Oracle® Communications

Networks Data Analytics Function Troubleshooting Guide

Release 23.3.0
F84700-02
November 2023

Oracle Communications Networks Data Analytics Function Troubleshooting Guide, Release 23.3.0

F84700-02

Contents

My Oracle Support

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

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

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

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

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

Acronyms

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

Table Acronyms

Acronym	Description
3GPP	3rd Generation Partnership Project
5GC	5G Core Network
5GS	5G System
AF	Application Function
API	Application Programming Interface
AMF	Access and Mobility Management Function
AnLF	Analytics Logical Function
CAP4C	Converged Analytics Platform for Communication
CNC	Cloud Native Core
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
CSP	Communications Service Provider
FE	Front End
FQDN	Fully Qualified Domain Name
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure
KPI	Key Performance Indicator
HA	High Availability
IMSI	International Mobile Subscriber Identity
K8s	Kubernetes
MDT	Mobile Data Terminal
ME	Monitoring Events
MICO	Mobile Initiated Connection Only
ML	Machine Learning
MLOPs	Machine Learning Operations
MTLF	Model Training Logical Function
Network Slice	A logical network that provides specific network capabilities and network characteristics.
NEF	Oracle Communications Cloud Native Core, Network Exposure Function
NF	Network Function
NRF	Oracle Communications Cloud Native Core, Network Repository Function
NSI	Network Slice instance. A set of Network Function instances and the required resources (such as compute, storage and networking resources) which form a deployed Network Slice.
NSSF	Oracle Communications Cloud Native Core, Network Slice Selection Function
OCNWDAF	Oracle Communications Networks Data Analytics Function

Table (Cont.) Acronyms

Acronym	Description
OAM	Operations, Administration, and Maintenance
PLMN	Public Land Mobile Network
RAN	Radio Access Network
REST	Representational State Transfer
SBA	Service Based Architecture
SBI	Service Based Interface
SMF	Session Management Function
SNMP	Simple Network Management Protocol
SUPI	Subscription Permanent Identifier
UDM	Unified Data Management
UE	User Equipment
UPF	User Plane Function
UDR	Oracle Communications Cloud Native Core, Unified Data Repository
UDM	Unified Data Management
URI	Uniform Resource Identifier

What's New in This Guide

This section introduces the documentation updates for Release 23.3.x in Oracle Communications Networks Data Analytics Function Troubleshooting Guide.

Release 23.3.0.0.1 - F84700-02, November 2023

There are no updates to this document in this release.

Release 23.3.0 - F84700-01, September 2023

The following troubleshooting procedures are added:

- Service Account Creation in Openshift Environment
- Incorrect Values in Helm Chart
- Stream Transformation or Storage Not Working

The Debug Tool procedures are updated:

- Debug Tool
- Deploy Debug Tool
- Tools Tested in Debug Container
- Debug Tool Configuration Parameters

1 Introduction

This document provides information about troubleshooting Oracle Communications Network Data Analytics Function (OCNWDAF).

1.1 Overview

Oracle Communications Network Data Analytics Function (OCNWDAF) is a Network Function (NF) in the 5G core network of the 5G Network Architecture.

The OCNWDAF enables the operator to collect and analyze the data in the network through an analytics function. The 5G technology requires prescriptive analytics to drive closed-loop automation and self-healing networks. In a 5G network, the consumers of data are 5G NFs, Application Functions (AFs), and Operations, Administration, and Maintenance (OAM) and the data producers are NFs.

1.2 References

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

- *Oracle Communications Networks Data Analytics Function Installation and Fault Recovery Guide*
- *Oracle Communications Networks Data Analytics Function User Guide*
- *Oracle Communications Networks Data Analytics Function Solution Guide*
- *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide.*

2
Logs

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

2.1 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 OCNWDAF, 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.

> **Note**
>
> Log levels are defined in the helm chart and as parameters of the Kubernetes pod, they can be updated by changing the Kubernetes pod deployment.

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.

The following table provides log level details that may be helpful to handle different NRF Client Service debugging issues:

Table 2-1 Log Levels

Scenarios	Pod	Logs to be searched	Log Level
Registration with NRF Successful	nrf-client-service	Register completed successfully / "nfServiceStatus":"REGISTERED"	INFO

Table 2-1 (Cont.) Log Levels

Scenarios	Pod	Logs to be searched	Log Level
Heartbeat message log	nrf-client-service	Update completed successfully	INFO
NRF configurations reloading	nrf-client-service	NRF client config reloaded	INFO
Check for exiting NF Instance Entry	nrf-client-service	No registered NF instance exists	WARN
Started Application	nrf-client-service	Successful application start	INFO
NRF Client Config Initialized	nrf-client-service	Initialize NRF client configuration	INFO
FQDN/BASEURL/ livenessProbeUrl Improper	nrf-client-service	response=<503,java.net. UnknownHostException	WARN
nudr-drservice liveness probe failure	nrf-client-service	NFService liveness probe failed	WARN
Check if Ports successfully listening	nrf-client-service	Undertow started on port(s)	INFO
Registration with NRF failed	nrf-client-service	Register failed	ERROR
De registration with NRF successful	nrf-client-service	Deregister completed successfully	INFO
De registration with NRF failed	nrf-client-service	Deregister failed	ERROR
NF Profile update failed	nrf-client-service	Update failed	ERROR

2.2 Collecting Logs

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

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

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

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

   ```
   kubectl logs <podname> -n <namespace> -c <containername>
   ```

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

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

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

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

For more information on how to collect the logs, see *Oracle Communication Cloud Native Core Data Collector Guide*.

2.3 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 ran.

nfDataCapture.sh is a script which can be used for collecting all the required logs from NF deployment for debugging issues. The script collects logs from all microservice PODs of specified Helm input, Helm deployment details, the status, description of all the Kafka topics, *status.server* properties, and description of all the pods, services and events.

Before running the script, ensure the following requirements are met:

- Ensure that you have appropriate privileges to access the system and run `kubectl` and `helm` commands.
- Perform this procedure on the same machine where the OCNWDAF is deployed using `helm` or `kubectl`.
- Run the **chmod +x nfDataCapture.sh** command on the tool to provide the executable permission.
- Run the following command to start the module:

```
./nfDataCapture.sh -n|--k8Namespace=[K8 Namespace] -k|--
kubectl=[KUBE_SCRIPT_NAME] -h|--helm=[HELM_SCRIPT_NAME]  -s|--
size=[SIZE_OF_EACH_TARBALL] -o|--toolOutputPath -helm3=[true|false]
```

Where:

- `<K8 Namespace>` is the Kubernetes Namespace where OCNWDAF is deployed.
- `<KUBE_SCRIPT_NAME>` is the Kube script name.
- `<HELM_SCRIPT_NAME>` is the Helm script name.
- `<SIZE_OF_EACH_TARBALL>` indicates the size of each tarball.

Example:

```
./nfDataCapture.sh --k8Namespace=ocnwdaf-ns
```

> **Note**
>
> - If the size of the tarball and location are not specified, a default sized tarball will be generated (10M) and the default location of output will be the tool working directory.
> - Kafka Detailed Status is true by default and if we do not want to collect the details we have to specify the argument `false` in the command.
> - By default, Helm2 is used. Use proper argument in command to use Helm3.

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

To verify the generated tars, run the commands:

```
cd <generated-tarball-name>
ls
```

- Only if the size of the tar generated is greater than "SIZE_OF_EACH_TARBALL" specified in the command (for example, ocnwdaf.debugData.2023.02.28_09.15.01.tar.gz), the tar is split into multiple tarballs based on the size specified.
- After running the command, tarballs will be created based on size specified in the following format:
  `<namespace>.debugData.<timestamp>`

Example:

```
ocnwdaf.debugData.2023.02.28_09.15.01
```

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

```
cat <splitted files*> <combinedTarBall>.tar.gz
```

```
cat ocnwdaf.debugData.2023.02.28_09.15.01* >
ocnwdaf.debugData.2023.02.28_09.15.01-combined.tar.gz
```

2.4 Understanding Logs

This chapter explains the logs you need to look into to handle different OCNWDAF debugging issues.

For more information on how to collect the logs, see *Oracle Communication Cloud Native Core Data Collector Guide*.

Log Formats

OCNWDAF supports the following log formats:

- **Executor logs**
  Format:

  ```
  <datetime> - <level> - <module>.<line> [<thread>] : <message>
  ```

  Where:
  - `datetime` - The date and time of the event.
  - `level` - Helps in defining the severity level of a log message.
  - `module` - Software component that created the message.
  - `line` - Line of the source code where the message happened.
  - `thread` - Name of the thread that is currently running.

  - `message` - Description of the event.

- **Controller logs**
  Format:

  ```
  <datetime>  <level> <process> --- [<thread>] <loggername> : <message>
  ```

  Where:

  - `datetime` - The date and time of the event.
  - `level` - Helps in defining the severity level of a log message.
  - `process` - Name of the process that is currently running.
  - `thread` - Name of the thread that is currently running.
  - `loggername` - The source class name (often abbreviated).
  - `message` - Description of the event.

3 Debug Tool

The Debug Tool provides third-party troubleshooting tools for debugging the runtime issues for the lab environment.

Following are the available tools:

- tcpdump
- ip
- netstat
- curl
- ping
- nmap
- dig

3.1 Preconfiguration Steps

This section explains the preconfiguration steps for using the debug tool:

> **Note**
>
> - For CNE 23.2.0 and later versions, follow Step a of Configuration in CNE.
> - For CNE versions prior to 23.2.0, follow Step b of Configuration in CNE.

1. ***Configuration in CNE***
   The following configurations must be performed in the Bastion Host.
   a. When OCNWDAF is installed on CNE version 23.2.0 or above:

> **Note**
>
> - In CNE version 23.2.0 or above, the default CNE 23.2.0 Kyverno policy, disallow-capabilities, do not allow NET_ADMIN and NET_RAW capabilities that are required for debug tool.
> - To run Debug tool on CNE 23.2.0 and above, the user must modify the existing Kyverno policy, disallow-capabilities, as below.

Adding a Namespace to an Empty Resource

i. Run the following command to verify if the current disallow-capabilities cluster policy has namespace in it.

Example:

```
$ kubectl get clusterpolicies disallow-capabilities -oyaml
```

Sample output:

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
...
...
spec:
  rules:
  -exclude:
      any:
      -resources:{}
```

ii. If there are no namespaces, then patch the policy using the following command to add <namespace> under resources:

```
$ kubectl patch clusterpolicy disallow-capabilities --type=json \
  -p='[{"op": "add", "path": "/spec/rules/0/exclude/any/0/
resources", "value": {"namespaces":["<namespace>"]} }]'
```

Example:

```
$ kubectl patch clusterpolicy disallow-capabilities --type=json \
  -p='[{"op": "add", "path": "/spec/rules/0/exclude/any/0/
resources", "value": {"namespaces":["ocnwdaf"]} }]'
```

Sample output:

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
...
...
spec:
  rules:
  -exclude:
      resources:
        namespaces:
        -ocnwdaf
```

iii. If in case it is needed to remove the namespace added in the above step, use the following command:

```
$ kubectl patch clusterpolicy disallow-capabilities --type=json \
  -p='[{"op": "replace", "path": "/spec/rules/0/exclude/any/0/
resources", "value": {} }]'
```

Sample output:

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
```

```
...
...
spec:
  rules:
  -exclude:
      any:
      -resources:{}
```

Adding a Namespace to an Existing Namespace List

i. Run the following command to verify if the current disallow-capabilities cluster policy has namespaces in it.
Example:

```
$ kubectl get clusterpolicies disallow-capabilities -oyaml
```

Sample output:

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
...
...
spec:
  rules:
  -exclude:
      any:
      -resources:
          namespaces:
          -namespace1
          -namespace2
          -namespace3
```

ii. If there are namespaces already added, then patch the policy using the following command to add <namespace> to the existing list:

```
$ kubectl patch clusterpolicy disallow-capabilities --type=json \
  -p='[{"op": "add", "path": "/spec/rules/0/exclude/any/0/resources/
namespaces/-", "value": "<namespace>" }]'
```

Example:

```
$ kubectl patch clusterpolicy disallow-capabilities --type=json \
  -p='[{"op": "add", "path": "/spec/rules/0/exclude/any/0/resources/
namespaces/-", "value": "ocnwdaf" }]'
```

Sample output:

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
...
...
spec:
  rules:
  -exclude:
```

```
    resources:
      namespaces:
      -namespace1
      -namespace2
      -namespace3
      -ocnwdaf
```

iii. If in case it is needed to remove the namespace added in the above step, use the following command:

```
$ kubectl patch clusterpolicy disallow-capabilities --type=json \
  -p='[{"op": "remove", "path": "/spec/rules/0/exclude/any/0/
resources/namespaces/<index>"}]'
```

Example:

```
$ kubectl patch clusterpolicy disallow-capabilities --type=json \
  -p='[{"op": "remove", "path": "/spec/rules/0/exclude/any/0/
resources/namespaces/3"}]'
```

Sample output:

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
...
...
spec:
  rules:
  -exclude:
      resources:
        namespaces:
        -namespace1
        -namespace2
        -namespace3
```

> **Note**
>
> While removing the namespace, provide the index value for namespace within the array. The index starts from '0'.

b. When OCNWDAF is installed on CNE version prior to 23.2.0
PodSecurityPolicy (PSP) Creation

Create a PSP by running the following command from the bastion host. The parameters **readOnlyRootFileSystem**, **allowPrivilegeEscalation**, and **allowedCapabilities** are required by debug container.

> **Note**
>
> Other parameters are mandatory for PSP creation and can be customized as per the CNE environment. The default values are recommended.

```
$ kubectl apply -f - <<EOF

apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: debug-tool-psp
spec:
  readOnlyRootFilesystem: false
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - NET_ADMIN
  - NET_RAW
  fsGroup:
    ranges:
    - max: 65535
      min: 1
    rule: MustRunAs
  runAsUser:
    rule: MustRunAsNonRoot
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
  - configMap
  - downwardAPI
  - emptyDir
  - persistentVolumeClaim
  - projected
  - secret
EOF
```

Role Creation

Run the following command to create a role for the PSP:

```
kubectl apply -f - <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: debug-tool-role
  namespace: cncc
rules:
- apiGroups:
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

```
  resourceNames:
  - debug-tool-psp
EOF
```

RoleBinding Creation

Run the following command to associate the service account for the OCNWDAF namespace with the role created for the PSP:

```
$ kubectl apply -f - <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: debug-tool-rolebinding
  namespace: ocnwdaf
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: debug-tool-role
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
EOF
```

2. ***Configuration in NF specific Helm***
   Following updates must be performed in custom_values.yaml file.
   a. Log in to the NF server.
   b. Open the custom_values file:

```
$ vim <custom_values file>
```

   c. Under global configuration, add the following:

```
# Allowed Values: DISABLED, ENABLED
# Preference is to set  "resources" request and limit to same values to
avoid HPA issues.
extraContainers: DISABLED
debugToolContainerMemoryLimit: 4Gi
extraContainersVolumesTpl: |
  - name: debug-tools-dir
    emptyDir:
      medium: Memory
      sizeLimit: {{ .Values.global.debugToolContainerMemoryLimit |
quote }}
extraContainersTpl: |
    - command:
        - /bin/sleep
        - infinity
      image: <image-name>:<image-tag>
      imagePullPolicy: Always
      name: tools
      resources:
        requests:
          ephemeral-storage: "512Mi"
```

```
          cpu: "0.5"
          memory: {{ .Values.global.debugToolContainerMemoryLimit |
quote }}
        limits:
          ephemeral-storage: "512Mi"
          cpu: "0.5"
          memory: {{ .Values.global.debugToolContainerMemoryLimit |
quote }}
      securityContext:
        allowPrivilegeEscalation: true
        capabilities:
          drop:
          - ALL
          add:
          - NET_RAW
          - NET_ADMIN
        runAsUser: <user-id>
      volumeMounts:
      - mountPath: /tmp/tools
        name: debug-tools-dir
```

> **Note**
>
> - Debug Tool Container comes up with the default user ID - 7000. If you want to override this default value, use the `runAsUser` field, or else, you can skip the field.
>
>   Default value: uid=7000(debugtool) gid=7000(debugtool) groups=7000(debugtool)
> - In case you want to customize the container name, replace the `name` field in the above values.yaml with the following:
>
>   ```
>   name: {{ printf "%s-tools-%s" (include "getprefix" .)
>   (include "getsuffix" .) | trunc 63 | trimPrefix "-" |
>   trimSuffix "-"  }}
>   ```
>
>   This will ensure that the container name is prefixed and suffixed with the necessary values.

d. Under service specific configurations for which debugging is required, add the following:

```
# Allowed Values: DISABLED, ENABLED, USE_GLOBAL_VALUE
extraContainers: USE_GLOBAL_VALUE
```

> **Note**
>
> - At the global level, `extraContainers` flag can be used to enable or disable injecting extra containers globally. This ensures that all the services that use this global value have extra containers enabled or disabled using a single flag.
> - At the service level, `extraContainers` flag determines whether to use the extra container configuration from the global level or enable or disable injecting extra containers for the specific service.

3.2 Deploy Debug Tool

Following is the procedure to run Debug Tool.

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

   ```
   $ kubectl get pods -n <k8s namespace>
   ```

   Example:

   ```
   $ kubectl get pods -n ocnwdaf
   ```

2. Run the following command to enter Debug Tool Container:

   ```
   $ kubectl exec -it <pod name> -c <debug_container name> -n <namespace> bash
   ```

3. Run the debug tools:

   ```
   bash -4.2$ <debug_tools>
   ```

   Example:

   ```
   bash -4.2$ tcpdump
   ```

4. Copy the output files from container to host:

   ```
   $ kubectl cp -c <debug_container name> <pod name>:<file location in
   container> -n <namespace> <destination location>
   ```

3.3 Tools Tested in Debug Container

Following is the list of debugging tools that are tested.

tcpdump

The following table describes the options that are testing using `tcpdump` tool.

Table 3-1 tcpdump

Options Tested	Description	Output	Capabilities
-D	Print the list of the network interfaces available on the system and on which *tcpdump* can capture packets.	`tcpdump -D` 1. eth02. 2. nflog (Linux netfilter log (NFLOG) interface) 3. nfqueue (Linux netfilter queue (NFQUEUE) interface) 4. any (Pseudo-device that captures on all interfaces) 5. lo [Loopback]	NET_ADMIN, NET_RAW
-i	Listen on *interface*	`tcpdump -i eth0` tcpdump: verbose output suppressed, use -v or -vv for full protocol decodelistening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes12:10:37.381199 IP cncc-core-ingress-gateway-7ffc49bb7f-2kkhc.46519 > kubernetes.default.svc.cluster.local.https: Flags [P.], seq 1986927241:1986927276, ack 1334332290, win 626, options [nop,nop,TS val 849591834 ecr 849561833], length 3512:10:37.381952 IP cncc-core-ingress-gateway-7ffc49bb7f-2kkhc.45868 > kube-dns.kube-system.svc.cluster.local.domain: 62870+ PTR? 1.0.96.10.in-addr.arpa. (40)	NET_ADMIN, NET_RAW
-w	Write the raw packets to file rather than parsing and printing them out.	`tcpdump -w capture.pcap -i eth0`	NET_ADMIN, NET_RAW
-r	Read packets from *file* (which was created with the **-w** option).	`tcpdump -r capture.pcap` reading from file /tmp/capture.pcap, link-type EN10MB (Ethernet)12:13:07.381019 IP cncc-core-ingress-gateway-7ffc49bb7f-2kkhc.46519 > kubernetes.default.svc.cluster.local.https: Flags [P.], seq 1986927416:1986927451, ack 1334332445, win 626, options [nop,nop,TS val 849741834 ecr 849711834], length 3512:13:07.381194 IP kubernetes.default.svc.cluster.local.https > cncc-core-ingress-gateway-7ffc49bb7f-2kkhc.46519: Flags [P.], seq 1:32, ack 35, win 247, options [nop,nop,TS val 849741834 ecr 849741834], length 3112:13:07.381207 IP cncc-core-ingress-gateway-7ffc49bb7f-2kkhc.46519 > kubernetes.default.svc.cluster.local.https: Flags [.], ack 32, win 626, options [nop,nop,TS val 849741834 ecr 849741834], length 0	NET_ADMIN, NET_RAW

ip

The following table describes the options that are testing using `ip` tool.

Table 3-2 ip

Options Tested	Description	Output	Capabilities
addr show	Look at protocol addresses.	`ip addr show` 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaultlink/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00inet 127.0.0.1/8 scope host lovalid_lft forever preferred_lft forever2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group defaultlink/ipip 0.0.0.0 brd 0.0.0.04: eth0@if190: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1440 qdisc noqueue state UP group defaultlink/ether aa:5a:27:8d:74:6f brd ff:ff:ff:ff:ff:ff link-netnsid 0inet 192.168.219.112/32 scope global eth0valid_lft forever preferred_lft forever	--
route show	List routes	`ip route show` default via 169.254.1.1 dev eth0 169.254.1.1 dev eth0 scope link	--
addrlabel list	List address labels	`ip addrlabel list` prefix ::1/128 label 0 prefix ::/96 label 3 prefix ::ffff:0.0.0.0/96 label 4 prefix 2001::/32 label 6 prefix 2001:10::/28 label 7 prefix 3ffe::/16 label 12 prefix 2002::/16 label 2 prefix fec0::/10 label 11 prefix fc00::/7 label 5 prefix ::/0 label 1	--

netstat

The following table describes the options that are testing using `netstat` tool.

Table 3-3 netstat

Options Tested	Description	Output	Capabilities
-a	Show both listening and non-listening (for TCP, this means established connections) sockets.	`netstat -a` Active Internet connections (servers and established)Proto Recv-Q Send-Q Local Address Foreign Address Statetcp 0 0 0.0.0.0:tproxy 0.0.0.0:* LISTENtcp 0 0 0.0.0.0:websm 0.0.0.0:* LISTENtcp 0 0 cncc-core-ingress:websm 10-178-254-194.ku:47292 TIME_WAITtcp 0 0 cncc-core-ingress:46519 kubernetes.defaul:https ESTABLISHEDtcp 0 0 cncc-core-ingress:websm 10-178-254-194.ku:47240 TIME_WAITtcp 0 0 cncc-core-ingress:websm 10-178-254-194.ku:47347 TIME_WAITudp 0 0 localhost:59351 localhost:ambit-lm ESTABLISHEDActive UNIX domain sockets (servers and established)Proto RefCnt Flags Type State I-Node Pathunix 2 [ ] STREAM CONNECTED 576064861	--

Table 3-3 (Cont.) netstat

Options Tested	Description	Output	Capabilities
-l	Show only listening sockets.	`netstat -l` Active Internet connections (only servers)Proto Recv-Q Send-Q Local Address Foreign Address Statetcp 0 0 0.0.0.0:tproxy 0.0.0.0:* LISTENtcp 0 0 0.0.0.0:websm 0.0.0.0:* LISTENActive UNIX domain sockets (only servers)Proto RefCnt Flags Type State I-Node Path	--
-s	Display summary statistics for each protocol.	`netstat -s` Ip:4070 total packets received0 forwarded0 incoming packets discarded4070 incoming packets delivered4315 requests sent outIcmp:0 ICMP messages received0 input ICMP message failed.ICMP input histogram:2 ICMP messages sent0 ICMP messages failedICMP output histogram:destination unreachable: 2	--
-i	Display a table of all network interfaces.	`netstat -i` Kernel Interface tableIface MTU RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flgeth0 1440 4131 0 0 0 4355 0 0 0 BMRUlo 65536 0 0 0 0 0 0 0 0 LRU	--

curl

The following table describes the options that are testing using `curl` tool.

Table 3-4 curl

Options Tested	Description	Output	Capabilities
-o	Write output to <file> instead of stdout.	`curl -o file.txt http://abc.com/file.txt`	--
-x	Use the specified HTTP proxy.	`curl -x proxy.com:8080 -o http://abc.com/file.txt`	--

ping

The following table describes the options that are testing using `ping` tool.

Table 3-5 ping

Options Tested	Description	Output	Capabilities
<ip>	Run a ping test to see whether the target host is reachable or not.	`ping 10.178.254.194`	NET_ADMIN, NET_RAW
-c	Stop after sending 'c' number of ECHO_REQUEST packets.	`ping -c 5 10.178.254.194`	NET_ADMIN, NET_RAW

Table 3-5 (Cont.) ping

Options Tested	Description	Output	Capabilities
-f (with non zero interval)	Flood ping. For every ECHO_REQUEST sent, a period "." is printed, while for every ECHO_REPLY received a backspace is printed.	`ping -f -i 2 10.178.254.194`	NET_ADMIN, NET_RAW

nmap

The following table describes the options that are testing using `nmap` tool.

Table 3-6 nmap

Options Tested	Description	Output	Capabilities

<ip>	Scan for Live hosts, Operating systems, packet filters and open ports running on remote hosts.	`nmap 10.178.254.194`
		
		`Starting Nmap 6.40 ( http://nmap.org ) at 2020-09-29 05:54 UTCNmap scan report for`
		
		`10-178-254-194.kubernetes.default.svc.cluster .local (10.178.254.194)Host is up (0.00046s latency).Not shown: 995 closed portsPORT STATE SERVICE22/tcp open ssh179/tcp open bgp6666/tcp open irc6667/tcp open irc30000/tcp open unknownNmap done: 1 IP address (1 host up) scanned in 0.04 seconds`
			--

Table 3-6 (Cont.) nmap

Options Tested	Description	Output	Capabilities

-v	Increase verbosity level	`nmap -v 10.178.254.194`
		
		`Starting Nmap 6.40 ( http://nmap.org ) at 2020-09-29 05:55 UTC`
		`Initiating Ping Scan at 05:55`
		`Scanning 10.178.254.194 [2 ports]`
		`Completed Ping Scan at 05:55, 0.00s elapsed (1 total hosts)`
		`Initiating Parallel DNS resolution of 1 host. at 05:55`
		`Completed Parallel DNS resolution of 1 host. at 05:55, 0.00s elapsed`
		`Initiating Connect Scan at 05:55`
		`Scanning 10-178-254-194.kubernetes.default.svc.cluster.local (10.178.254.194) [1000 ports]`
		`Discovered open port 22/tcp on 10.178.254.194`
		`Discovered open port 30000/tcp on 10.178.254.194`
		`Discovered open port 6667/tcp on 10.178.254.194`
		`Discovered open port 6666/tcp on 10.178.254.194`
		`Discovered open port 179/tcp on 10.178.254.194`
		`Completed Connect Scan at 05:55, 0.02s elapsed (1000 total ports)`
		`Nmap scan report for 10-178-254-194.kubernetes.default.svc.cluster.local (10.178.254.194)`
		`Host is up (0.00039s latency).`
		`Not shown: 995 closed ports`
		`PORT STATE SERVICE`
		`22/tcp open ssh`
		`179/tcp open bgp`
		`6666/tcp open irc`
		`6667/tcp open irc`
		`30000/tcp open unknown`
		
		`Read data files from: /usr/bin/../share/nmap`
		`Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds`
			--

Table 3-6 (Cont.) nmap

Options Tested	Description	Output	Capabilities

-iL	Scan all the listed IP addresses in a file. Sample file	`nmap -iL sample.txt`
		
		`Starting Nmap 6.40 ( http://nmap.org ) at 2020-09-29 05:57 UTC`
		`Nmap scan report for localhost (127.0.0.1)`
		`Host is up (0.00036s latency).`
		`Other addresses for localhost (not scanned): 127.0.0.1`
		`Not shown: 998 closed ports`
		`PORT STATE SERVICE`
		`8081/tcp open blackice-icecap`
		`9090/tcp open zeus-admin`
		
		`Nmap scan report for 10-178-254-194.kubernetes.default.svc.cluster.local (10.178.254.194)`
		`Host is up (0.00040s latency).`
		`Not shown: 995 closed ports`
		`PORT STATE SERVICE`
		`22/tcp open ssh`
		`179/tcp open bgp`
		`6666/tcp open irc`
		`6667/tcp open irc`
		`30000/tcp open unknown`
		
		`Nmap done: 2 IP addresses (2 hosts up) scanned in 0.06 seconds`
			--

dig

The following table describes the options that are testing using `dig` tool.

Table 3-7 dig

Options Tested	Description	Output	Capabilities
<ip>	It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried.	`dig 10.178.254.194` **Note**: The IP should be reachable from inside the container.	--
-x	Query DNS Reverse lookup.	`dig -x 10.178.254.194`	--

3.4 Debug Tool Configuration Parameters

Following are the parameters used to configure debug tool.

CNE Parameters

Table 3-8 CNE Parameters

Parameter	Description
apiVersion	APIVersion defines the version schema of this representation of an object.
kind	Kind is a string value representing the REST resource this object represents.
metadata	Standard object's metadata.
metadata.name	Name must be unique within a namespace.
spec	This defines the policy enforced.
spec.allowPrivilegeEscalation	Gates whether or not a user is allowed to set the security context of a container to allowPrivilegeEscalation=true.
spec.allowedCapabilities	Provides a list of capabilities that are allowed to be added to a container.
spec.fsGroup	Controls the supplemental group applied to some volumes. RunAsAny allows any fsGroup ID to be specified.
spec.runAsUser	Controls which user ID the containers are run with. RunAsAny allows any runAsUser to be specified.
spec.seLinux	RunAsAny allows any seLinuxOptions to be specified.
spec.supplementalGroups	Controls which group IDs containers add. RunAsAny allows any supplementalGroups to be specified.
spec.volumes	Provides a list of allowed volume types. The allowable values correspond to the volume sources that are defined when creating a volume.

Role Creation Parameters

Table 3-9 Role Creation

Parameter	Description
apiVersion	APIVersion defines the schema version of this representation of an object.
kind	Kind is a string value representing the REST resource this object represents.
metadata	Standard object's metadata.
metadata.name	Name must be unique within a namespace.
metadata.namespace	Namespace defines the space within which each name must be unique.
rules	Rules holds all the PolicyRules for this Role
apiGroups	APIGroups is the name of the APIGroup that contains the resources.
rules.resources	Resources is a list of resources this rule applies to.
rules.verbs	Verbs is a list of Verbs that apply to ALL the ResourceKinds and AttributeRestrictions contained in this rule.
rules.resourceNames	ResourceNames is an optional allowed list of names that the rule applies to.

Table 3-10 Role Binding Creation

Parameter	Description
apiVersion	APIVersion defines the version of schema of this representation of an object.
kind	Kind is a string value representing the REST resource this object represents.
metadata	Standard object's metadata.
metadata.name	Name must be unique within a namespace.
metadata.namespace	Namespace defines the space within which each name must be unique.
roleRef	RoleRef can reference a Role in the current namespace or a ClusterRole in the global namespace.
roleRef.apiGroup	APIGroup is the group for the resource being referenced
roleRef.kind	Kind is the type of resource being referenced
roleRef.name	Name is the name of resource being referenced
subjects	Subjects holds references to the objects the role applies to.
subjects.kind	Kind of object being referenced. Values defined by this API group are "User", "Group", and "ServiceAccount".
subjects.apiGroup	APIGroup holds the API group of the referenced subject.
subjects.name	Name of the object being referenced.

Debug Tool Configuration Parameters

Table 3-11 Debug Tool Configuration Parameters

Parameter	Description
extraContainers	Specifies the spawns debug container along with application container in the pod.
debugToolContainerMemoryLimit	Indicates the memory assigned for the debug tool container.
extraContainersVolumesTpl	Specifies the extra container template for the debug tool volume.
extraContainersVolumesTpl.name	Indicates the name of the volume for debug tool logs storage.
extraContainersVolumesTpl.emptyDir.medium	Indicates the location where `emptyDir` volume is stored.
extraContainersVolumesTpl.emptyDir.sizeLimit	Indicates the `emptyDir` volume size.
command	String array used for container command.
image	Docker image name
imagePullPolicy	Image Pull Policy
name	Name of the container
resources	Compute Resources required by this container
resources.limits	Limits describes the maximum amount of compute resources allowed
resources.requests	Requests describes the minimum amount of compute resources required
resources.limits.cpu	CPU limits
resources.limits.memory	Memory limits
resources.limits.ephemeral-storage	Ephemeral Storage limits

Table 3-11 (Cont.) Debug Tool Configuration Parameters

Parameter	Description
resources.requests.cpu	CPU requests
resources.requests.memory	Memory requests
resources.requests.ephemeral-storage	Ephemeral Storage requests
securityContext	Security options the container should run with.
securityContext.allowPrivilegeEscalation	AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This directly controls if the no_new_privs flag will be set on the container process
secuirtyContext.readOnlyRootFilesyste m	Whether this container has a read-only root filesystem. Default is false.
securityContext.capabilities	The capabilities to add or drop when running containers. Defaults to the default set of capabilities granted by the container runtime.
securityContext.capabilities.drop	Removed capabilities
secuirtyContext.capabilities.add	Added capabilities
securityContext.runAsUser	The UID to run the entry point of the container process.
volumeMounts.mountPath	Indicates the path for volume mount.
volumeMounts.name	Indicates the name of the directory for debug tool logs storage.

4
Troubleshooting OCNWDAF

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

4.1 Generic Checklist

The following sections provide a generic checklist for troubleshooting tips.

Deployment related tips

Perform the following checks after the deployment:

- Are OCNWDAF deployment, pods, and services created?
  Are OCNWDAF deployment, pods, and services running and available?

  Run the following the command:

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

  Inspect the output, check the following columns:

  – AVAILABLE of deployment
  – READY, STATUS, and RESTARTS of a pod
  – PORT(S) of service
- Check if the microservices can access each other through the REST interface.
  Run the following command:

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

Application related tips

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

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

You can use '-f' to follow the logs or 'grep' for specific pattern in the log output.

4.2 Deployment Related Issue

This section describes the most common deployment related issues and their resolution steps. It is recommended to perform the resolution steps provided in this guide. If the issue still persists, then contact My Oracle Support.

4.2.1 Installation

This section describes the most common installation related issues and their resolution steps.

4.2.1.1 Pod Creation Failure

A pod creation can fail due to various reasons. Some of the possible scenarios are as follows:

Verifying Pod Image Correctness

To verify pod image:

- Check whether any of the pods is in the **ImagePullBackOff** state.
- Check if the image name used for all the pods are correct. Verify the image names and versions in the OCNWDAF installation file. For more information about the custom value file, see *Oracle Communications Networks Data Analytics Function Installation and Fault Recovery Guide*.

Verifying Resource Allocation Failure

To verify any resource allocation failure:

- Run the following command to verify whether any pod is in the pending state.
  `kubectl describe <nwdaf-drservice pod id> --n <ocnef-namespace>`
- Verify whether any warning on insufficient CPU exists in the describe output of the respective pod. If it exists, it means there are insufficient CPUs for the pods to start. Address this hardware issue.

Verifying Resource Allocation Issues on Webscale Environment

Webscale environment has openshift container installed. There can be cases where,

- Pods does not scale after you run the installation command and the installation fails with timeout error. In this case, check for preinstall hooks failure. Run the **oc get job** command to create the jobs. Describe the job for which the pods are not getting scaled and check if there are **quota limit exceeded** errors with CPU or memory.
- Any of the actual microservice pods do not scale post the hooks completion. In this case, run the **oc get rs** command to get the list of replicaset created for the NF deployment.

Then, describe the replicaset for which the pods are not getting scaled and check for **resource quota limit exceeded** errors with CPU or memory.

- Installation times-out after all the microservice pods are scaled as expected with the expected number of replicas. In this case, check for post install hooks failure. Run the **oc get job** command to get the post install jobs and do a describe on the job for which the pods are not getting scaled and check if there are **quota limit exceeded** errors with CPU or memory.
- Resource quota exceed beyond limits.

Verifying Resources Assigned to Previous Installation

If a previous installations, uninstall procedure was not successful and the uninstall process was forced, it is possible that some resources are still assigned to the previous installation. This can be detected by running the following command:

```
kubectl -n <namepsace> describe pod <podname>
```

While searching for events, if you detect messages similar to the following message, it indicates that there are resources still assigned to the previous installation and should be purged.

```
0/n nodes are available: n pods has unbound immediate PersistenVolumeClaims
```

4.2.1.2 Pod Startup Failure

Follow the guidelines shared below to debug the pod startup failure liveness check issues:

- If dr-service, diameter-proxy, and diam-gateway services are stuck in the Init state, then the reason could be that config-server is not yet up. A sample log on these services is as follows:

  ```
  "Config Server is Not yet Up, Wait For config server to be up."
  ```

  To resolve this, you must either check for the reason of config-server not being up or if the config-server is not required, then disable it.
- If the notify and on-demand migration service is stuck in the Init state, then the reason could be the dr-service is not yet up. A sample log on these services is as follows:

  ```
  "DR Service is Not yet Up, Wait For dr service to be up."
  ```

  To resolve this, check for failures on dr-service.

4.2.1.3 NRF Registration Failure

The OCNWDAF registration with NRF may fail due to various reasons. Some of the possible scenarios are as follows:

- Confirm whether registration was successful from the nrf-client-service pod.
- Check the ocnwdaf-nrf-client-nfmanagement logs. If the log has **"OCNWDAF is Unregistered"** then:
  - Check if all the services mentioned under allorudr/slf (depending on OCNWDAF mode) in the installation file has same spelling as that of service name and are enabled.

  - Once all services are up, OCNWDAF must register with NRF.
- If you see a log for **SERVICE_UNAVAILABLE(503)**, check if the primary and secondary NRF configurations (primaryNrfApiRoot/secondaryNrfApiRoot) are correct and they are UP and Running.

4.2.1.4 Incorrect Service Account Creation

Problem

Pods display an error when appropriate service accounts are not created for the pods.

Error Code or Error Message

Sample error message:

Figure 4-1 Sample Error Message

```
[cloud-user@occne224-cluster-bastion-1 ~]$ kubectl describe pod/ocn-nwdaf-db-creation-hook-6rmpx -n ocn-nwdaf
Events:
  Type     Reason     Age              From               Message
  ----     ------     ----             ----               -------
  Normal   Scheduled  36s              default-scheduler  Successfully assigned ocn-nwdaf/ocn-nwdaf-db-creation-hook-6r
mpx to occne224-cluster-k8s-node-6
  Normal   Pulled     7s (x5 over 37s)  kubelet           Container image "occne-repo-host:5000/ocnwdaf-pre-install-hoo
k-image:1.1.3" already present on machine
  Warning  Failed     7s (x5 over 37s)  kubelet           Error: container has runAsNonRoot and image will run as root
(pod: "ocn-nwdaf-db-creation-hook-6rmpx_ocn-nwdaf(187172d0-e9c0-42b8-a468-305f1224af55)", container: ocnwdafhelmhook)
```

Solution

Ensure the service account creation hook in the parent chart's *values.yaml* file is enabled and runs properly.

4.2.1.5 Service Account Creation in Openshift Environment

Problem

While deploying OCNWDAF in an Openshift environment, service account creation can result in Helm installation issues.

Solution

Add the service account manually in the namespace before deployment, run the following command to add the service account:

```
oc adm policy add-scc-to-user anyuid --serviceaccount=<namespace>-ocnwdaf-sa -
n <namespace>
```

Where,

- `oc`: OpenShift CLI
- `adm`: Admin
- `scc`: Security Context Constraint

4.2.1.6 Incorrect Values in Helm Chart

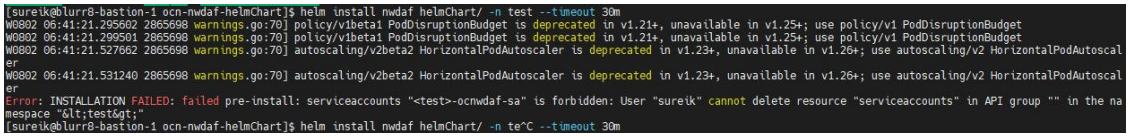
Problem

During Helm chart deployment, if instances of the command parameter `<replace here>` are not replaced with values or replaced with incorrect values, services are not deployed.

Error Code or Error Message

Sample error message:

Figure 4-2 Incorrect Values in Helm Chart



Solution

Uninstall the deployment and verify the *values.yaml* file in the Helm chart. Search for `<replace here>` instances and provide correct values.

4.2.1.7 Install Timeout Error

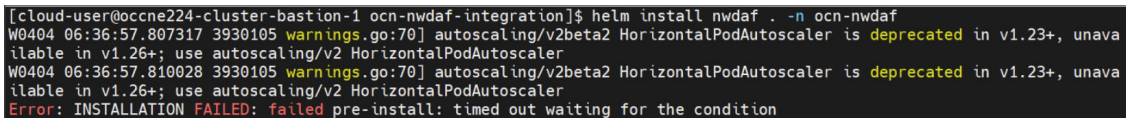
Problem

This error occurs when a hook restarts more than five times.

Error Code or Error Message

Sample error message:

Figure 4-3 Sample Error Message



Solution

Check whether the MySQL host or MySQL port is mentioned correctly in the *values.yaml* file of both the parent and the NRF client Helm charts. Verify the pod logs for more information.

Run the following command to verify the logs:

```
kubectl logs <name-of-pod/hook> -n <namespace>
```

4.2.1.8 Pods Enter Pending State

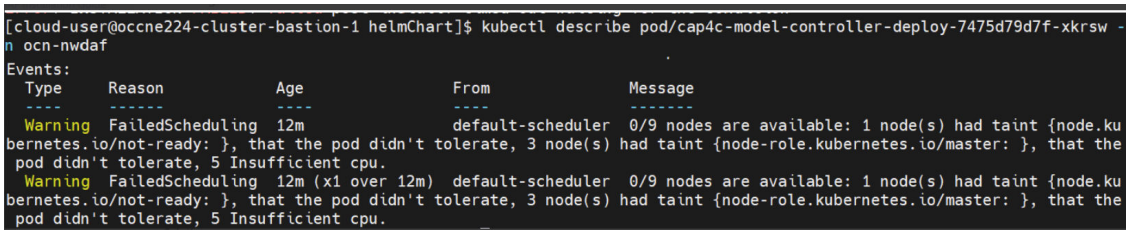
Problem

Pods enter a pending state due to resource shortage in the setup.

Error Code or Error Message

Sample error message:

Figure 4-4 Sample Error Message



Solution

Free up all unnecessary resources present in the cluster that are consuming a lot of cluster resources.

4.2.1.9 Resource Creation Failure

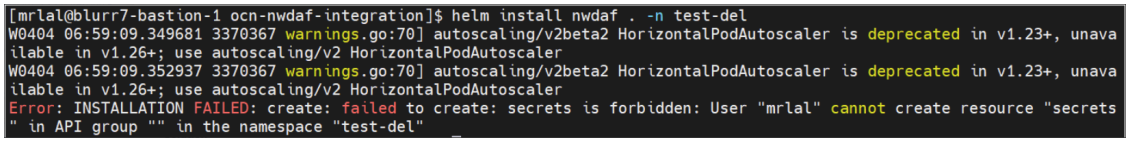
Problem

The deployment namespace does not have appropriate permissions to create resources.

Error Code or Error Message

Sample error message:

Figure 4-5 Sample Error Message



Solution

Create a child namespace for the parent namespace that has appropriate permissions, run the following command:

```
kubectl hns create <child-namespace> -n <parent-namespace>
```

4.2.1.10 Service Configuration or Parameter Mismatch

Problem

Service configuration or parameter mismatch might result in the service entering a *CrashBackLoop* off mode.

Solution

Update the properties in the corresponding services *values.yaml* file and perform a Helm install.

4.2.1.11 Service Nodeport Error

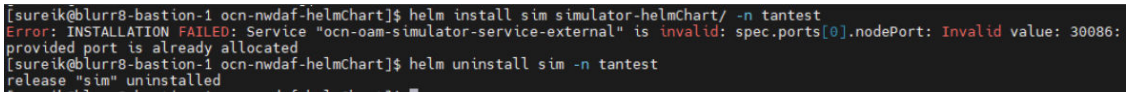
Problem

The service nodeport was previously assigned to other services running in the cluster.

Error Code or Error Message

Sample error message:

Figure 4-6 Sample Error Message



Solution

To resolve this error, edit the *values.yaml* file and provide a random port number to the service nodeport.

4.2.1.12 Common Services Gateway Service Name Mismatch

Problem

Suppose the service name of the common services gateway differs from "nwdaf-ingress-gateway-service" and "nwdaf-egress-gateway-service". This results in errors in the functioning of the gateways and forwarding of external requests to the respective services.

Solution

- Run the following command:

  ```
  kubectl edit service <service-name> -n <namespace>
  ```

- Edit the service names of the common services gateways to "nwdaf-ingress-gateway-service" for the Ingress Gateway and "nwdaf-egress-gateway-service" for the Egress Gateway respectively.

4.2.1.13 Run Only DB Creation Hook

Set the *dbConfigStatus* flag in *values.yaml* file under */helmchart* directory to *dbonly* to run only the DB creation hook. The Helm installation command will not deploy any other resource or make any other configuration change. Users can use different Helm installation names in the Helm install command to configure the latest database by updating the scripts *ocnwdaf-db-config.yaml* under */helmchart/templates* directory and *prehookconfig.yaml* under */helmchart/charts/ocn-nwdaf-geo-redundancy-agent/templates* directory.

4.2.1.14 Helm Chart Upgrade

Helm upgrade is performed for all deployment changes (for example, updating the image used in the microservice) that do not require a reinstallation. Run the following command to perform a Helm upgrade:

```
helm upgrade <installation name> <path to the updated chart directory> -
n $K8_NAMESPACE --timeout <timeout>h
```

> **Note**
>
> - Provide the correct installation name on which the installation was performed.
> - The `timeout` variable is optional. It is based on the speed of image pull from the nodes of the Bastion. The recommended timeout value is "4 h".
> - Helm upgrade must be performed only on the main Helm chart under */helmChart* directory. It must not be performed on the subcharts under */charts* directory. To update any subchart, make changes in the respective subchart and perform Helm upgrade on the main Helm chart under */helmChart* directory.
> - To enable DB creation hook or to prepare the dependencies hook, set the *upgradeStatus* flag in *values.yaml* file under */helmChart* directory to true before performing a Helm upgrade. To disable the hooks, set the *upgradeStatus* flag to false.
> - Before performing a Helm upgrade on the *ocn-nwdaf-communication*, *nwdaf-cap4c-zookeper-chart*, *nwdaf-cap4c-kafka-chart*, *ocn-nrf-simulator-service*, and *nwdaf-cap4c-spring-cloud-config-server-chart* services, set the *upgradeStatus* flag in *values.yaml* file under */helmChart* directory to true. If there are no changes in the services, set the *upgradeStatus* flag to false.
> - Use the prepare dependencies hook for Helm upgrade only when the *upgradeStatus* flag for *nwdaf-cap4c-kafka-chart* and *nwdaf-cap4c-spring-cloud-config-server-chart* microservices is set to false. To upgrade these microservices with the prepare dependencies hook, use the prepare dependencies hook in a separate Helm upgrade procedure, then perform an upgrade of the microservices.

Listed below are some use cases for performing a Helm upgrade:

- To update the fields such as image name, resources allotted, environment variables, and so on, make the required changes in the respective subcharts and run the Helm upgrade command on the updated chart.
- To enable or disable services, set the subcharts enable or disable flag in the centralized *values.yaml* file under the */helmchart* directory to true or false (as required). The services with enable flag set to false are terminated.
- To reinstall the DB, enable the *dbCreationHook upgradeStatus* flag in *values.yaml* file under */helmChart* directory to true. The DB creation hook runs according to the configured *dbConfigStatus* flag in the file. For example, if the *dbConfigStatus* flag is set to *nwdafdb*, only the *nwdafdb* creation hook is run during upgrade.
- To transfer Spring Cloud config files from *nwdaf-pre-installer.tar.gz* to the *spring-cloud-config-server* microservice, and to create new Kafka topics in the Kafka microservice, use the prepare dependencies hook by updating *prepareDependencyHook upgradeStatus* flag in *values.yaml* file under */helmChart* directory to true. The Kafka pods and Spring Cloud Config server pods must be in *Ready* State before enabling *upgradeStatus* flag in *values.yaml* file under */helmChart* directory to true.

4.2.1.15 Stream Transformation or Storage Not Working

Problem

Stream transformation or storage is not functioning correctly.

Solution

If stream transformation or storage is not functioning as expected, verify the consumer group list and lag for each topic as listed below:

1. To verify if stream storage is not storing data in the database, run the following script to verify the current number of records:

```
K8_NAMESPACE=...
MYSQL_USER=...
MYSQ_PASSWORD=...

kubectl -n ${K8_NAMESPACE} exec -it mysql-pod -- mysql -u ${MYSQL_USER} -p$
{MYSQ_PASSWORD} -e "SELECT COUNT(*) FROM cap4c_kafka_ingestor_db.<replace-
with-db>;"
```

2. Run the following script and verify the output to see if the topics are configured in Kafka:

```
K8_NAMESPACE=...
KAFKA_POD=...

kubectl -n ${K8_NAMESPACE} exec -it ${KAFKA_POD} -- kafka-consumer-
groups.sh  --list --bootstrap-server localhost:9092
```

3. Run the following script and verify the output to see if the consumer group is consuming data from the requested topic:

```
K8_NAMESPACE=...
KAFKA_POD=...
KAFKA_CONSUMER_GROUP=...

kubectl -n ${K8_NAMESPACE} exec -it ${KAFKA_POD} -- kafka-consumer-
groups.sh --bootstrap-server localhost:9092 --describe --group $
{KAFKA_CONSUMER_GROUP}
```

4. Run the following script and verify the output to see if data is produced in the expected format for the requested topic:

```
K8_NAMESPACE=...
KAFKA_POD=...
KAFKA_TOPIC=...

kubectl -n ${K8_NAMESPACE} exec -it ${KAFKA_POD} -- kafka-console-
consumer.sh --bootstrap-server localhost:9092 --topic ${KAFKA_TOPIC}  --
max-messages 1
```

5. Run the following script and verify the logs of the service which is not consuming the topic:

```
SERVICE_POD=...

kubectl logs ${SERVICE_POD} -f
```

4.2.1.16 Slice Load and Geographical Data

Slice load and Geographical data is used for simulation. The nwdaf_configuration service database tables are populated with slice and cell data for the scripts to run correctly. This information is used as test data. Verify the tables if you face any issues during installation.

4.2.2 Postinstallation

4.2.2.1 Helm Test Error Scenario

Following are the error scenarios that may be identified using Helm test.

1. Run the following command to get the Helm Test pod name:

   ```
   kubectl get pods -n <deployment-namespace>
   ```

2. When a Helm test is performed, a new Helm test pod is created. Check for the Helm Test pod that is in an error state.
3. Get the logs using the following command:

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

   Example:

   ```
   kubectl get <helm_test_pod> -n ocnwdaf
   ```

   For further assistance, collect the logs and contact MOS.

4.2.2.2 Uninstall Helm Chart

Perform the following steps to uninstall the Helm chart:

1. Run the following command to delete all jobs running in the cluster:

   ```
   kubectl delete jobs --all -n <namespace>
   ```

2. Run the following command to delete resources like pods, deployments, services, and so on running in the cluster:

   ```
   kubectl delete all --all -n <namespace>
   ```

3. Run the following Helm uninstall command:

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

4.2.2.3 Purge Kafka Topics for New Installation

If in a previous OCNWDAF installation, Kafka topics contained messages, the topics should be retained in the new installation but not the messages. Follow the procedure below to prevent purge of Kafka topics:

1. Connect to Kafka pod in your Kubernetes environment, run the command:

   ```
   kubectl -n <namespace> exec -it <podname> -- bash
   ```

2. Change directory, move to the directory that contains the binary files:

   ```
   cd kafka_2.13-3.1.0/bin/
   ```

3. Obtain the list of topics, run the command:

   ```
   kafka-topics.sh --list --bootstrap-server localhost:9092
   ```

4. Delete each topic (repeat this step for each topic):

   ```
   kafka-topics.sh --bootstrap-server localhost:9092 --delete --topic
   <topicname>
   ```

On completion of this procedure, the Kafka topics exist, but the messages do not exist.

> **Note**
>
> After every installation is recommended to purge the topics before uninstalling them.

4.3 Database Related Issues

This section describes the most common database related issues and their resolution steps. It is recommended to perform the resolution steps provided in this guide. If the issue still persists, then contact My Oracle Support.

4.3.1 Debugging MySQL DB Errors

If you are facing issues related to subscription creation, follow the procedure below to login to MySQL DB:

> **Note**
>
> Once the MySQL cluster is created, the *cndbtier_install container* generates the password and stores it in the *occne-mysqlndb-root-secret* secret.

1. Retrieve the MySQL root password from *occne-mysqlndb-root-secret* secret.
   Run the command:

   ```
   $ kubectl -n occne-cndbtier get secret occne-mysqlndb-root-secret -o
   jsonpath='{.data}'map[mysql_root_password:TmV4dEdlbkNuZQ==]
   ```

2. Decode the encoded output received as an output of the previous step to get the actual password:

   ```
   $ echo TmV4dEdlbkNuZQ== | base64 --decode
   NextGenCne
   ```

3. Login to MySQL pod, run the command:

   ```
   $ kubectl -n occnepsa exec -it ndbmysqld-0 -- bash
   ```

> **Note**
>
> Default container name is: `mysqlndbcluster`.
> Run the command `kubectl describe pod/ndbmysqld-0 -n occnepsa` to see all the containers in this pod.

4. Login using MySQL client as the root user, run the command:

   ```
   $ mysql -h 127.0.0.1 -uroot -p
   ```

5. Enter current root password for MySQL root user obtained from step 2.
6. To debug each microservice, perform the following steps:
   - For the **ocn-nwdaf-subscription** service, run the following SQL commands:

     ```
     use <dbName>;
     use nwdaf_subscription;
     select * from nwdaf_subscription;
     select * from amf_ue_event_subscription
     select * from smf_ue_event_subscription
     ```

   - For the **ocn-nrf-simulator** service, run the following SQL commands:

     ```
     use <dbName>;
     use nrf;
     select * from profile;
     ```

   - For the **ocn-smf-simulator** service, run the following SQL commands:

     ```
     use <dbName>;
     use nrf;
     select * from smf_event_subscription;
     ```

   - For the **ocn-amf-simulator** service, run the following SQL commands:

     ```
     use <dbName>;
     use nrf;
     select * from amf_event_subscription;
     ```

   - For the **ocn-nwdaf-data-collection** service, run the following SQL commands:

     ```
     use <dbName>;
     use nwdaf_data_collection;
     select * from amf_event_notification_report_list;
     select * from amf_ue_event_report;
     select * from cap4c_ue_notification;
     select * from slice_load_level_notification;
     select * from smf_event_notification_report_list;
     select * from smf_ue_event_report;
     select * from ue_mobility_notification;
     ```

- For the **ocn-nwdaf-configuration-service** service, run the following SQL commands:

```
use <dbName>;
use nwdaf_configuration_service;
select * from slice;
select * from tracking_are;
select * from slice_tracking_area;
select * from cell;
```

4.4 Apache Kafka Related Issues

To debug issues related to Apache Kafka pipelines (such as, unable to read messages from the pipeline or write messages to the pipeline) perform the following steps:

1. Get the Kafka pods, run the command:

   ```
   kubectl -n performance-ns get pods -o wide | grep "kafka"
   ```

2. Select any pod and access the pod using the command:

   ```
   kubectl -n performance-ns exec -it kafka-sts-0 -- bash
   ```

3. Move to the directory containing the binary files, run the command:

   ```
   cd kafka_2.13-3.1.0/bin/
   ```

4. Obtain the list of topics, run the command:

   ```
   kafka-topics.sh --list --bootstrap-server localhost:9092
   ```

5. For each topic, run the following command:

   ```
   kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic
       <topic-name>
   ```

4.5 CAP4C Related Issues

CAP4C comprises of the following services:

- cap4c-model-controller
- cap4c-model-executor
- cap4c-kafka-ingestor
- cap4c-api-gateway
- cap4c-configuration-manager
- cap4c-stream-analytics
- cap4c-stream-transformer
- nwdaf-cap4c-reporting-service
- nwdaf-cap4c-scheduler-service

To obtain more information on the service pods, follow the steps listed below:

1. Each of these services is deployed as pod in Kubernetes. To find the status of the pods in Kubernetes run the following command:

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

Sample output:

```
NAME                                                READY   STATUS
RESTARTS   AGE

cap4c-model-controller-deploy-779cbdcf8f-w2pfh       1/1     Running
0          4d8h

cap4c-model-executor-deploy-f9c96db54-ttnhd          1/1     Running
0          4d5h

cap4c-stream-analytics-deploy-744878569-5xr2w        1/1     Running
0          4d8h
```

2. To verify the pod information, print the detail of each pod to:

```
$ kubectl describe pod cap4c-model-controller-deploy-779cbdcf8f-w2pfh  -n
    <namespace>
```

Sample output:

```
Name:         cap4c-model-controller-deploy-779cbdcf8f-w2pfh

Namespace:    performance-ns

Priority:     0

Node:         sunstreaker-k8s-node-2/192.168.200.197

Start Time:   Fri, 26 Aug 2022 15:31:39 +0000

Labels:       app=cap4c-model-controller

              pod-template-hash=779cbdcf8f

Annotations:  cni.projectcalico.org/containerID:
480ca581a828184ccf6fabf7ec7cfb68920624f48d57148f6d93db4512bc5335

              cni.projectcalico.org/podIP: 10.233.76.134/32

              cni.projectcalico.org/podIPs: 10.233.76.134/32

              kubernetes.io/psp: restricted

              seccomp.security.alpha.kubernetes.io/pod: runtime/default

Status:       Running
```

3. List the service configuration for the pods, run the command:

```
$ kubectl get svc -n <namespace>
```

Sample output:

```
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP
         PORT(S)    AGE cap4c-executor    ClusterIP   10.233.5.218
<none>
         8888:32767/TCP 4d8h
```

4.6 Service Related Issues

This section describes the most common service related issues and their resolution steps. It is recommended to perform the resolution steps provided in this guide. If the issue still persists, then contact My Oracle Support.

4.6.1 Errors from Microservices

The OCNWDAF microservices are listed below:

- ocn-nwdaf-analytics
- ocn-nwdaf-mtlf-service
- ocn-nwdaf-subscription-service
- ocn-amf-simulator-service
- ocn-smf-simulator-service
- ocn-nrf-simulator-service
- ocn-oam-simulator-service
- mesa-simulator
- cap4c-model-controller
- cap4c-model-executor
- cap4c-stream-analytics
- cap4c-kafka-ingestor
- nwdaf-cap4c-reporting-service
- nwdaf-cap4c-kafka
- nwdaf-cap4c-scheduler-service
- nwdaf-cap4c-spring-cloud-config-server
- nwdaf-portal
- nwdaf-portal-service
- nwdaf-cap4c-redis
- nwdaf-cap4c-zookeeper
- nwdaf-cap4c-initial-setup-script
- ocats-nwdaf

- ocats-nwdaf-notify
- ocats-nwdaf-notify-nginx
- nf-test
- ocn-nwdaf-geo-redundacy-agent
- ocingress_gateway
- ocegress_gateway
- oc-config-server
- oc-app-info
- oc-perf-info
- nrf-client
- ocn-nwdaf-data-collection-controller
- cap4c-configuration-manager-service
- cap4c-stream-transformer
- nwdaf-cap4c-nginx
- cap4c-api-gateway

To debug microservice related errors, obtain the logs in the pods that are facing issues, run the following commands for each microservice:

1. To obtain the pod information, run the following command:

   ```
   kubectl get pods -n <nameSpace> -o wide
   ```

2. To obtain the log information for the pods, run the following command:

   ```
   kubectl logs <podName> -n <nameSpace>
   ```

Sample commands:

- `kubectl logs ocn-nwdaf-subscription-84f8b74cc7-d7lk9 -n performance-ns`
- `kubectl logs ocn-nwdaf-data-collection-57b948989c-xs7dq -n performance-ns`
- `kubectl logs ocn-amf-simulator-584ccb8fd4-pcdn6 -n performance-ns`

5
OCNWDAF Alerts

This chapter includes information about the following alerts:

- Application Level Alerts
- System Level Alerts

5.1 Application Level Alerts

This section lists the application level alerts.

OCN_NWDAF_ANALYTICS_NOT_RUNNING

Table 5-1 OCN_NWDAF_ANALYTICS_NOT_RUNNING

Field	Details
Description	The microservice is not available or not reachable.
Cause	Microservice ocn-nwdaf-analytics is down.

OCN_NWDAF_COMMUNICATION_NOT_RUNNING

Table 5-2 OCN_NWDAF_COMMUNICATION_NOT_RUNNING

Field	Details
Description	The microservice is not available or not reachable.
Cause	Microservice ocn-nwdaf-communication is down.

OCN_NWDAF_CONFIGURATION_SERVICE_NOT_RUNNING

Table 5-3 OCN_NWDAF_CONFIGURATION_SERVICE_NOT_RUNNING

Field	Details
Description	The microservice is not available or not reachable.
Cause	Microservice ocn-nwdaf-configuration-service is down.

OCN_NWDAF_DATA_COLLECTION_NOT_RUNNING

Table 5-4 OCN_NWDAF_DATA_COLLECTION_NOT_RUNNING

Field	Details
Description	The microservice is not available or not reachable.
Cause	Microservice ocn-nwdaf-data-collection is down.

OCN_NWDAF_GATEWAY_NOT_RUNNING

Table 5-5 OCN_NWDAF_GATEWAY_NOT_RUNNING

Field	Details
Description	The microservice is not available or not reachable.
Cause	Microservice ocn-nwdaf-gateway is down.

OCN_NWDAF_MTLF_NOT_RUNNING

Table 5-6 OCN_NWDAF_MTLF_NOT_RUNNING

Field	Details
Description	The microservice is not available or not reachable.
Cause	Microservice ocn-nwdaf-mtlf is down.

OCN_NWDAF_SUBSCRIPTION_NOT_RUNNING

Table 5-7 OCN_NWDAF_SUBSCRIPTION_NOT_RUNNING

Field	Details
Description	The microservice is not available or not reachable.
Cause	Microservice ocn-nwdaf-subscription is down.

HIGH_ABNORMAL_BEHAVIOUR_REQUEST_RATE

Table 5-8 HIGH_ABNORMAL_BEHAVIOUR_REQUEST_RATE

Field	Details
Description	The number of requests received per second is high.
Cause	Traffic is high, above 1000 requests per second.
URI Endpoint	`nnwdaf-analyticsinfo/v1/analytics?event-id=ABNORMAL_BEHAVIOUR`
Affected Functions	ABNORMAL_BEHAVIOUR

HIGH_UE_MOBILITY_REQUEST_RATE

Table 5-9 HIGH_UE_MOBILITY_REQUEST_RATE

Field	Details
Description	The number of requests received per second is high.
Cause	Traffic is high, above 1000 requests per second.
URI Endpoint	`nnwdaf-analyticsinfo/v1/analytics?event-id=UE_MOBILITY`
Affected Functions	UE_MOBILITY

HIGH_EVENT_SUBSCRIPTION_REQUEST_RATE

Table 5-10 HIGH_EVENT_SUBSCRIPTION_REQUEST_RATE

Field	Details
Description	The number of requests received per second is high.
Cause	Traffic is high, above 1000 requests per second.
URI Endpoint	`nnwdaf-eventssubscription/v1/subscriptions`
Affected Functions	UE_MOBILITY, SLICE_LOAD_LEVEL, ABNORMAL_BEHAVIOUR

HIGH_ABNORMAL_BEHAVIOUR_REQUEST_FAILURE_RATE

Table 5-11 HIGH_ABNORMAL_BEHAVIOUR_REQUEST_FAILURE_RATE

Field	Details
Description	The number of requests failing per second is high.
Cause	The request failing rate is more than the 70%.
URI Endpoint	`nnwdaf-analyticsinfo/v1/analytics?event-id=ABNORMAL_BEHAVIOUR`
Affected Functions	ABNORMAL_BEHAVIOUR

HIGH_UE_MOBILITY_REQUEST_FAILURE_RATE

Table 5-12 HIGH_ABNORMAL_BEHAVIOUR_REQUEST_FAILURE_RATE

Field	Details
Description	The number of requests failing per second is high.
Cause	The request failing rate is more than the 70%.
URI Endpoint	`nnwdaf-analyticsinfo/v1/analytics?event-id=UE_MOBILITY`
Affected Functions	UE_MOBILITY

HIGH_EVENT_SUBSCRIPTION_REQUEST_FAILURE_RATE

Table 5-13 HIGH_EVENT_SUBSCRIPTION_REQUEST_FAILURE_RATE

Field	Details
Description	The number of requests failing per second is high.
Cause	The request failing rate is more than the 70%.
URI Endpoint	`nnwdaf-eventssubscription/v1/subscriptions`
Affected Functions	UE_MOBILITY, SLICE_LOAD_LEVEL, ABNORMAL_BEHAVIOUR

5.2 System Level Alerts

This section lists the system level alerts.

OCN_NWDAF_ANALYTICS_HIGH_CPU_LOAD

Table 5-14 OCN_NWDAF_ANALYTICS_HIGH_CPU_LOAD

Field	Details
Description	CPU load is high at the pod where the microservice is running.
Affected Functions	All
Cause	CPU load is more than 80% of the allocated resources.

OCN_NWDAF_COMMUNICATION_HIGH_CPU_LOAD

Table 5-15 OCN_NWDAF_COMMUNICATION_HIGH_CPU_LOAD

Field	Details
Description	CPU load is high at the pod where the microservice is running.
Affected Functions	All
Cause	CPU load is more than 80% of the allocated resources.

OCN_NWDAF_CONFIGURATION_SERVICE_HIGH_CPU_LOAD

Table 5-16 OCN_NWDAF_CONFIGURATION_SERVICE_HIGH_CPU_LOAD

Field	Details
Description	CPU load is high at the pod where the microservice is running.
Affected Functions	All
Cause	CPU load is more than 80% of the allocated resources.

OCN_NWDAF_DATA_COLLECTION_HIGH_CPU_LOAD

Table 5-17 OCN_NWDAF_DATA_COLLECTION_HIGH_CPU_LOAD

Field	Details
Description	CPU load is high at the pod where the microservice is running.
Affected Functions	All
Cause	CPU load is more than 80% of the allocated resources.

OCN_NWDAF_GATEWAY_HIGH_CPU_LOAD

Table 5-18 OCN_NWDAF_GATEWAY_HIGH_CPU_LOAD

Field	Details
Description	CPU load is high at the pod where the microservice is running.
Affected Functions	All
Cause	CPU load is more than 80% of the allocated resources.

OCN_NWDAF_MTLF_HIGH_CPU_LOAD

Table 5-19 OCN_NWDAF_MTLF_HIGH_CPU_LOAD

Field	Details
Description	CPU load is high at the pod where the microservice is running.
Affected Functions	All
Cause	CPU load is more than 80% of the allocated resources.

OCN_NWDAF_SUBSCRIPTION_HIGH_CPU_LOAD

Table 5-20 OCN_NWDAF_SUBSCRIPTION_HIGH_CPU_LOAD

Field	Details
Description	CPU load is high at the pod where the microservice is running.
Affected Functions	All
Cause	CPU load is more than 80% of the allocated resources.

OCN_NWDAF_ANALYTICS_HIGH_JVM_HEAP_MEMORY_USAGE

Table 5-21 OCN_NWDAF_ANALYTICS_HIGH_JVM_HEAP_MEMORY_USAGE

Field	Details
Description	The average of the memory heap usage is high.
Affected Functions	All
Cause	The heap memory usage is more than the 80%.

OCN_NWDAF_COMMUNICATION_HIGH_JVM_HEAP_MEMORY_USAGE

Table 5-22 OCN_NWDAF_COMMUNICATION_HIGH_JVM_HEAP_MEMORY_USAGE

Field	Details
Description	The average of the memory heap usage is high.

Table 5-22 (Cont.) OCN_NWDAF_COMMUNICATION_HIGH_JVM_HEAP_MEMORY_USAGE

Field	Details
Affected Functions	All
Cause	The heap memory usage is more than the 80%.

OCN_NWDAF_CONFIGURATION_SERVICE_HIGH_JVM_HEAP_MEMORY_USAGE

Table 5-23 OCN_NWDAF_CONFIGURATION_SERVICE_HIGH_JVM_HEAP_MEMORY_USAGE

Field	Details
Description	The average of the memory heap usage is high.
Affected Functions	All
Cause	The heap memory usage is more than the 80%.

OCN_NWDAF_DATA_COLLECTION_HIGH_JVM_HEAP_MEMORY_USAGE

Table 5-24 OCN_NWDAF_DATA_COLLECTION_HIGH_JVM_HEAP_MEMORY_USAGE

Field	Details
Description	The average of the memory heap usage is high.
Affected Functions	All
Cause	The heap memory usage is more than the 80%.

OCN_NWDAF_GATEWAY_HIGH_JVM_HEAP_MEMORY_USAGE

Table 5-25 OCN_NWDAF_GATEWAY_HIGH_JVM_HEAP_MEMORY_USAGE

Field	Details
Description	The average of the memory heap usage is high.
Affected Functions	All
Cause	The heap memory usage is more than the 80%.

OCN_NWDAF_MTLF_HIGH_JVM_HEAP_MEMORY_USAGE

Table 5-26 OCN_NWDAF_MTLF_HIGH_JVM_HEAP_MEMORY_USAGE

Field	Details
Description	The average of the memory heap usage is high.
Affected Functions	All
Cause	The heap memory usage is more than the 80%.

OCN_NWDAF_SUBSCRIPTION_HIGH_JVM_HEAP_MEMORY_USAGE

Table 5-27 OCN_NWDAF_SUBSCRIPTION_HIGH_JVM_HEAP_MEMORY_USAGE

Field	Details
Description	The average of the memory heap usage is high.
Affected Functions	All
Cause	The heap memory usage is more than the 80%.