava:30) at java.base/java.lang.Thread.run(Thread.java:1583)
```

Solution:

1. Delete the stale entry from the subscription table. For example, remove the entry corresponding to the subscription ID `6c0b06a8-06e1-47f3-91b3-6e3828284cca` with the POD IP `10.233.82.205:12595` in the presented example.
2. Restart the configuration service after deleting the entry from the subscription table in the configuration database.

2.2.37 Export Notifications Not Received on Export Service

Scenario:

During an upgrade or when configuration and export services are started simultaneously there maybe a case when the export service does not receive the export configuration notification from the configuration service

Problem:

Export service does not process anything as the export notification is not received

Solution:

Restart the export service POD

```
$ kubectl delete po <ocnaddexport-xxxx-xxxx-xxx> -n ocnadd-deploy
```

2.2.38 Correlation Configuration Does Not Get Deployed Post Rollback

Scenario:

Correlation configuration does not get deployed after the rollback.

Problem:

1. The UI is not accessible after the Configuration Service is restarted following a rollback from the current release to target release.
2. The Correlation Configuration does not get deployed after the Configuration Service is restarted following a rollback from the current release to target release.

Error Code or Error Message:

Figure 2-1 Error Code or Error Message

```
Below logs are observed in config service when UI is accessed :


OCL 2025-01-31 10:04:22.307 [virtual-executor89] ERROR i.m.h.s.RouteExecutor - Unexpected error occurred: failed to lazily initialize a collection of
role: com.oracle.cgbu.cne.ocnadd.configuration.correlation.entity.CorrelationConfiguration.supportedXdrContents: could not initialize proxy - no
Session
org.hibernate.LazyInitializationException: failed to lazily initialize a collection of role:
com.oracle.cgbu.cne.ocnadd.configuration.correlation.entity.CorrelationConfiguration.supportedXdrContents: could not initialize proxy - no Session
        at org.hibernate.collection.spi.AbstractPersistentCollection.throwLazyInitializationException(AbstractPersistentCollection.java:634)
        at org.hibernate.collection.spi.AbstractPersistentCollection.withTemporarySessionIfNeeded(AbstractPersistentCollection.java:217)
        at org.hibernate.collection.spi.AbstractPersistentCollection.initialize(AbstractPersistentCollection.java:613)
        at org.hibernate.collection.spi.AbstractPersistentCollection.read(AbstractPersistentCollection.java:136)
        at org.hibernate.collection.spi.PersistentBag.toString(PersistentBag.java:587)
        at java.base/java.lang.StringConcatHelper.stringOf(StringConcatHelper.java:467)
        at
```

Solution:

This issue can be addressed by applying the following workaround:

Workaround 1: Edit Deployment for Configuration Service

1. Edit the Configuration Service deployment and add the following parameter to the environment variables:

```
- name:
logger.levels.com.oracle.cgbu.cne.ocnadd.configuration.correlation.service.
configuration
  value: DEBUG
```

Workaround 2: Update the Custom Values File of the Management Group

1. Add the following parameter under the `ocnaddconfiguration` service section in the custom values file of the management group:

```
logger.levels.com.oracle.cgbu.cne.ocnadd.configuration.correlation.service.
configuration: DEBUG
```

For example:

```
ocnaddconfiguration:
  ocnaddconfiguration:
    name: ocnaddconfiguration
    env:

logger.levels.com.oracle.cgbu.cne.ocnadd.configuration.correlation.service.
configuration: DEBUG
```

2. Perform the Helm upgrade for the management group:

```
helm upgrade <management-release-name> -f ocnadd-custom-values-<mgmt-
group>.yaml --namespace <source-release-namespace> <helm_chart>
```

For example:

```
helm upgrade mgmt -f custom-templates/ocnadd-custom-values-mgmt.yaml -n
mgmt-ns ocnadd
```

2.2.39 Cleanup of Redundant xDR Tables

Scenario

xDR tables may remain in the Storage Adapter schema even after xDR storage has been disabled or removed. The following scenarios can lead to this condition:

- xDR storage is disabled in the correlation configuration.
- The correlation configuration has been deleted.
- The Storage Adapter was unavailable in a previous release, and a rollback was performed from the current release with xDR storage enabled.

Solution

If xDRs are no longer required, it is recommended to delete the xDR tables directly using the MySQL client.

2.3 CNLB Troubleshooting Scenarios

This section should be referred to for troubleshooting issues encountered after OCNADD deployment and configuration in a CNLB-based CNE 24.3.0 environment. The issues mentioned below are general problems that can be investigated to identify the root cause and find appropriate solutions.

Secondary Site Unable to Reach Primary Site RA on Load Balancer IP (CNLB IP):

Troubleshooting Steps:

1. Verify indentation and configuration of CNLB annotations first:

   ```
   $ kubectl edit -n <namespace> deployment.apps/ocnaddredundancyagent
   ```

   Indentation and configuration should be like below example:

   ```
   template:
     metadata:
       annotations:
         k8s.v1.cni.cncf.io/networks: default/nf-oam-int1@nf-oam-int1
         oracle.com.cnc/cnlb: '[{"backendPortName": "rrbackport",
   "cnlbIp":"10.123.155.33","cnlbPort":"13000"}]'
   ```

2. If the annotation indentation is correct, proceed to check the attached Network Attachment Definition (NAD) information:

   ```
   $ kubectl get net-attach-def <NAD Name Configured in RA> -n default -o yaml
   ```

   For example:

   ```
   spec:
     config: '{"cniVersion": "0.4.0", "name": "nf-oam-int1", "plugins": [{"type": "macvlan",
       "mode": "bridge", "master": "bond0.110", "ipam": {"type": "whereabouts", "range":
       "172.16.110.0/24", "range_start": "172.16.110.129", "range_end": "172.16.110.190",
       "gateway": "172.16.110.1"}}, {"type": "sbr"}]}'
   ```

3. Retrieve the service set information by running the following command:

   ```
   $ kubectl exec -ti -n <occne-infra namespace> $(kubectl -n <occne-infra
   namespace> get po --no-headers -l app=cnlb-manager -o custom-
   columns=:.metadata.name) -- curl http://localhost:5001/service-info
   ```

   Identify the `serviceIpSet` that contains the `frontEndIP` and `frontEndPort` of the configured CNLB annotation in the RA deployment.

   For example: The RA is assigned to service set to `0`. Ensure that `rrbackendport` and `10.121.44.157` are not repeated in more than one entry in any of service set.

   ```
   {
     "backendIpList": ["172.16.113.131"],
   ```

```
  "backendPort": 13000,
  "backendPortName": "rrbackendport",
  "frontEndIp": "10.123.155.33",
  "frontEndPort": 13000,
  "gatewayIp": "172.16.113.1",
  "networkName": "sig3"
}
```

a. If multiple entries are found, run the following command to check which other services are using the same `frontEndIp` (`cnlbIP`). Resolve the issue by following the *Oracle Communication Network Analytics Data Director User Guide* to re-enable RA CNLB and assign a new IP.

```
$ kubectl get deploy,sts,ds -A -o json | jq -r '.items[] |
select(.spec.template.metadata.annotations."k8s.v1.cni.cncf.io/
networks" != null) | {kind, namespace: .metadata.namespace,
name: .metadata.name,
podAnnotations: .spec.template.metadata.annotations."k8s.v1.cni.cncf.io/
networks", cnlbIPS: .spec.template.metadata.annotations."oracle.com.cnc/
cnlb"}'
```

b. Identify the CNLB App Assigned to Service Set 0

i. Run the following command to identify which active CNLB app is assigned to service set 0

```
kubectl exec -ti -n occne-infra $(kubectl -n <occne-infra
namespace> get po --no-headers -l app=cnlb-manager -o custom-
columns=:.metadata.name) -- curl http://localhost:5001/node-pod-role
```

Sample output:

ii. Determine which CNLB app has the pod IP `10.233.71.213`:

```
$ kubectl get po -n occne-infra -o wide | grep 10.233.71.213
```

Sample output:

iii. Check CNLB POD Status and Logs and Investigate identified CNLB pod:

```
kubectl logs -n <occne-infra namespace> cnlb-app-58cb4fd4bc-xpbwm
```

Analyze the logs to determine if the issue is related to the CNLB pod.

Ingress/Egress NAD information missing in Dropdown for Ingress Feed and Adapter Feed in GUI

1. First, investigate the CNLB job logs for any errors. If error logs are found, verify that there is a sufficient number of external IPs or that the network is correctly defined in the `cnlb.ini` file.

   ```
   $ kubectl logs -n <dd-mgmt-namespace> <cnlb-job-pod-name>
   ```

   For example:

   ```
   $ kubectl logs -n ocnadd-mgmt ocnaddcnlbannotationgen-h5fsb
   ```

2. Re-run the CNLB job to update the ConfigMap with the latest available NAD information. Refer to the *Oracle Communication Network Analytics Data Director User Guide* for instructions in the chapter "Create CNLB Annotation Generation Job."
3. Verify the content of the ingress/egress ConfigMap to ensure it has been created with the available ingress and egress NAD information.

   ```
   $ kubectl get cm -n <dd-mgmt-namespace> ocnadd-configmap-cnlb-egress -o
   yaml
   $ kubectl get cm -n <dd-mgmt-namespace> ocnadd-configmap-cnlb-ingress -o
   yaml
   ```

   For example:

   Ensure that at least one NAD entry is present in the ConfigMaps "ocnadd-configmap-cnlb-egress" and "ocnadd-configmap-cnlb-ingress" to enable traffic segregation.

2.4 Extended Storage with Druid Troubleshooting Scenarios

This section should be referred to for troubleshooting issues encountered after OCNADD integration with the DRUID database cluster. The issues mentioned below are generic and can be investigated using the Druid unified console or APIs.

Getting the list of active Supervisor Spec from Druid

A Supervisor Spec is created for every correlation configuration that supports extended storage when the extended storage type is set to Druid database. The API below provides a way to fetch the active Supervisor Specs running for Kafka ingestion.

URL: *http://<druid_router_ip>:<druid_router_port>/druid/indexer/v1/supervisor*

Endpoint: `/druid/indexer/v1/supervisor`

Method: `GET`

Request Body: None

Response: `200 OK` – Returns a `List<String>`

Description: Returns an array of strings representing the names of active supervisors. If there are no active supervisors, it returns an empty array.

Example Response:

```
[
  "corr1_tdr",
  "corr2_sudr"
]
```

2. List Active Supervisors (Full Supervisor Objects)
Endpoint: `/druid/indexer/v1/supervisor?full=null`

Method: `GET`

Request Body: None

Response: `200 OK` – Returns a `List<Supervisor>`

Description: Returns an array of Supervisor objects. If there are no active supervisors, it returns an empty array.

Example:

```
[
  {
    "id": "joe_corr_feed_2XDR_dvm1",
    "state": "RUNNING",
    "detailedState": "RUNNING",
    "healthy": true,
    "spec": {
      "type": "kafka",
      "spec": {
        "dataSchema": {
          "dataSource": "joe_corr_feed_2XDR_dvm1",
          "timestampSpec": {
            "column": "beginTime",
            "format": "iso",
            "missingValue": null
          },
          "dimensionsSpec": {
            "dimensions": [
```

```
              { "type": "string", "name": "version", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "beginTime", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "endTime", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "configurationName",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "xdrStatus", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "long", "name": "totalPduCount",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": false },
              { "type": "long", "name": "totalLength", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": false },
              { "type": "string", "name": "transactionTime",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "path", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "methodType", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "egressAuthority",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "feedSourceNfFqdn",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "feedSourceNfId",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "producerFqdn",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "contentType",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "supi", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "ingressAuthority",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "consumerId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "consumerFqdn",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "ueId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "consumerNfType",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "xdrId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "gpsi", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "pei", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "statusCode", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "userAgent", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "transactionId",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "feedSourceNfType",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
```

```
              { "type": "string", "name": "producerId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "registrationTime",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "smfInstanceId",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "smfSetId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "snssai", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "dnn", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "pcfInstanceId",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "consumerVia",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "producerVia",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "consumerPlmn",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "string", "name": "producerPlmn",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
              { "type": "long", "name": "pduSessionId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": false },
              { "type": "json", "name": "messages", "formatVersion": 5,
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true }
            ],
            "dimensionExclusions": ["__time"],
            "includeAllDimensions": false,
            "useSchemaDiscovery": false
          },
          "metricsSpec": [],
          "granularitySpec": {
            "type": "uniform",
            "segmentGranularity": "HOUR",
            "queryGranularity": { "type": "none" },
            "rollup": false,
            "intervals": []
          },
          "transformSpec": {
            "filter": null,
            "transforms": [
              { "type": "expression", "name": "beginTime", "expression":
"timestamp_format(__time)" },
              { "type": "expression", "name": "xdrId", "expression":
"timestamp(endTime)+1" }
            ]
          }
        },
        "ioConfig": {
          "topic": "ADC-CORR-FEED-2-CORRELATED",
          "topicPattern": null,
          "inputFormat": { "type": "json", "keepNullColumns": false },
          "replicas": 1,
          "taskCount": 1,
          "taskDuration": "PT900S",
```

```
          "consumerProperties": {
            "auto.offset.reset": "earliest",
            "bootstrap.servers": "kafka-broker-0.kafka-broker-headless.iq-s-
mgmt.svc.dv-m1.lab.in.oracle.com:9094,kafka-broker-1.kafka-broker-headless.iq-
s-mgmt.svc.dv-m1.lab.in.oracle.com:9094,kafka-broker-2.kafka-broker-
headless.iq-s-mgmt.svc.dv-m1.lab.in.oracle.com:9094,kafka-broker-3.kafka-
broker-headless.iq-s-mgmt.svc.dv-m1.lab.in.oracle.com:9094",
            "ssl.protocol": "TLSv1.3",
            "security.protocol": "SASL_SSL",
            "ssl.keystore.location": "/var/securityfiles/keystore/
serverKeyStore.p12",
            "ssl.truststore.location": "/var/securityfiles/keystore/
trustStore.p12",
            "ssl.truststore.type": "PKCS12",
            "ssl.keystore.type": "PKCS12",
            "druid.dynamic.config.provider": {
              "type": "environment",
              "variables": {
                "ssl.keystore.password": "CLIENT_KS_PASSWORD",
                "ssl.truststore.password": "CLIENT_TS_PASSWORD"
              }
            },
            "sasl.mechanism": "SCRAM-SHA-256",
            "sasl.jaas.config":
"org.apache.kafka.common.security.scram.ScramLoginModule required
username=\"extuser1\" password=\"extuser1\";"
          },
          "autoScalerConfig": null,
          "pollTimeout": 100,
          "startDelay": "PT5S",
          "period": "PT30S",
          "useEarliestOffset": true,
          "completionTimeout": "PT1800S",
          "lateMessageRejectionPeriod": null,
          "earlyMessageRejectionPeriod": null,
          "lateMessageRejectionStartDateTime": null,
          "configOverrides": null,
          "idleConfig": null,
          "stopTaskCount": null,
          "emitTimeLagMetrics": false,
          "stream": "JOE-CORR-FEED-2-CORRELATED",
          "useEarliestSequenceNumber": true
        },
        "tuningConfig": {
          "type": "kafka",
          "appendableIndexSpec": { "type": "onheap",
"preserveExistingMetrics": false },
          "maxRowsInMemory": 150000,
          "maxBytesInMemory": 0,
          "skipBytesInMemoryOverheadCheck": false,
          "maxRowsPerSegment": 50000,
          "maxTotalRows": null,
          "intermediatePersistPeriod": "PT1M",
          "maxPendingPersists": 0,
          "indexSpec": {
            "bitmap": { "type": "roaring" },
```

```
                "dimensionCompression": "lz4",
                "stringDictionaryEncoding": { "type": "utf8" },
                "metricCompression": "lz4",
                "longEncoding": "longs"
              },
              "indexSpecForIntermediatePersists": {
                "bitmap": { "type": "roaring" },
                "dimensionCompression": "lz4",
                "stringDictionaryEncoding": { "type": "utf8" },
                "metricCompression": "lz4",
                "longEncoding": "longs"
              },
              "reportParseExceptions": false,
              "handoffConditionTimeout": 900000,
              "resetOffsetAutomatically": true,
              "segmentWriteOutMediumFactory": null,
              "workerThreads": null,
              "chatRetries": 8,
              "httpTimeout": "PT10S",
              "shutdownTimeout": "PT80S",
              "offsetFetchPeriod": "PT30S",
              "intermediateHandoffPeriod": "P2147483647D",
              "logParseExceptions": false,
              "maxParseExceptions": 2147483647,
              "maxSavedParseExceptions": 0,
              "numPersistThreads": 1,
              "maxColumnsToMerge": -1,
              "skipSequenceNumberAvailabilityCheck": false,
              "repartitionTransitionDuration": "PT120S"
            }
          },
          "dataSchema": {
            "dataSource": "joe_corr_feed_2XDR_dvm1",
            "timestampSpec": {
              "column": "beginTime",
              "format": "iso",
              "missingValue": null
            },
            "dimensionsSpec": {
              "dimensions": [
                { "type": "string", "name": "version", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
                { "type": "string", "name": "beginTime", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
                { "type": "string", "name": "endTime", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
                { "type": "string", "name": "configurationName",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
                { "type": "string", "name": "xdrStatus", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
                { "type": "long", "name": "totalPduCount", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": false },
                { "type": "long", "name": "totalLength", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": false },
                { "type": "string", "name": "transactionTime",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
```

```
            { "type": "string", "name": "path", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "methodType", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "egressAuthority",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "feedSourceNfFqdn",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "feedSourceNfId",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "producerFqdn", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "contentType", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "supi", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "ingressAuthority",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "consumerId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "consumerFqdn", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "ueId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "consumerNfType",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "xdrId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "gpsi", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "pei", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "statusCode", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "userAgent", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "transactionId",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "feedSourceNfType",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "producerId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "registrationTime",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "smfInstanceId",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "smfSetId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "snssai", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "dnn", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "pcfInstanceId",
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "consumerVia", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
```

```
            { "type": "string", "name": "producerVia", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "consumerPlmn", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "string", "name": "producerPlmn", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": true },
            { "type": "long", "name": "pduSessionId", "multiValueHandling":
"SORTED_ARRAY", "createBitmapIndex": false },
            { "type": "json", "name": "messages", "formatVersion": 5,
"multiValueHandling": "SORTED_ARRAY", "createBitmapIndex": true }
          ],
          "dimensionExclusions": ["__time"],
          "includeAllDimensions": false,
          "useSchemaDiscovery": false
        },
        "metricsSpec": [],
        "granularitySpec": {
          "type": "uniform",
          "segmentGranularity": "HOUR",
          "queryGranularity": { "type": "none" },
          "rollup": false,
          "intervals": []
        },
        "transformSpec": {
          "filter": null,
          "transforms": [
            { "type": "expression", "name": "beginTime", "expression":
"timestamp_format(__time)" },
            { "type": "expression", "name": "xdrId", "expression":
"timestamp(endTime)+1" }
          ]
        }
      },
      "tuningConfig": {
        "type": "kafka",
        "appendableIndexSpec": { "type": "onheap", "preserveExistingMetrics":
false },
        "maxRowsInMemory": 150000,
        "maxBytesInMemory": 0,
        "skipBytesInMemoryOverheadCheck": false,
        "maxRowsPerSegment": 50000,
        "maxTotalRows": null,
        "intermediatePersistPeriod": "PT1M",
        "maxPendingPersists": 0,
        "indexSpec": {
          "bitmap": { "type": "roaring" },
          "dimensionCompression": "lz4",
          "stringDictionaryEncoding": { "type": "utf8" },
          "metricCompression": "lz4",
          "longEncoding": "longs"
        },
        "indexSpecForIntermediatePersists": {
          "bitmap": { "type": "roaring" },
          "dimensionCompression": "lz4",
          "stringDictionaryEncoding": { "type": "utf8" },
          "metricCompression": "lz4",
```

```
            "longEncoding": "longs"
          },
          "reportParseExceptions": false,
          "handoffConditionTimeout": 900000,
          "resetOffsetAutomatically": true,
          "segmentWriteOutMediumFactory": null,
          "workerThreads": null,
          "chatRetries": 8,
          "httpTimeout": "PT10S",
          "shutdownTimeout": "PT80S",
          "offsetFetchPeriod": "PT30S",
          "intermediateHandoffPeriod": "P2147483647D",
          "logParseExceptions": false,
          "maxParseExceptions": 2147483647,
          "maxSavedParseExceptions": 0,
          "numPersistThreads": 1,
          "maxColumnsToMerge": -1,
          "skipSequenceNumberAvailabilityCheck": false,
          "repartitionTransitionDuration": "PT120S"
        },
        "ioConfig": {
          "topic": "JOE-CORR-FEED-2-CORRELATED",
          "topicPattern": null,
          "inputFormat": { "type": "json", "keepNullColumns": false },
          "replicas": 1,
          "taskCount": 1,
          "taskDuration": "PT900S",
          "consumerProperties": {
            "auto.offset.reset": "earliest",
            "bootstrap.servers": "kafka-broker-0.kafka-broker-headless.iq-s-
mgmt.svc.dv-m1.lab.in.oracle.com:9094,kafka-broker-1.kafka-broker-headless.iq-
s-mgmt.svc.dv-m1.lab.in.oracle.com:9094,kafka-broker-2.kafka-broker-
headless.iq-s-mgmt.svc.dv-m1.lab.in.oracle.com:9094,kafka-broker-3.kafka-
broker-headless.iq-s-mgmt.svc.dv-m1.lab.in.oracle.com:9094",
            "ssl.protocol": "TLSv1.3",
            "security.protocol": "SASL_SSL",
            "ssl.keystore.location": "/var/securityfiles/keystore/
serverKeyStore.p12",
            "ssl.truststore.location": "/var/securityfiles/keystore/
trustStore.p12",
            "ssl.truststore.type": "PKCS12",
            "ssl.keystore.type": "PKCS12",
            "druid.dynamic.config.provider": {
              "type": "environment",
              "variables": {
                "ssl.keystore.password": "CLIENT_KS_PASSWORD",
                "ssl.truststore.password": "CLIENT_TS_PASSWORD"
              }
            },
            "sasl.mechanism": "SCRAM-SHA-256",
            "sasl.jaas.config":
"org.apache.kafka.common.security.scram.ScramLoginModule required
username=\"extuser1\" password=\"extuser1\";"
          },
          "autoScalerConfig": null,
          "pollTimeout": 100,
```

```
        "startDelay": "PT5S",
        "period": "PT30S",
        "useEarliestOffset": true,
        "completionTimeout": "PT1800S",
        "lateMessageRejectionPeriod": null,
        "earlyMessageRejectionPeriod": null,
        "lateMessageRejectionStartDateTime": null,
        "configOverrides": null,
        "idleConfig": null,
        "stopTaskCount": null,
        "emitTimeLagMetrics": false,
        "stream": "JOE-CORR-FEED-2-CORRELATED",
        "useEarliestSequenceNumber": true
      },
      "context": null,
      "suspended": false
    },
    "suspended": false
  }
]
```


3. Get Supervisor Specification
Endpoint: `/druid/indexer/v1/supervisor/{supervisorId}`

Method: `GET`

Request Body: None

Response:

- `200 OK` – Returns `IngestionSpec`
- `404 NotFound` – None

Description: Retrieves the specification for a single supervisor. The returned specification includes:

- `dataSchema`
- `ioConfig`
- `tuningConfig`

Example:


```
{
  "type": "kafka",
  "spec": {
    "dataSchema": {
      "dataSource": "DD_FEED2XDR",
      "timestampSpec": {
        "column": "beginTime",
        "format": "iso",
        "missingValue": null
      },
      "dimensionsSpec": {
        "dimensions": [
          {
            "type": "string",
```

```
      "name": "version",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "beginTime",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "endTime",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "configurationName",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "xdrStatus",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "long",
      "name": "totalPduCount",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": false
    },
    {
      "type": "long",
      "name": "totalLength",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": false
    },
    {
      "type": "string",
      "name": "transactionTime",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "path",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "methodType",
      "multiValueHandling": "SORTED_ARRAY",
```

```
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "egressAuthority",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "feedSourceNfFqdn",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "feedSourceNfId",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "producerFqdn",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "contentType",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "supi",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "ingressAuthority",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "consumerId",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "consumerFqdn",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
```

```
{
  "type": "string",
  "name": "ueId",
  "multiValueHandling": "SORTED_ARRAY",
  "createBitmapIndex": true
},
{
  "type": "string",
  "name": "consumerNfType",
  "multiValueHandling": "SORTED_ARRAY",
  "createBitmapIndex": true
},
{
  "type": "string",
  "name": "xdrId",
  "multiValueHandling": "SORTED_ARRAY",
  "createBitmapIndex": true
},
{
  "type": "string",
  "name": "gpsi",
  "multiValueHandling": "SORTED_ARRAY",
  "createBitmapIndex": true
},
{
  "type": "string",
  "name": "pei",
  "multiValueHandling": "SORTED_ARRAY",
  "createBitmapIndex": true
},
{
  "type": "string",
  "name": "statusCode",
  "multiValueHandling": "SORTED_ARRAY",
  "createBitmapIndex": true
},
{
  "type": "string",
  "name": "userAgent",
  "multiValueHandling": "SORTED_ARRAY",
  "createBitmapIndex": true
},
{
  "type": "string",
  "name": "transactionId",
  "multiValueHandling": "SORTED_ARRAY",
  "createBitmapIndex": true
},
{
  "type": "string",
  "name": "feedSourceNfType",
  "multiValueHandling": "SORTED_ARRAY",
  "createBitmapIndex": true
},
{
  "type": "string",
```

```
      "name": "producerId",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "registrationTime",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "smfInstanceId",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "smfSetId",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "snssai",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "dnn",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "pcfInstanceId",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "consumerVia",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "producerVia",
      "multiValueHandling": "SORTED_ARRAY",
      "createBitmapIndex": true
    },
    {
      "type": "string",
      "name": "consumerPlmn",
      "multiValueHandling": "SORTED_ARRAY",
```

```
            "createBitmapIndex": true
          },
          {
            "type": "string",
            "name": "producerPlmn",
            "multiValueHandling": "SORTED_ARRAY",
            "createBitmapIndex": true
          },
          {
            "type": "long",
            "name": "pduSessionId",
            "multiValueHandling": "SORTED_ARRAY",
            "createBitmapIndex": false
          },
          {
            "type": "json",
            "name": "messages",
            "formatVersion": 5,
            "multiValueHandling": "SORTED_ARRAY",
            "createBitmapIndex": true
          }
        ],
        "dimensionExclusions": [
          "__time"
        ],
        "includeAllDimensions": false,
        "useSchemaDiscovery": false
      },
      "metricsSpec": [],
      "granularitySpec": {
        "type": "uniform",
        "segmentGranularity": "HOUR",
        "queryGranularity": {
          "type": "none"
        },
        "rollup": false,
        "intervals": []
      },
      "transformSpec": {
        "filter": null,
        "transforms": [
          {
            "type": "expression",
            "name": "beginTime",
            "expression": "timestamp_format(__time)"
          },
          {
            "type": "expression",
            "name": "xdrId",
            "expression": "timestamp(endTime)+1"
          }
        ]
      }
    },
    "ioConfig": {
      "topic": "FEED2-CORRELATED",
```

```
    "topicPattern": null,
    "inputFormat": {
      "type": "json",
      "keepNullColumns": false,
      "assumeNewlineDelimited": false,
      "useJsonNodeReader": false
    },
    "replicas": 1,
    "taskCount": 1,
    "taskDuration": "PT3600S",
    "consumerProperties": {
      "bootstrap.servers": "192.168.56.108:9092"
    },
    "autoScalerConfig": null,
    "pollTimeout": 100,
    "startDelay": "PT5S",
    "period": "PT30S",
    "useEarliestOffset": false,
    "completionTimeout": "PT1800S",
    "lateMessageRejectionPeriod": null,
    "earlyMessageRejectionPeriod": null,
    "lateMessageRejectionStartDateTime": null,
    "configOverrides": null,
    "idleConfig": null,
    "stopTaskCount": null,
    "stream": "FEED2-CORRELATED",
    "useEarliestSequenceNumber": false
  },
  "tuningConfig": {
    "type": "kafka",
    "appendableIndexSpec": {
      "type": "onheap",
      "preserveExistingMetrics": false
    },
    "maxRowsInMemory": 150000,
    "maxBytesInMemory": 0,
    "skipBytesInMemoryOverheadCheck": false,
    "maxRowsPerSegment": 50000,
    "maxTotalRows": null,
    "intermediatePersistPeriod": "PT10M",
    "maxPendingPersists": 0,
    "indexSpec": {
      "bitmap": {
        "type": "roaring"
      },
      "dimensionCompression": "lz4",
      "stringDictionaryEncoding": {
        "type": "utf8"
      },
      "metricCompression": "lz4",
      "longEncoding": "longs"
    },
    "indexSpecForIntermediatePersists": {
      "bitmap": {
        "type": "roaring"
      },
```

```
        "dimensionCompression": "lz4",
        "stringDictionaryEncoding": {
          "type": "utf8"
        },
        "metricCompression": "lz4",
        "longEncoding": "longs"
      },
      "reportParseExceptions": false,
      "handoffConditionTimeout": 900000,
      "resetOffsetAutomatically": false,
      "segmentWriteOutMediumFactory": null,
      "workerThreads": null,
      "chatRetries": 8,
      "httpTimeout": "PT10S",
      "shutdownTimeout": "PT80S",
      "offsetFetchPeriod": "PT30S",
      "intermediateHandoffPeriod": "P2147483647D",
      "logParseExceptions": false,
      "maxParseExceptions": 2147483647,
      "maxSavedParseExceptions": 0,
      "numPersistThreads": 1,
      "skipSequenceNumberAvailabilityCheck": false,
      "repartitionTransitionDuration": "PT120S"
    }
  },
  "context": null,
  "suspended": false
}
```


Getting the Health, Status and Stats of the supervisor

URL: *http://<druid_router_ip>:<druid_router_port>/druid/indexer/v1/supervisor*

1. Get Supervisor Health
Endpoint: `/druid/indexer/v1/supervisor/{supervisorId}/health`

Method: `GET`

Request Body: None

Response:

- `200 OK` – Returns a JSON Object
- `404 NotFound` – `"Invalid SupervisorID"`
- `503 Service Unavailable` – `"Supervisor is unhealthy"`

Description: Retrieves the current health report for a single supervisor. The health of a supervisor is determined by the supervisor's state (as returned by the `/status` endpoint).

Example Response:


```
{
  "healthy": false
}
```


2. Get Supervisor Stats
Endpoint: `/druid/indexer/v1/supervisor/{supervisorId}/stats`

Method: `GET`

Request Body: None

Response:

- `200 OK` – Returns a JSON Object
- `404 NotFound` – `"Cannot find any supervisor with id: [\"XXX\"]"`

Description: Returns a snapshot of the current ingestion row counters for each task being managed by the supervisor, along with moving averages for the row counters.

Example:


```
{
  "0": {
    "index_kafka_adc_corr_feed_2XDR_dvm1_a971c56a6a3aa33_kioblpna": {
      "movingAverages": {
        "buildSegments": {
          "5m": {
            "processed": 11.075906329418917,
            "processedBytes": 13420.280089272523,
            "unparseable": 0.0,
            "thrownAway": 0.0,
            "processedWithError": 0.0
          },
          "15m": {
            "processed": 16.880517867690518,
            "processedBytes": 20190.08795040925,
            "unparseable": 0.0,
            "thrownAway": 0.0,
            "processedWithError": 0.0
          },
          "1m": {
            "processed": 10.000016421654086,
            "processedBytes": 12169.78312232924,
            "unparseable": 0.0,
            "thrownAway": 0.0,
            "processedWithError": 0.0
          }
        }
      },
      "totals": {
        "buildSegments": {
          "processed": 8497,
          "processedBytes": 10334893,
          "processedWithError": 0,
          "thrownAway": 0,
          "unparseable": 0
        }
      }
    }
  }
}
```


3. Get Supervisor Status
Endpoint: `/druid/indexer/v1/supervisor/{supervisorId}/status`

Method: `GET`

Request Body: None

Response:

- `200 OK` – Returns a JSON Object
- `404 NotFound` – `"Cannot find any supervisor with id: [\"XXX\"]"`

Description: Retrieves the current status report for a single supervisor. The report contains:

- The state of the supervisor tasks
- An array of recently thrown exceptions

Example:


```
{
  "id": "adc_corr_feed_2XDR_dvm1",
  "generationTime": "2025-08-11T04:44:09.346Z",
  "payload": {
    "dataSource": "adc_corr_feed_2XDR_dvm1",
    "stream": "ADC-CORR-FEED-2-CORRELATED",
    "partitions": 6,
    "replicas": 1,
    "durationSeconds": 900,
    "activeTasks": [
      {
        "id": "index_kafka_adc_corr_feed_2XDR_dvm1_1951ed0543fa752_hggimlaf",
        "startingOffsets": {
          "1": 390784,
          "0": 389009,
          "3": 389412,
          "2": 389161,
          "5": 389612,
          "4": 390184
        },
        "startTime": "2025-08-11T04:41:11.149Z",
        "remainingSeconds": 721,
        "type": "ACTIVE",
        "currentOffsets": {
          "1": 391078,
          "0": 389256,
          "3": 389701,
          "2": 389366,
          "5": 389901,
          "4": 390473
        },
        "lag": {
          "1": 0,
          "0": 0,
          "3": 0,
          "2": 0,
          "5": 0,
          "4": 0
        }
      }
    ],
```

```
    "publishingTasks": [],
    "latestOffsets": {
      "1": 391078,
      "0": 389256,
      "3": 389701,
      "2": 389366,
      "5": 389901,
      "4": 390473
    },
    "minimumLag": {
      "1": 0,
      "0": 0,
      "3": 0,
      "2": 0,
      "5": 0,
      "4": 0
    },
    "aggregateLag": 0,
    "offsetsLastUpdated": "2025-08-11T04:43:43.314Z",
    "suspended": false,
    "healthy": true,
    "state": "RUNNING",
    "detailedState": "RUNNING",
    "recentErrors": []
  }
}
```


Indexing Jobs not processing due to missing offsets

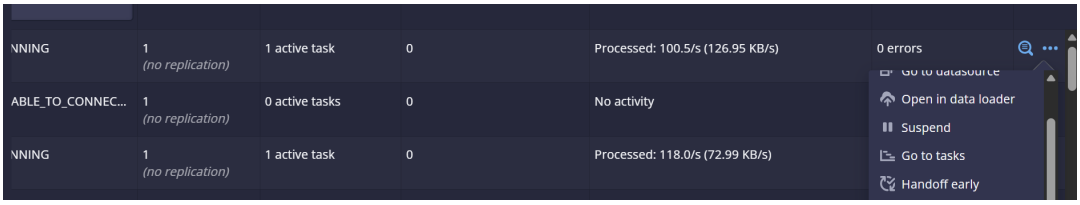
Problem: The Kafka ingestion job is not processing data due to missing offsets in Druid. A **"Druid Kafka offset error"**, specifically an `OffsetOutOfRangeException`, occurs when a Druid Kafka indexing task tries to consume data from a Kafka topic offset that no longer exists. This usually happens when messages in Kafka are deleted according to the retention policy before Druid can ingest them.

Solution:

1. **Login to the Druid web console usingthe following URL:**

   ```
   http://<druid_router_ip>:<druid_router_port>/unified-
   console.html#supervisors
   ```

2. **Access the affected supervisor:**
   Scroll to the right side of the supervisor configuration and click the `...` menu. You will see the **Suspend** option.

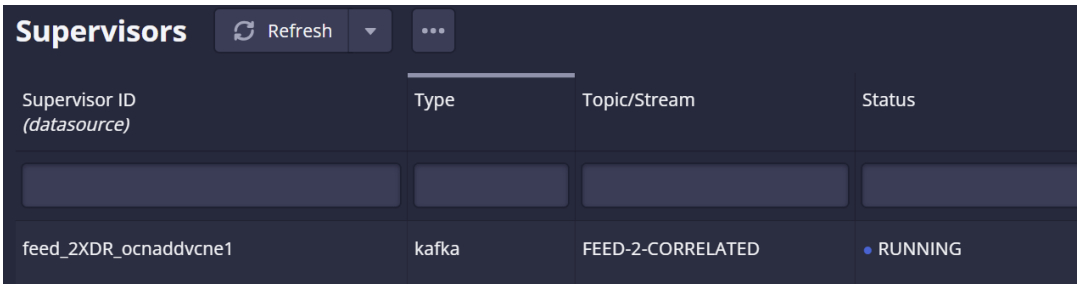


3. **Suspend the supervisor:**
   Click **Suspend** to pause the supervisor.

4. **Hard reset the offsets:**
   Click the . . . menu again and select **Hard Reset**.

5. **Resume the supervisor:**
   Click **Resume** to restart the supervisor configuration.

6. **Verify ingestion:**
   The supervisor should enter the **Running** state, and data ingestion should start successfully.



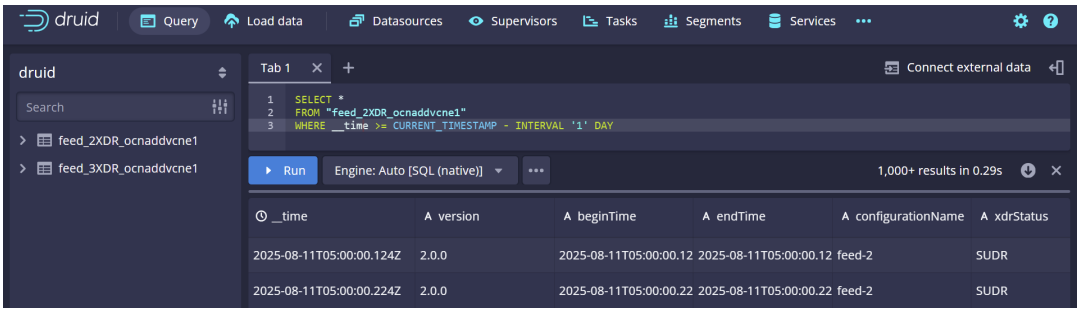
Running Sql native query on Druid console

1. **Login to the Druid web console using the following URL:**

   ```
   http://<druid_router_ip>:<druid_router_port>/unified-console.html
   ```

2. **Select the Query tab:**
   From the top menu, click on **Query**.

3. **Run the SQL query:**
   Run the SQL query for the corresponding datasource directly on the web console, as shown below.



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
List of Alerts

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

5.1 System Level Alerts

This section lists the system level alerts for OCNADD.

Table 5-1 OCNADD_POD_CPU_USAGE_ALERT

Field	Details
Triggering Condition	POD CPU usage is above the set threshold (default 85%)
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 CNE	**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 }} % ' **PromQL 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=~".*kraft.*"}[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 (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*ui.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*1) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*export.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*4) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*storageadapter.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*3) or (sum(rate(container_cpu_usage_seconds_total{image!="" , pod=~".*ingressadapter.*"}[5m])) by (pod,namespace) > {{ .Values.global.cluster.cpu_threshold }}*3)

Table 5-1 (Cont.) OCNADD_POD_CPU_USAGE_ALERT

Field	Details
Alert Details OCI	**Summary:** Alarm "OCNADD_POD_CPU_USAGE_ALERT" is in a "**X**" state; because **n** metrics meet the trigger rule: "pod_cpu_usage_seconds_total[10m]{pod=~"*alarm*|*admin*|*health*|*config*|*kraft*"}.rate().groupby(namespace,pod).sum()*100>=85|| pod_cpu_usage_seconds_total[10m]{pod=~"*ui*|*aggregation*|*filter*"}.rate().groupby(namespace,pod).sum()*100>=2*85|| pod_cpu_usage_seconds_total[10m] {pod=~"*corr*"}.rate().groupby(namespace,pod).sum()*100>=3*85|| pod_cpu_usage_seconds_total[10m] {pod=~"*kafka*"}.rate().groupby(namespace,pod).sum()*100>=6*85|| pod_cpu_usage_seconds_total[10m] {pod=~"*export*"}.rate().groupby(namespace,pod).sum()*100>=4*85|| pod_cpu_usage_seconds_total[10m] {pod=~"*storageadapter*"}.rate().groupby(namespace,pod).sum()*100>=3*85|| pod_cpu_usage_seconds_total[10m] {pod=~"*ingressadapter*"}.rate().groupby(namespace,pod).sum()*100>=3*85", with a trigger delay of 1 minute where, X = FIRING/OK, n = Different services that violated the rule. **MQL Expression:** pod_cpu_usage_seconds_total[10m]{pod=~"*alarm*|*admin*|*health*|*config*|*kraft*"}.rate().groupby(namespace,pod).sum()*100>={{ CPU Threshold }}||pod_cpu_usage_seconds_total[10m]{pod=~"*ui*|*aggregation*|*filter*"}.rate().groupby(namespace,pod).sum()*100>=2*{{ CPU Threshold }}||pod_cpu_usage_seconds_total[10m] {pod=~"corr*"}.rate().groupby(namespace,pod).sum()*100>=3*{{ CPU Threshold }}||pod_cpu_usage_seconds_total[10m] {pod=~"*kafka*"}.rate().groupby(namespace,pod).sum()*100>=6*{{ CPU Threshold }} || pod_cpu_usage_seconds_total[10m] {pod=~"*export*"}.rate().groupby(namespace,pod).sum()*100>=4*{{ CPU Threshold }} || pod_cpu_usage_seconds_total[10m] {pod=~"*storageadapter*"}.rate().groupby(namespace,pod).sum()*100>=3*{{ CPU Threshold }} || pod_cpu_usage_seconds_total[10m] {pod=~"*ingressadapter*"}.rate().groupby(namespace,pod).sum()*100>=3*{{ CPU Threshold }} **Note**: CPU Threshold will be assigned will executing the terraform script
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 a 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.yaml` file. If guidance is required, contact My Oracle Support.

Table 5-2 OCNADD_POD_MEMORY_USAGE_ALERT

Field	Details
Triggering Condition	POD Memory usage is above set threshold (default 90%)
Severity	Major
Description	OCNADD Pod High Memory usage detected for a continuous period of 5min

Table 5-2 (Cont.) OCNADD_POD_MEMORY_USAGE_ALERT

Field	Details
Alert Details CNE	**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 }} % ' **PromQL 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=~".*kraft.*"}) 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) > 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) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*export.*"}) by (pod,namespace) >

Table 5-2 (Cont.) OCNADD_POD_MEMORY_USAGE_ALERT

Field	Details
	64*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*storageadapter.*"}) by (pod,namespace) > 64*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100) or (sum(container_memory_working_set_bytes{image!="" , pod=~".*ingressadapter.*"}) by (pod,namespace) > 8*1024*1024*1024*{{ .Values.global.cluster.memory_threshold }}/100)
Alert Details OCI	**Summary:** Alarm "OCNADD_POD_MEMORY_USAGE_ALERT" is in a "X" state; because n metrics meet the trigger rule: "container_memory_usage_bytes[5m]{pod=~"*adapter*|*kafka*|*kraft*|*ocnadd*|*corr*|*export*|*storageadapter*|*ingressadapter*"}.groupby(namespace,pod).sum()/container_spec_memory_limit_bytes[5m]{pod=~"*adapter*|*kafka*|*kraft*|*ocnadd*|*corr*|*export*|*storageadapter*|*ingressadapter*"}.groupby(namespace,pod).sum()*100>=90", with a trigger delay of 1 minute where, X = FIRING/OK, n = Different services that violated the rule. **MQL Expression:** container_memory_usage_bytes[10m]{pod=~"*adapter*|*kafka*|*kraft*|*ocnadd*|corr*|*export*|*storageadapter*|*ingressadapter*"}.groupby(namespace,pod).sum()/container_spec_memory_limit_bytes[10m]{pod=~"*adapter*|*kafka*|*kraft*|*ocnadd*|corr*|*export*|*storageadapter*|*ingressadapter*"}.groupby(namespace,pod).sum()*100>={{ Memory Threshold }}
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 a 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_custom_values.yaml` file. If guidance is required, contact My Oracle Support.

Table 5-3 OCNADD_POD_RESTARTED

Field	Details
Triggering Condition	A POD has restarted
Severity	Minor
Description	A POD has restarted in the last 2 min

Table 5-3 (Cont.) OCNADD_POD_RESTARTED

Field	Details
Alert Details CNE	**Summary:** 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: A Pod has restarted' **PromQL Expression**: expr: kube_pod_container_status_restarts_total{namespace="{{ .Values.global.cluster.nameSpace.name }}"} > 1
Alert Details OCI	**MQL Expression:** No MQL equivalent is available
OID	1.3.6.1.4.1.323.5.3.53.1.29.5006
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 a similar metric as exposed by the monitoring system.
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.

5.2 Application Level Alerts

This section lists the application level alerts for OCNADD.

Table 5-4 OCNADD_CONFIG_SVC_DOWN

Field	Details
Triggering Condition	The configuration service went down or not accessible
Severity	Critical
Description	OCNADD Configuration service not available for more than 2 min

Table 5-4 (Cont.) OCNADD_CONFIG_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddconfiguration service is down' **PromQL Expression:** expr: up{service="ocnaddconfiguration"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_CONFIG_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m] {podOwner="ocnaddconfiguration"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddconfiguration"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.20.2002
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.
Resolution	The alert is cleared automatically when the OCNADD Configuration 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..

Table 5-5 OCNADD_ALARM_SVC_DOWN

Field	Details
Triggering Condition	The alarm service went down or not accessible
Severity	Critical
Description	OCNADD Alarm service not available for more than 2 min

Table 5-5 (Cont.) OCNADD_ALARM_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddalarm service is down' **PromQL Expression:** expr: up{service="ocnaddalarm"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_ALARM_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m] {podOwner="ocnaddalarm"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddalarm"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.24.2002
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.
Resolution	The alert is cleared automatically when the OCNADD Alarm 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.

Table 5-6 OCNADD_HEALTH_MONITORING_SVC_DOWN

Field	Details
Triggering Condition	The health monitoring service went down or not accessible
Severity	Critical
Description	OCNADD Health monitoring service not available for more than 2 min

Table 5-6 (Cont.) OCNADD_HEALTH_MONITORING_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: summary: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddhealthmonitoring service is down' **PromQL Expression:** expr: up{service="ocnaddhealthmonitoring"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_HEALTH_MONITORING_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="ocnaddhealthmonitoring"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddhealthmonitoring"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.28.2002
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.
Resolution	The alert is cleared automatically when the OCNADD Health monitoring 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.

Table 5-7 OCNADD_SCP_AGGREGATION_SVC_DOWN

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

Table 5-7 (Cont.) OCNADD_SCP_AGGREGATION_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddscpaggregation service is down' **PromQL Expression:** expr: up{service="ocnaddscpaggregation"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_SCP_AGGREGATION_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="ocnaddscpaggregation"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddscpaggregation"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.22.2002
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.
Resolution	The alert is cleared automatically when the OCNADD SCP Aggregation 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.

Table 5-8 OCNADD_NRF_AGGREGATION_SVC_DOWN

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

Table 5-8 (Cont.) OCNADD_NRF_AGGREGATION_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddnrfaggregation service is down' **PromQL Expression:** expr: up{service="ocnaddnrfaggregation"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_NRF_AGGREGATION_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="ocnaddnrfaggregation"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddnrfaggregation"}.mean()!=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 a similar metric as exposed by the monitoring system.
Resolution	The alert is cleared automatically when the OCNADD NRF Aggregation 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.

Table 5-9 OCNADD_SEPP_AGGREGATION_SVC_DOWN

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

Table 5-9 (Cont.) OCNADD_SEPP_AGGREGATION_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddseppaggregation service is down' **PromQL Expression:** expr: up{service="ocnaddseppaggregation"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_SEPP_AGGREGATION_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="ocnaddseppaggregation"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddseppaggregation"}.mean()!=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 a similar metric as exposed by the monitoring system.
Resolution	The alert is cleared automatically when the OCNADD SEPP Aggregation 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.

Table 5-10 OCNADD_BSF_AGGREGATION_SVC_DOWN

Triggering Condition	The BSF Aggregation service went down or not accessible
Severity	Critical
Description	OCNADD BSF Aggregation service not available for more than 2 min

Table 5-10 (Cont.) OCNADD_BSF_AGGREGATION_SVC_DOWN

Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddbsfaggregation service is down' **PromQL Expression:** expr: up{service="ocnaddbsfaggregation"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_BSF_AGGREGATION_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="ocnaddbsfaggregation"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddbsfaggregation"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.40.2002
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.
Resolution	The alert is cleared automatically when the OCNADD BSF Aggregation 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.

Table 5-11 OCNADD_PCF_AGGREGATION_SVC_DOWN

Triggering Condition	The PCF Aggregation service went down or not accessible
Severity	Critical
Description	OCNADD PCF Aggregation service not available for more than 2 min

Table 5-11 (Cont.) OCNADD_PCF_AGGREGATION_SVC_DOWN

Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddpcfaggregation service is down' **PromQL Expression:** expr: up{service="ocnaddpcfaggregation"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_PCF_AGGREGATION_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="ocnaddpcfaggregation"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddpcfaggregation"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.41.2002
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.
Resolution	The alert is cleared automatically when the OCNADD PCF Aggregation 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.

Table 5-12 OCNADD_NON_ORACLE_AGGREGATION_SVC_DOWN

Triggering Condition	The Non Oracle Aggregation service went down or not accessible
Severity	Critical
Description	OCNADD Non Oracle Aggregation service not available for more than 2 min

Table 5-12 (Cont.) OCNADD_NON_ORACLE_AGGREGATION_SVC_DOWN

Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddnonoracleaggregation service is down' **PromQL Expression:** expr: up{service="ocnaddnonoracleaggregation"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_NON_ORACLE_AGGREGATION_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="ocnaddnonoracleaggregation"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddnonoracleaggregation"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.37.2002
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.
Resolution	The alert is cleared automatically when the OCNADD Non Oracle Aggregation service starts instance 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.

Table 5-13 OCNADD_ADMIN_SVC_DOWN

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

Table 5-13 (Cont.) OCNADD_ADMIN_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: ocnaddadminservice service is down' **PromQL Expression:** expr: up{service="ocnaddadminservice"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_ADMIN_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m] {podOwner="ocnaddadminservice"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddadminservice"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.30.2002
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.
Resolution	The alert is cleared automatically when the OCNADD Admin 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.

Table 5-14 OCNADD_CONSUMER_ADAPTER_SVC_DOWN

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

Table 5-14 (Cont.) OCNADD_CONSUMER_ADAPTER_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Consumer Adapter service is down' **PromQL Expression:** expr: up{service=~".*adapter.*", role="adapter"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_CONSUMER_ADAPTER_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="correlation"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner=~"adapter*"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.25.2002
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.
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.

Table 5-15 OCNADD_FILTER_SVC_DOWN

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

Table 5-15 (Cont.) OCNADD_FILTER_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Filter service is down' **PromQL Expression:** expr: up{service=~".*filter.*"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_FILTER_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m] {podOwner="ocnaddfilter"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ocnaddfilter"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.34.2002
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.
Resolution	The alert is cleared automatically when the OCNADD Filter 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.

Table 5-16 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

Table 5-16 (Cont.) OCNADD_CORRELATION_SVC_DOWN

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Correlation service is down' **PromQL Expression:** expr: up{service=~".*correlation.*"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_CORRELATION_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m] {podOwner="correlation"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="correlation"}.mean()!=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.
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.

Table 5-17 OCNADD_EXPORT_SVC_DOWN

Triggering Condition	The OCNADD Export service went down or not accessible
Severity	Critical
Description	OCNADD Export service not available for more than 2 min

Table 5-17 (Cont.) OCNADD_EXPORT_SVC_DOWN

Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Export service is down' **PromQL Expression:** expr: up{service=~".*export.*"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_EXPORT_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m] {podOwner="export"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="export"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.39.2002
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.
Resolution	The alert is cleared automatically when the OCNADD export 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.

Table 5-18 OCNADD_STORAGE_ADAPTER_SVC_DOWN

Triggering Condition	The OCNADD Storage adapter service went down or not accessible
Severity	Critical
Description	OCNADD Storage adapter service not available for more than 2 min

Table 5-18 (Cont.) OCNADD_STORAGE_ADAPTER_SVC_DOWN

Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Storage adapter service is down' **PromQL Expression:** expr: up{service=~".*storage-adapter.*", role="storageAdapter"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_STORAGE_ADAPTER_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="storageadapter"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="storageadapter"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.38.2002
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.
Resolution	The alert is cleared automatically when the OCNADD Storage adapter 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.

Table 5-19 OCNADD_INGRESS_ADAPTER_SVC_DOWN

Triggering Condition	The OCNADD Ingress Adapter service went down or not accessible
Severity	Critical
Description	OCNADD Ingress Adapter service not available for more than 2 min

Table 5-19 (Cont.) OCNADD_INGRESS_ADAPTER_SVC_DOWN

Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Ingress Adapter service is down' **PromQL Expression:** expr: up{service=~".*ingress-adapter.*", role="ingressadapter"} != 1
Alert Details OCI	**Summary:** Alarm "OCNADD_INGRESS_ADAPTER_SVC_DOWN" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "podStatus[10m]{podOwner="ingressadapter"}.mean()!=1", with a trigger delay of 1 minute **MQL Expression:** podStatus[10m]{podOwner="ingressadapter"}.mean()!=1
OID	1.3.6.1.4.1.323.5.3.53.1.36.2002
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.
Resolution	The alert is cleared automatically when the OCNADD Ingress Adapter 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.

Table 5-20 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 the configured warning threshold (80%) for the period of 5 min

Table 5-20 (Cont.) OCNADD_MPS_WARNING_INGRESS_THRESHOLD_CROSSED

Field	Details
Alert Details CNE	**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 }}' **PromQL Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.8*{{ .Values.global.cluster.mps }}
Alert Details OCI	**Summary:** Alarm "OCNADD_MPS_WARNING_INGRESS_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "kafka_stream_processor_node_process_total[10m] {microservice=~"*aggregation*"}.rate().groupBy(k8Namespace).sum()>.8*{{ MPS Threshold }}", with a trigger delay of 1 minute **MQL Expression:** kafka_stream_processor_node_process_total[10m] {microservice=~"*aggregation*"}.rate().groupBy(k8Namespace).sum()>.8*{{ MPS Threshold }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5007
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the MPS rate goes below the warning threshold level of 80%.

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 CNE	**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 }}' **PromQL Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.9*{{ .Values.global.cluster.mps }}

Table 5-21 (Cont.) OCNADD_MPS_MINOR_INGRESS_THRESHOLD_CROSSED

Field	Details
Alert Details OCI	**Summary:** Alarm "OCNADD_MPS_MINOR_INGRESS_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "kafka_stream_processor_node_process_total[10m] {microservice=~"*aggregation*"}.rate().groupBy(k8Namespace).sum()>.9*{{ MPS Threshold }}", with a trigger delay of 1 minute **MQL Expression:** kafka_stream_processor_node_process_total[10m] {microservice=~"*aggregation*"}.rate().groupBy(k8Namespace).sum()>.9*{{ MPS Threshold }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5008
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the MPS rate goes below the minor threshold alert level of 90%.

Table 5-22 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 the configured major threshold alert (95%) for the period of 5 min
Alert Details CNE	**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 }}' **PromQL Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.95*{{ .Values.global.cluster.mps }}
Alert Details OCI	**Summary**: Alarm "OCNADD_MPS_MAJOR_INGRESS_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "kafka_stream_processor_node_process_total[10m] {microservice=~"*aggregation*"}.rate().groupBy(k8Namespace).sum()>0.95*{{ MPS Threshold }}", with a trigger delay of 1 minute **MQL Expression:** kafka_stream_processor_node_process_total[10m] {microservice=~"*aggregation*"}.rate().groupBy(k8Namespace).sum()>0.95*{{ MPS Threshold }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5009
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the MPS rate goes below the major threshold alert level of 95%.

Table 5-23 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 CNE	**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 }}' **PromQL Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace) > 1.0*{{ .Values.global.cluster.mps }}
Alert Details OCI	**Summary**: Alarm "OCNADD_MPS_CRITICAL_INGRESS_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "kafka_stream_processor_node_process_total[10m] {microservice=~"*aggregation*"}.rate().groupBy(k8Namespace).sum()>1.0*{{ MPS Threshold }}", with a trigger delay of 1 minute **MQL Expression**: kafka_stream_processor_node_process_total[10m] {microservice=~"*aggregation*"}.rate().groupBy(k8Namespace).sum()>1.0*{{ MPS Threshold }}
OID	1.3.6.1.4.1.323.5.3.53.1.36.5010
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the MPS rate goes below the critical threshold alert level of 100% of supported MPS

Table 5-24 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 CNE	**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 }}' **PromQL Expression:** expr: sum(irate(kafka_stream_processor_node_process_total{service=~".*aggregation.*"}[5m])) by (namespace) > 1.0*{{ .Values.global.cluster.mps }}

Table 5-24 (Cont.) OCNADD_MPS_CRITICAL_INGRESS_THRESHOLD_CROSSED

Field	Details
Alert Details OCI	**Summary**: Alarm "OCNADD_MPS_CRITICAL_INGRESS_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "kafka_stream_processor_node_process_total[10m] {microservice=~'"*aggregation*"'}.rate().groupBy(k8Namespace).sum()>1.0*{{ MPS Threshold }}", with a trigger delay of 1 minute **MQL Expression**: kafka_stream_processor_node_process_total[10m] {microservice=~'"*aggregation*"'}.rate().groupBy(k8Namespace).sum()>1.0*{{ MPS Threshold }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5010
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the MPS rate goes below the critical threshold alert level of 100% of supported MPS

Table 5-25 OCNADD_MPS_WARNING_EGRESS_THRESHOLD_CROSSED

Field	Details
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 CNE	**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 }}' **PromQL Expression:** expr: sum (irate(ocnadd_egress_requests_total[5m])) by (namespace) > 0.80*{{ .Values.global.cluster.mps }}
Alert Details OCI	**Summary**: Alarm "OCNADD_MPS_WARNING_EGRESS_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_requests_total[10m] {app=~'"*adapter"}.rate().groupBy(worker_group,app).sum()>0.80*{{ MPS Threshold }}"", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_requests_total[10m] {app=~'"*adapter"}.rate().groupBy(worker_group,app).sum()>0.80*{{ MPS Threshold }}"
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

Table 5-26 OCNADD_MPS_MINOR_EGRESS_THRESHOLD_CROSSED

Field	Details
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
Alert Details CNE	**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 }}' **PromQL Expression:** expr: sum (irate(ocnadd_egress_requests_total[5m])) by (namespace) > 0.90*{{ .Values.global.cluster.mps }}
Alert Details OCI	**Summary**: Alarm "OCNADD_MPS_MINOR_EGRESS_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_requests_total[10m] {app=~"*adapter"}.rate().groupBy(worker_group,app).sum()>0.90*{{ MPS Threshold }}"", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_requests_total[10m] {app=~"*adapter"}.rate().groupBy(worker_group,app).sum()>0.90*{{ MPS Threshold }}
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

Table 5-27 OCNADD_MPS_MAJOR_EGRESS_THRESHOLD_CROSSED

Field	Details
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 CNE	**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 }}

Table 5-27 (Cont.) OCNADD_MPS_MAJOR_EGRESS_THRESHOLD_CROSSED

Field	Details
Alert Details OCI	**Summary**: Alarm "OCNADD_MPS_MAJOR_EGRESS_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_requests_total[10m] {app=~"*adapter"}.rate().groupBy(worker_group,app).sum()>0.95*{{ MPS Threshold }}"", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_requests_total[10m] {app=~"*adapter"}.rate().groupBy(worker_group,app).sum()>0.95*{{ MPS Threshold }}"
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

Table 5-28 OCNADD_MPS_CRITICAL_EGRESS_THRESHOLD_CROSSED

Field	Details
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 CNE	**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 }}' **PromQL Expression:** expr: sum (irate(ocnadd_egress_requests_total[5m])) by (namespace) > 1.0*{{ .Values.global.cluster.mps }}
Alert Details OCI	**Summary**: Alarm "OCNADD_MPS_CRITICAL_EGRESS_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_requests_total[10m] {app=~"*adapter"}.rate().groupBy(worker_group,app).sum()>1.0*{{ MPS Threshold }}"", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_requests_total[10m] {app=~"*adapter"}.rate().groupBy(worker_group,app).sum()>1.0*{{ MPS Threshold }}"
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

Table 5-29 OCNADD_MPS_CRITICAL_EGRESS_THRESHOLD_CROSSED_FOR_A_CONSUMER

Field	Details
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
Alert Details CNE	**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 }}
Alert Details OCI	**Summary**: Alarm "OCNADD_MPS_CRITICAL_EGRESS_THRESHOLD_CROSSED_FOR_A_CONSUMER" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_requests_total[10m] {app=~"*adapter"}.rate().groupBy(worker_group,app).sum()>1.0 {{ MPS Threshold }}", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_requests_total[10m] {app=~"*adapter"}.rate().groupBy(worker_group,app).sum()>1.0 {{ MPS Threshold }}
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

Table 5-30 OCNADD_E2E_AVG_RECORD_LATENCY_WARNING_THRESHOLD_CROSSED

Field	Details
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

Table 5-30 (Cont.)
OCNADD_E2E_AVG_RECORD_LATENCY_WARNING_THRESHOLD_CROSSED

Field	Details
Alert Details CNE	**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 }}
Alert Details OCI	**Summary**: Alarm "OCNADD_E2E_AVG_RECORD_LATENCY_WARNING_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()>0.040&&ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()<=0.045", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()>0.040&&ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()<=0.045 **Note: {{ .Values.global.cluster.max_latency }} is considered to be 0.05**
OID	1.3.6.1.4.1.323.5.3.53.1.29.5016
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

Table 5-31 OCNADD_E2E_AVG_RECORD_LATENCY_MINOR_THRESHOLD_CROSSED

Field	Details
Triggering Condition	The total observed latency is above the configured minor threshold alert level of 90%

Table 5-31 (Cont.)
OCNADD_E2E_AVG_RECORD_LATENCY_MINOR_THRESHOLD_CROSSED

Field	Details
Severity	Minor
Description	Average E2E Latency is above the configured minor threshold alert level (90%) for the period of 5 min
Alert Details CNE	**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' **PromQL 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 }}
Alert Details OCI	**Summary**: Alarm "OCNADD_E2E_AVG_RECORD_LATENCY_MINOR_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()>0.045&&ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()<=0.0475", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()>0.045&&ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()<=0.0475 **Note: {{ .Values.global.cluster.max_latency }} is considered to be 0.05**
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

Table 5-32 OCNADD_E2E_AVG_RECORD_LATENCY_MAJOR_THRESHOLD_CROSSED

Field	Details
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 CNE	**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' **PromQL 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 }}
Alert Details OCI	**Summary**: Alarm "OCNADD_E2E_AVG_RECORD_LATENCY_MAJOR_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()>0.0475&&ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()<=0.05", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()>0.0475&&ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()<=0.05 **Note: {{ .Values.global.cluster.max_latency }} is considered to be 0.05**
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

Table 5-33 OCNADD_E2E_AVG_RECORD_LATENCY_CRITICAL_THRESHOLD_CROSSED

Field	Details
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
Alert Details CNE	**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' **PromQL 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 }}
Alert Details OCI	**Summary**: Alarm "OCNADD_E2E_AVG_RECORD_LATENCY_CRITICAL_THRESHOLD_CROSSED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()>0.05", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_e2e_request_processing_latency_seconds_sum[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_e2e_request_processing_latency_seconds_count[10m].rate().groupBy(worker_group,app).sum()>0.05 **Note: {{ .Values.global.cluster.max_latency }} is considered to be 0.05**
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

Table 5-34 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 the 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 5-34 (Cont.) OCNADD_KAFKA_PACKET_DROP_THRESHOLD_1PERCENT_MPS

Field	Details
Alert Details CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Packet Drop rate is above 1% threshold of Max messages per second: {{ .Values.global.cluster.mps }}' **PromQL Expression:** expr: sum(rate(kafka_stream_task_dropped_records_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.01*{{ .Values.global.cluster.mps }}
Alert Details OCI	**Summary**: Alarm "OCNADD_KAFKA_PACKET_DROP_THRESHOLD_1PERCENT_MPS" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "kafka_stream_task_dropped_records_total[10m] {microservice=~"*aggregation"}.rate().groupBy(k8Namespace).sum()*100> {{ MPS Threshold }}", with a trigger delay of 1 minute **MQL Expression**: kafka_stream_task_dropped_records_total[10m] {microservice=~"*aggregation"}.rate().groupBy(k8Namespace).sum()*100> {{ MPS Threshold }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5020
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

Table 5-35 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 the 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 CNE	**Summary**: 'namespace: {{ "{{" }}$labels.namespace}}, podname: {{ "{{" }}$labels.pod}}, timestamp: {{ "{{" }} with query "time()" }}{{ "{{" }} . | first | value | humanizeTimestamp }}{{ "{{" }} end }}: Packet Drop rate is above 10% threshold of Max messages per second: {{ .Values.global.cluster.mps }}' **PromQL Expression:** expr: sum(rate(kafka_stream_task_dropped_records_total{service=~".*aggregation.*"}[5m])) by (namespace) > 0.1*{{ .Values.global.cluster.mps }}

Table 5-35 (Cont.) OCNADD_KAFKA_PACKET_DROP_THRESHOLD_10PERCENT_MPS

Field	Details
Alert Details OCI	**Summary**: Alarm "OCNADD_KAFKA_PACKET_DROP_THRESHOLD_10PERCENT_MPS" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "kafka_stream_task_dropped_records_total[10m] {microservice=~'"*aggregation"}.rate().groupBy(k8Namespace).sum()*100>10*{{MPS Threshold }}", with a trigger delay of 1 minute **MQL Expression**: kafka_stream_task_dropped_records_total[10m] {microservice=~'"*aggregation"}.rate().groupBy(k8Namespace).sum()*100>10*{{MPS Threshold }}
OID	1.3.6.1.4.1.323.5.3.53.1.29.5021
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

Table 5-36 OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_0.1PERCENT

Field	Details
Triggering Condition	The Egress adapter failure rate towards the 3rd party application is above the configured threshold of 0.1% of the total supported MPS
Severity	Info
Description	Egress external connection failure rate towards 3rd party application is crossing the info threshold of 0.1% in the period of 5 min
Alert Details CNE	**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' **PromQL 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

Table 5-36 (Cont.) OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_0.1PERCENT

Field	Details
Alert Details OCI	**Summary**: Alarm "OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_01PERCENT" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100>=0.1&&ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100<1", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100>=0.1&&ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100<1
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

Table 5-37 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 3rd party application is crossing the warning threshold of 1% in the period of 5 min
Alert Details CNE	**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' **PromQL Expression:** expr: (sum(rate(ocnadd_egress_failed_request_total[5m])) by (namespace))/(sum(rate(ocnadd_egress_requests_total[5m])) by (namespace)) *100 >= 1 < 10

Table 5-37 (Cont.) OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_1PERCENT

Field	Details
Alert Details OCI	**Summary**: Alarm "OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_1PERCENT" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100>=1&&ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100<10", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100>=1&&ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100<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
Resolution	The alert is cleared automatically when the failure rate towards third-party consumers goes below the threshold (1%) alert level of supported MPS

Table 5-38 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 the total supported MPS
Severity	Minor
Description	Egress external connection failure rate towards 3rd party application is crossing a minor threshold of 10% in the period of 5 min
Alert Details CNE	**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' **PromQL Expression:** expr: (sum(rate(ocnadd_egress_failed_request_total[5m])) by (namespace))/(sum(rate(ocnadd_egress_requests_total[5m])) by (namespace)) *100 >= 10 < 25

Table 5-38 (Cont.) OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_10PERCENT

Field	Details
Alert Details OCI	**Summary**: Alarm "OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_10PERCENT" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100>=10&&ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100<25", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100>=10&&ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100<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 consumers goes below the threshold (10%) alert level of supported MPS

Table 5-39 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 the total supported MPS
Severity	Major
Description	Egress external connection failure rate towards 3rd party application is crossing the major threshold of 25% in the period of 5 min
Alert Details CNE	**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' **PromQL Expression:** expr: (sum(rate(ocnadd_egress_failed_request_total[5m])) by (namespace))/(sum(rate(ocnadd_egress_requests_total[5m])) by (namespace)) *100 >= 25 < 50

Table 5-39 (Cont.) OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_25PERCENT

Field	Details
Alert Details OCI	**Summary**: Alarm "OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_25PERCENT" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100>=25&&ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100<50", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100>=25&&ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100<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 consumers goes below the threshold (25%) alert level of supported MPS

Table 5-40 OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_50PERCENT

Field	Details
Triggering Condition	The Egress adpater failure rate towards the 3rd party application is above the configured threshold of 50% of the total supported MPS
Severity	Critical
Description	Egress external connection failure rate towards 3rd party application is crossing the critical threshold of 50% in the period of 5 min
Alert Details CNE	**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 50 Percent of Total Egress external connections' **PromQL Expression:** expr:(sum(rate(ocnadd_egress_failed_request_total[5m])) by (namespace))/(sum(rate(ocnadd_egress_requests_total[5m])) by (namespace)) *100 >= 50

Table 5-40 (Cont.) OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_50PERCENT

Field	Details
Alert Details OCI	**Summary**: Alarm "OCNADD_EGRESS_FAILURE_RATE_THRESHOLD_50PERCENT" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*100>=50", with a trigger delay of 1 minute **MQL Expression**: ocnadd_egress_failed_request_total[10m].rate().groupBy(worker_group,app).sum()/ ocnadd_egress_requests_total[10m].rate().groupBy(worker_group,app).sum()*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 consumers goes below the threshold (50%) alert level of supported MPS

Table 5-41 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 the last 5 min
Alert Details CNE	**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' **PromQL 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
Alert Details OCI	**Not Available**
OID	1.3.6.1.4.1.323.5.3.53.1.29.5027
Metric Used	kafka_stream_processor_node_process_total
Resolution	The alert is cleared automatically when the increase in MPS comes back to lower than 10% of the supported MPS

Table 5-42 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

Table 5-42 (Cont.)
OCNADD_INGRESS_TRAFFIC_RATE_DECREASE_SPIKE_10PERCENT

Field	Details
Description	The ingress traffic decrease is more than 10% of the supported MPS in the last 5 min
Alert Details CNE	**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' **PromQL 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
Alert Details OCI	**Not Available**
OID	1.3.6.1.4.1.323.5.3.53.1.29.5028
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

Table 5-43 OCNADD_TRANSACTION_SUCCESS_CRITICAL_THRESHOLD_DROPPED

Field	Details
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
Alert Details CNE	**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

Table 5-43 (Cont.)
OCNADD_TRANSACTION_SUCCESS_CRITICAL_THRESHOLD_DROPPED

Field	Details
Alert Details OCI	**Summary**: Alarm "OCNADD_TRANSACTION_SUCCESS_CRITICAL_THRESHOLD_DROPPED" is in a "OK/FIRING" state; because 0/1 metrics meet the trigger rule: "ocnadd_total_transactions_total[10m]{status="SUCCESS",app=~"*corr*"}.rate().groupBy(workername,app).sum()/ocnadd_total_transactions_total[10m]{app=~"*corr*"}.rate().groupBy(workername,app).sum()*100<90", with a trigger delay of 1 minute **MQL Expression**: ocnadd_total_transactions_total[10m]{status="SUCCESS",app=~"*corr*"}.rate().groupBy(workername,app).sum()/ocnadd_total_transactions_total[10m]{app=~"*corr*"}.rate().groupBy(workername,app).sum()*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%