Oracle® Communications

Cloud Native Core, Network Slice Selection Function Troubleshooting Guide

Release 25.2.200
G40407-01
February 2026

Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide, Release 25.2.200

G40407-01

Contents

Preface

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

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

Conventions

The following text conventions are used in this document:

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

My Oracle Support

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

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

1. Select **2** for New Service Request.
2. Select **3** for Hardware, Networking and Solaris Operating System Support.
3. Select one of the following options:
   - 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**.

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 terms used in this document:

Table 1   Acronyms

Acronym	Description
AMF	Access and Mobility Management Function
HTTPS	Hypertext Transfer Protocol Secure
NF	Network Function
NRF	Oracle Communications Cloud Native Core, Network Repository Function
NSSAI	Network Slice Selection Assistance Information
NSSF	Oracle Communications Cloud Native Core, Network Slice Selection Function
Network Slice	A logical network that provides specific network capabilities and network characteristics.
PLMN	Public Land Mobile Network
Requested NSSAI	NSSAI provided by the UE to the serving PLMN during registration.
SMF	Session Management Function
S-NSSAI	Single Network Slice Selection Assistance Information
TAI	Tracking Area Identifier
UE	User Equipment

What's New in This Guide

This section lists the documentation updates for Release 25.2.2xx.

Release 25.2.200- G32783-01, February 2026

- Alerts have been cleaned up for a more focused monitoring and troubleshooting experience. Descriptions and OIDs of several existing alerts have been updated and some a outdated alerts have also been removed.
- Removed the following deprecated Alerts:
  - OcnssfTransactionErrorRateAbove0.1Percent
  - OcnssfIngressGatewayPodCongestionStateWarning
  - OcnssfIngressGatewayPodCongestionStateMajor
  - OcnssfIngressGatewayPodResourceStateWarning
  - OcnssfIngressGatewayPodResourceStateMajor
  - ocnssfPolicyNotFoundWarning
  - ocnssfPolicyNotFoundMajor
  - ocnssfPolicyNotFoundCritical
  - SubscriptionToNrfFailed

1 Introduction

This document provides information about troubleshooting Oracle Communications Cloud Native Core, Network Slice Selection Function (NSSF).

1.1 Overview

Network slices enable the operators to provide customized networks for:

- various functionalities such as mobility
- various performance requirements such as latency, availability, and reliability.

Network slices can differ for supported features and network function optimizations. Network Slices can have different S-NSSAIs with different Slice or Service Types.

The operator can deploy multiple instances of Network Slices that deliver the same features but for different groups of User Equipments (UEs). For example, when the operator delivers various committed services dedicated to a customer, the Network Slices can have different Single Network Slice Selection Assistance Information (S-NSSAIs) with the same Slice or Service Type but different Slice Differentiators.

NSSF fulfills the requirement for determining the individual network function of a slice.

This guide provides information about resolving problems you might experience while installing and configuring NSSF. This document also provides information about tools available to help you collect and analyze diagnostic data.

After identifying the issue, perform the steps to resolve the issue.

Note

The performance and capacity of the NSSF system may vary based on the call model, Feature or Interface configuration, and underlying CNE and hardware environment.

1.2 References

For more information, see the following documents:

- *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Network Slice Selection Function User Guide*
- *Oracle Communications Cloud Native Core, Network Slice Selection Function NIR Guide*
- *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*

2
Logs

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

2.1 Log Levels

This section provides information on log levels supported by Oracle Communications Cloud Native Core, Network Slice Selection Function (NSSF).

A log level helps in defining the severity level of a log message. Using this information, the logs can be filtered based on the system requirements. For example, 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.

Supported Log Levels

For NSSF, you can set the log level for a microservice to any of the following valid values:

- **TRACE**: This log level describes events showing step-by-step execution of your code. You may ignore this log during the standard operation. However, it may be useful during extended debugging sessions.
- **DEBUG**: This log level is useful while debugging an event and when more granular information is required.
- **INFO**: This is the standard log level indicating an event has occurred. For example, when the application enters a certain state.
- **WARN**: This log level indicates that some unexpected behavior has occurred in the application, but the application has not failed. Even with this unexpected behavior, the code can continue running.
- **ERROR**: This log level indicates an issue has occurred in the application that prevents one or more functionalities from working correctly.

Configuring Log Levels

Use the logLevel parameter to view and update the logging level configurations in the `ocnssf-custom-values.yaml` file. For example, see the following snippet from the `ocnssf-custom-values.yaml` file for Helm test hook related configuration.

```
test:
    config:
      logLevel: WARN
      timeout: 120
```

Examples with various Log Level Values

The following section provides examples for log messages containing different log level values:

For level `ERROR`:

```
{
  "instant": {
    "epochSecond": 1635408023,
    "nanoOfSecond": 739237045
  },
  "thread": "XNIO-1 task-1",
  "level": "ERROR",
  "loggerName":
"com.oracle.cgbu.cne.nssf.nsavailability.service.NsAvailabilityService",
  "message": "Unsupported PLMN/S received , supported plmn list: [Plmn
[mcc=100, mnc=101], Plmn [mcc=100, mnc=02], Plmn [mcc=101, mnc=101], Plmn
[mcc=102, mnc=102], Plmn [mcc=100, mnc=100], Plmn [mcc=200, mnc=200], Plmn
[mcc=100, mnc=102], Plmn [mcc=100, mnc=001], Plmn [mcc=103, mnc=103]]",
  "endOfBatch": false,
  "loggerFqcn": "org.apache.logging.log4j.spi.AbstractLogger",
  "contextMap": {
    "ocLogId": "1635408023716_15768_ocnssf-ingress-gateway-6d554c596-q86hp"
  },
  "threadId": 1446,
  "threadPriority": 5,
  "ts": "2021-10-28 08:00:23.739+0000",
  "ocLogId": "1635408023716_15768_ocnssf-ingress-gateway-6d554c596-q86hp",
  "pod": "ocnssf-nsavailability-5f858b7745-mk2pv",
  "processId": "1",
  "vendor": "Oracle",
  "application": "ocnssf",
  "engVersion": "1.0.0",
  "mktgVersion": "1.0.0.0.0",
  "microservice": "nsavailability",
  "namespace": "ocnssf",
  "node_name": "remote-setup-kamal"
}
```

For level `INFO`:

```
{
  "instant": {
    "epochSecond": 1635408080,
    "nanoOfSecond": 653030110
  },
  "thread": "scheduling-1",
  "level": "INFO",
  "loggerName": "com.oracle.common.scheduler.ReloadConfig",
  "message": "All configurations intact, no updation in the configuration",
  "endOfBatch": false,
  "loggerFqcn": "org.apache.logging.log4j.spi.AbstractLogger",
  "contextMap": {},
  "threadId": 28,
  "threadPriority": 5,
  "ts": "2021-10-28 08:01:20.653+0000",
  "ocLogId": "${ctx:ocLogId}",
  "pod": "ocnssf-nsavailability-5f858b7745-mk2pv",
  "processId": "1",
  "vendor": "Oracle",
  "application": "ocnssf",
  "engVersion": "1.0.0",
  "mktgVersion": "1.0.0.0.0",
  "microservice": "nsavailability",
  "namespace": "ocnssf",
  "node_name": "remote-setup-kamal"
}
```

2.2 Collecting Logs

This section describes how to collect logs from pods and containers. The steps are as follows:

Collect the NSSF Logs to check the error scenarios

Problem: The error scenarios are checked by collecting the NSSF logs.

Solution:

Perform the following steps to get the logs from nssf:

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

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

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

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

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

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

   Example:

   ```
   kubectl logs ocnssf-nsselection-57cff5665c-skk4l -n ocnssf >
   ocnssf_logs1.log
   ```

4. (Optional) Run the following command for the log stream with file redirection starting with last 100 lines of log:

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

For more information on kubectl commands, see Kubernetes website.

2.3 Understanding Logs

This section provides information on how to read logs for various services of NSSF in Kibana.

NSSF Log Attributes

The log message format is same for all the NSSF services. The following table describes key attributes of a log message:

Table 2-1 NSSF Log Attributes

Attribute Name	Description	Details
`instant`	Timestamp details for when the log event occurred. **Note:** Contains both epoch seconds and nanoseconds for precision.	**Data Type:** Object
`thread`	Name of the thread executing the log. **Note:** Useful in multi-threaded applications.	**Data Type:** String
`level`	Log severity level. **Note:** Common levels: DEBUG, INFO, WARN, ERROR.	**Data Type:** String
`loggerName`	Fully qualified logger class name. **Note:** Identifies the logging component.	**Data Type:** String
`message`	Log message details. **Note:** Describes the issue being logged.	**Data Type:** String
`endOfBatch`	Indicates if this is the last log in batch processing. **Note:** Often used in batch logging frameworks.	**Data Type:** Boolean
`loggerFqcn`	Fully qualified class name of the logger implementation. **Note:** Helps identify the logging backend.	**Data Type:** String
`contextMap`	Contextual key-value pairs (if any). **Note:** Stores contextual metadata.	**Data Type:** Object
`threadId`	Numeric ID of the thread executing the log. **Note:** Helps trace logs in multi-threaded processes.	**Data Type:** Integer
`threadPriority`	Priority level of the thread. **Note:** Helps in debugging thread behavior.	**Data Type:** Integer
`ts`	Timestamp when log entry was created. **Note:** Provides human-readable log time.	**Data Type:** String (ISO 8601)
`ocLogId`	Unique identifier for log entry. **Note:** Used for tracing logs.	**Data Type:** String
`pod`	Name of the Kubernetes pod. **Note:** Helps track logs in containerized environments.	**Data Type:** String
`processId`	Process ID of the application instance. **Note:** Identifies application instance.	**Data Type:** String
`vendor`	Vendor information. **Note:** Identifies software vendor.	**Data Type:** String
`application`	Application name. **Note:** Identifies application logging the message.	**Data Type:** String
`engVersion`	Engineering version of the application. **Note:** Engineering release version.	**Data Type:** String
`mktgVersion`	Marketing version of the application. **Note:** Marketing release version.	**Data Type:** String
`microservice`	Name of the microservice component. **Note:** Helps identify the specific service.	**Data Type:** String
`namespace`	Kubernetes namespace. **Note:** Groups services within Kubernetes.	**Data Type:** String

Table 2-1 (Cont.) NSSF Log Attributes

Attribute Name	Description	Details
`node_name`	Name of the Kubernetes node where the pod is running. **Note:** Helps in tracking node-level deployment.	**Data Type:** String

Sample log statement of nsconfig:

```
{
    "instant": {
        "epochSecond": 1645187968,
        "nanoOfSecond": 864222104
    },
    "thread": "main",
    "level": "INFO",
    "loggerName": "com.oracle.cgbu.cne.nssf.nsconfig.Application",
    "message": "Started Application in 21.489 seconds (JVM running for
23.688)",
    "endOfBatch": false,
    "loggerFqcn": "org.apache.commons.logging.LogAdapter$Log4jLog",
    "contextMap": {},
    "threadId": 1,
    "threadPriority": 5,
    "ts": "2022-02-18 12:39:28.864+0000",
    "ocLogId": "${ctx:ocLogId}",
    "pod": "ocnssf-nsconfig-85bd697b74-kxrfv",
    "processId": "1",
    "vendor": "Oracle",
    "application": "ocnssf",
    "engVersion": "22.1.1",
    "mktgVersion": "22.1.1.0.0",
    "microservice": "nsconfig",
    "namespace": "nssf",
    "node_name": "bumblebee-k8s-node-15"
}
```

appinfo Log Attributes

The log message format is same for all the NSSF services. The following table describes key attributes of a log message:

Table 2-2 appinfo Log Attributes

Attribute Name	Description	Details
`name`	Logger name that identifies the source of the log. **Note:** Helps categorize logs from different components.	**Data Type:** String
`message`	The main log message providing relevant details about the event. **Note:** Typically contains request details in access logs.	**Data Type:** String

Table 2-2 (Cont.) appinfo Log Attributes

Attribute Name	Description	Details
`level`	Severity level of the log. **Note:** Common levels include DEBUG, INFO, WARN, ERROR.	**Data Type:** String
`filename`	Name of the file where the log was generated. **Note:** Helps trace the log source in the codebase.	**Data Type:** String
`lineno`	Line number in the source file where the log was generated. **Note:** Useful for debugging.	**Data Type:** Integer
`module`	Python module name associated with the log. **Note:** Helps identify the exact module in multi-module applications.	**Data Type:** String
`func`	Name of the function where the log was generated. **Note:** Useful for debugging and tracing log sources.	**Data Type:** String
`thread`	Name of the thread executing the log. **Note:** Important for multi-threaded applications.	**Data Type:** String
`messageTimestamp`	Timestamp when the log was created, following the format yyyy-MM-dd'T'HH:mm:ss.SSSZ. **Note:** Provides an accurate event timeline.	**Data Type:** String

Sample log statement of appinfo:

```
{
  "name": "gunicorn.access",
  "message": "::ffff:10.244.1.106 - - [21/May/2022:13:54:29 +0000] \"GET /
status/category/sepp HTTP/1.1\" 200 7 \"-\" \"okhttp/3.14.9\"",
  "level": "INFO",
  "filename": "glogging.py",
  "lineno": 349,
  "module": "glogging",
  "func": "access",
  "thread": "MainThread",
  "messageTimestamp": "2022-05-21T13:54:29.385+0000"
}
```

perf-info Log Attributes

Table 2-3 perf-info Log Attributes

Attribute Name	Description <br> Note	Details
`name`	Logger name or category. **Note:** Identifies the logging component.	**Data Type:** String
`message`	Log message details. **Note:** Describes the issue being logged.	**Data Type:** String
`level`	Log severity level. **Note:** Common levels: DEBUG, INFO, WARN, ERROR.	**Data Type:** String

Table 2-3 (Cont.) perf-info Log Attributes

Attribute Name	Description <br> Note	Details
`filename`	Name of the source file where the log was generated. **Note:** Helps trace log source.	**Data Type:** String
`lineno`	Line number in the source file. **Note:** Useful for debugging.	**Data Type:** Integer
`module`	Python module where the log originated. **Note:** Indicates module name.	**Data Type:** String
`func`	Function name that generated the log. **Note:** Helps track function-level logging.	**Data Type:** String
`thread`	Name of the thread handling the request. **Note:** Useful in multi-threaded applications.	**Data Type:** String
`messageTimestamp`	Timestamp when the log entry was created. **Note:** Provides human-readable log time.	**Data Type:** String (ISO 8601)

Sample log statement of perf-info:

```
{
  "name": "stat_helper",
  "message": "Failed to reach prometheus",
  "level": "ERROR",
  "filename": "stat_helper.py",
  "lineno": 106,
  "module": "stat_helper",
  "func": "get_db_param",
  "thread": "MainThread",
  "messageTimestamp": "2022-05-21T13:57:39.639+0000"
}
```

config-server Log Attributes

Table 2-4 config-server Log Attributes

Attribute Name	Description <br> Note	Details
`instant`	Timestamp details for when the log event occurred. **Note:** Contains both epoch seconds and nanoseconds for precision.	**Data Type:** Object
`thread`	Name of the thread executing the log. **Note:** Useful in multi-threaded applications.	**Data Type:** String
`level`	Log severity level. **Note:** Common levels: DEBUG, INFO, WARN, ERROR.	**Data Type:** String
`loggerName`	Fully qualified logger class name. **Note:** Identifies the logging component.	**Data Type:** String
`message`	Log message details. **Note:** Describes the issue being logged.	**Data Type:** String

Table 2-4 (Cont.) config-server Log Attributes

Attribute Name	Description <br> Note	Details
`endOfBatch`	Indicates if this is the last log in a batch processing. **Note:** Often used in batch logging frameworks.	**Data Type:** Boolean
`loggerFqcn`	Fully qualified class name of the logger implementation. **Note:** Helps identify the logging backend.	**Data Type:** String
`threadId`	Numeric ID of the thread executing the log. **Note:** Helps trace logs in multi-threaded processes.	**Data Type:** Integer
`threadPriority`	Priority level of the thread. **Note:** Helps in debugging thread behavior.	**Data Type:** Integer
`messageTimestamp`	Timestamp when the log entry was created. **Note:** Provides human-readable log time.	**Data Type:** String (ISO 8601)

Sample log statement of config-server:

```
{
    "instant": {
        "epochSecond": 1653140996,
        "nanoOfSecond": 895472496
    },
    "thread": "main",
    "level": "INFO",
    "loggerName": "ocnssf.cne.common.metrics.cgroup.CgroupMetricsHelper",
    "message": "Creating cgroup metric finder",
    "endOfBatch": false,
    "loggerFqcn": "org.apache.logging.slf4j.Log4jLogger",
    "threadId": 1,
    "threadPriority": 5,
    "messageTimestamp": "2022-05-21T13:49:56.895+0000"
}
```

nrf-client-nfdiscovery Log Attributes

Table 2-5 nrf-client-nfdiscovery Log Attributes

Attribute Name	Description <br> Note	Details
`instant`	Timestamp details for when the log event occurred. **Note:** Contains both epoch seconds and nanoseconds for precision.	**Data Type:** Object
`thread`	Name of the thread executing the log. **Note:** Useful in multi-threaded applications.	**Data Type:** String
`level`	Log severity level. **Note:** Common levels: DEBUG, INFO, WARN, ERROR.	**Data Type:** String
`loggerName`	Fully qualified logger class name. **Note:** Identifies the logging component.	**Data Type:** String

Table 2-5 (Cont.) nrf-client-nfdiscovery Log Attributes

Attribute Name	Description Note	Details
`message`	Log message details. **Note:** Describes the issue being logged.	**Data Type:** String
`endOfBatch`	Indicates if this is the last log in a batch processing. **Note:** Often used in batch logging frameworks.	**Data Type:** Boolean
`loggerFqcn`	Fully qualified class name of the logger implementation. **Note:** Helps identify the logging backend.	**Data Type:** String
`threadId`	Numeric ID of the thread executing the log. **Note:** Helps trace logs in multi-threaded processes.	**Data Type:** Integer
`threadPriority`	Priority level of the thread. **Note:** Helps in debugging thread behavior.	**Data Type:** Integer
`source`	Details of the Java class, method, and file where the log was generated. **Note:** Contains exact file, method, and line number for debugging.	**Data Type:** Object
`messageTimestamp`	Timestamp when the log entry was created. **Note:** Provides human-readable log time.	**Data Type:** String (ISO 8601)

Sample log statement of nrf-client-nfdiscovery:

```
{
    "instant": {
        "epochSecond": 1653141021,
        "nanoOfSecond": 819399951
    },
    "thread": "main",
    "level": "WARN",
    "loggerName": "com.oracle.cgbu.cnc.nrf.util.NrfClientProperties",
    "message": "getHttpsProxyPort():Invalid httpsProxyPort",
    "endOfBatch": false,
    "loggerFqcn": "org.apache.logging.slf4j.Log4jLogger",
    "threadId": 1,
    "threadPriority": 5,
    "source": {
        "class": "com.oracle.cgbu.cnc.nrf.util.NrfClientProperties",
        "method": "getHttpsProxyPort",
        "file": "NrfClientProperties.java",
        "line": 260
    },
    "messageTimestamp": "2022-05-21T13:50:21.819+0000"
}
```

nrf-client-nfmanagement Log Attributes

Table 2-6 nrf-client-nfmanagement Log Attributes

Attribute Name	Description	Details
`instant`	Timestamp details for when the log event occurred. **Note:** Contains both epoch seconds and nanoseconds for precision.	**Data Type:** Object
`thread`	Name of the thread executing the log. **Note:** Useful in multi-threaded applications.	**Data Type:** String
`level`	Log severity level. **Note:** Common levels: DEBUG, INFO, WARN, ERROR.	**Data Type:** String
`loggerName`	Fully qualified logger class name. **Note:** Identifies the logging component.	**Data Type:** String
`message`	Log message details. **Note:** Describes the issue being logged.	**Data Type:** String
`endOfBatch`	Indicates if this is the last log in batch processing. **Note:** Often used in batch logging frameworks.	**Data Type:** Boolean
`loggerFqcn`	Fully qualified class name of the logger implementation. **Note:** Helps identify the logging backend.	**Data Type:** String
`threadId`	Numeric ID of the thread executing the log. **Note:** Helps trace logs in multi-threaded processes.	**Data Type:** Integer
`threadPriority`	Priority level of the thread. **Note:** Helps in debugging thread behavior.	**Data Type:** Integer
`source`	Details of the Java class, method, and file where the log was generated. **Note:** Contains exact file, method, and line number for debugging.	**Data Type:** Object
`messageTimestamp`	Timestamp when the log entry was created. **Note:** Provides human-readable log time.	**Data Type:** String (ISO 8601)

Sample log statement of nrf-client-nfmanagement:

```
{
    "instant": {
        "epochSecond": 1653141225,
        "nanoOfSecond": 57167090
    },
    "thread": "taskScheduler-2",
    "level": "WARN",
    "loggerName": "com.oracle.cgbu.cnc.nrf.NRFManagement",
    "message": "NfServices is not present inNfProfile.",
    "endOfBatch": false,
    "loggerFqcn": "org.apache.logging.slf4j.Log4jLogger",
    "threadId": 27,
    "threadPriority": 5,
    "source": {
        "class": "com.oracle.cgbu.cnc.nrf.NRFManagement",
        "method": "setPerformance",
        "file": "NRFManagement.java",
        "line": 1758
    },
    "messageTimestamp": "2022-05-21T13:53:45.057+0000"
}
```

Ingress Gateway Log Attributes

Table 2-7 Ingress Gateway Log Attributes

Attribute Name	Description	Details
`thread`	Logging Thread Name.	**Data Type:** String
`level`	Log Level of the log printed.	**Data Type:** String
`loggerName`	Class or Module which printed the log.	**Data Type:** String
`message`	Message related to the log providing brief details. **Note:** Indicates that the method `PreGatewayFilter` is being exited.	**Data Type:** String
`endOfBatch`	Log4j2 Internal default flag. **Note:** Default from Log4j2: `false`.	**Data Type:** Boolean
`loggerFqcn`	Log4j2 Internal - Fully Qualified class name of the logger module.	**Data Type:** String
`instant`	Epoch timestamp. **Note:** It consists of two values - `epochSecond` and `nanoOfSecond` for high precision.	**Data Type:** Object
`contextMap`	Contents of Log4j ThreadContext map.	**Data Type:** Object
`threadId`	Thread ID generated internally by Log4j2.	**Data Type:** Integer
`threadPriority`	Thread Priority set internally by Log4j2.	**Data Type:** Integer
`messageTimestamp`	Timestamp of log from application container. **Note:** Format: `yyyy-MM-dd'T'HH:mm:ss.SSSZ`.	**Data Type:** String
`ocLogId`	End-to-End Log Identifier across the NSSF microservices. **Note:** Helps to correlate logs across microservices in NSSF application.	**Data Type:** String
`pod`	Pod Name.	**Data Type:** String
`processId`	Process ID internally assigned.	**Data Type:** String
`instanceType`	Instance type.	**Data Type:** String
`ingressTxId`	Transaction ID that is added to Log4j ThreadContext map and is unique to every transaction.	**Data Type:** String

Sample log statement of Ingress Gateway:

```
{
    "thread": "ingress-h2c-epoll-3",
    "level": "DEBUG",
    "loggerName": "ocnssf.cne.gateway.filters.PreGatewayFilter",
    "message": "Exiting PreGatewayFilter",
    "endOfBatch": false,
    "loggerFqcn": "org.apache.logging.log4j.spi.AbstractLogger",
    "instant": {
        "epochSecond": 1604650229,
        "nanoOfSecond": 4993000
    },
    "contextMap": {
        "hostname": "ocnssf-ingressgateway-69f6544b8d-cdbgx",
        "ingressTxId": "ingress-tx-1087436877",
        "ocLogId": "1604650229002_72_ocnssf-ingressgateway-69f6544b8d-cdbgx"
    },
    "threadId": 72,
    "threadPriority": 5,
    "messageTimestamp": "2020-11-06 08:10:29.004",
    "ocLogId": "1604650229002_72_ocnssf-ingressgateway-69f6544b8d-cdbgx",
    "pod": "ocnssf-ingressgateway-69f6544b8d-cdbgx",
    "processId": "1",
    "instanceType": "prod",
    "ingressTxId": "ingress-tx-1087436877"
}
```

Egress Gateway Log Attributes

Table 2-8 Egress Gateway Log Attributes

Attribute Name	Description	Details
`thread`	Logging Thread Name.	**Data Type:** String
`level`	Log Level of the log printed.	**Data Type:** String
`loggerName`	Class or Module which printed the log.	**Data Type:** String
`message`	Message related to the log providing brief details. **Note:** Contains property name and value information.	**Data Type:** String
`endOfBatch`	Log4j2 Internal default flag. **Note:** Default from Log4j2: `false`.	**Data Type:** Boolean
`loggerFqcn`	Log4j2 Internal - Fully Qualified class name of the logger module.	**Data Type:** String
`instant`	Epoch timestamp. **Note:** It consists of two values - `epochSecond` and `nanoOfSecond` for high precision.	**Data Type:** Object
`contextMap`	Elements in Log4j ThreadContext map.	**Data Type:** Object
`threadId`	Thread ID generated internally by Log4j2.	**Data Type:** Integer
`threadPriority`	Thread Priority set internally by Log4j2.	**Data Type:** Integer
`messageTimestamp`	Timestamp of log from application container. **Note:** Format: `yyyy-MM-dd'T'HH:mm:ss.SSSZ`.	**Data Type:** String
`ocLogId`	End-to-End Log Identifier across the NSSF microservices. **Note:** Helps to correlate logs across microservices in NSSF application.	**Data Type:** String
`pod`	Name of the egress pod.	**Data Type:** String
`processId`	Process ID internally assigned.	**Data Type:** String
`instanceType`	Instance type.	**Data Type:** String

Table 2-8 (Cont.) Egress Gateway Log Attributes

Attribute Name	Description	Details
egressTxId	Transaction ID that is added to Log4j ThreadContext map and is unique to every transaction.	**Data Type:** String

Sample log statement of Egress Gateway:

```
{
    "thread": "main",
    "level": "DEBUG",
    "loggerName": "ocnssf.cne.gateway.config.ScpDynamicBeanConfiguration",
    "message": "Property name: server.port and value: 8080",
    "endOfBatch": false,
    "loggerFqcn": "org.apache.logging.log4j.spi.AbstractLogger",
    "instant": {
        "epochSecond": 1604564777,
        "nanoOfSecond": 135977000
    },
    "contextMap": {},
    "threadId": 1,
    "threadPriority": 5,
    "messageTimestamp": "2020-11-05 08:26:17.135",
    "ocLogId": "1604650229002_72_ocnssf-ingressgateway-69f6544b8d-cdbgx",
    "pod": "ocnssf-egressgateway-69f6544b8d-cdbgx",
    "processId": "1",
    "instanceType": "prod",
    "egressTxId": "egress-tx-1087436877"
}
```

3

Debug Tool

Overview

The Debug Tool provides third-party troubleshooting tools for debugging the runtime issues in a lab environment. Following are the available tools:

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

Prerequisites

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**
   Perform the following configurations in the Bastion Host. You need admin privileges to perform these configurations.
   a. When NSSF 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":["ocnssf"]} }]'
```

Sample output:

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

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": "ocnssf" }]'
   ```

   Sample output:

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

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

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 NSSF is installed on CNE version prior to 23.2.0
PodSecurityPolicy (PSP) Creation

i. Log in to the Bastion Host.

ii. Run the following command from the Bastion Host to create a new PSP. The parameters `readOnlyRootFileSystem`, `allowPrivilegeEscalation`, `allowedCapabilities` are required by debug container.

Note

Other parameters are mandatory for PSP creation and can be customized as per the CNE environment. 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 attach the service account for your NF namespace with the role created for the tool PSP:

```
$ kubectl apply -f - <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: debug-tool-rolebinding
  namespace: ocnssf
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**
   Perform the following updates in `ocnssf_custom_values_25.2.200.yaml` file.
   a. Log in to the NF server.
   b. Open the `ocnssf_custom_values_25.2.200.yaml` file:

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

   Example:

   ```
   $vim ocnssf_custom_values_25.2.200.yaml
   ```

   c. Under global configuration, add the following:

   ```
   # Allowed Values: DISABLED, ENABLED
     podSecurityPolicy: "DISABLED"
     extraContainers: "DISABLED"
     debugToolContainerMemoryLimit: 4Gi
     extraContainersImageDetails:
       image: ocdebugtool/ocdebug-tools
       tag: debug_container_tag
       imagePullPolicy: Always
     extraContainersVolumesTpl: |
       - name: debug-tools-dir
         emptyDir:
           medium: Memory
           sizeLimit: {{ .Values.global.debugToolContainerMemoryLimit |
   quote }}
     extraContainersTpl: |-
   ```

```
      - command:
          - /bin/sleep
          - infinity
        name: tools
        resources:
          requests:
            ephemeral-storage: "512Mi"
            cpu: "0.5"
            memory: {{ .Values.global.debugToolContainerMemoryLimit |
quote }}
          limits:
            ephemeral-storage: "512Mi"
            cpu: "1"
            memory: {{ .Values.global.debugToolContainerMemoryLimit |
quote }}
        securityContext:
          allowPrivilegeEscalation: true
          capabilities:
            drop:
            - ALL
            add:
            - NET_RAW
            - NET_ADMIN
          runAsUser: 1012
        volumeMounts:
        - mountPath: /tmp/tools
          name: debug-tools-dir
```

Note

- Debug Tool Container comes up with the default user ID - 7000. To override the default value, use "runAsUser" field, otherwise the field can be skipped.

  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 `ocnssf_custom_values_25.2.200.yaml` file with the following:

  ```
  name: {{ printf "%s-tools-%s" (include "getprefix" .)
  (include "getsuffix" .) | trunc 63 | trimPrefix "-" |
  trimSuffix "-"  }}
  ```

  This ensures 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, use `extraContainers` flag to enable or disable injection of 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, use `extraContainers` flag to determine whether to use the extra container configuration from the global level or enable or disable injecting extra containers for the specific service.

Run the Debug Tool

To run Debug Tool, perform the following steps:

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

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

   Example:

   ```
   $ kubectl get pods -n ocnssf
   ```

   Sample Output:

   ```
   NAME                                                     READY   STATUS
   RESTARTS   AGE
   ocnssf-appinfo-659745778c-58jqc                           1/1     Running
   6       2d
   ocnssf-egress-gateway-5d5cd8bb9-fln4w                     1/1     Running
   7       2d
   ocnssf-ingress-gateway-98b65b4d9-cs8jb                    1/1     Running
   7       2d
   ocnssf-nsavailability-8699784c8d-8wwd5                    1/1     Running
   9       2d
   ocnssf-nsconfig-5dd646cf76-66k9g                          1/1     Running
   9       2d
   ocnssf-nsselection-67b7bd9bcf-2nv56                       1/1     Running
   7       2d
   ocnssf-nssubscription-5c96d8b9cc-fzwnt                    1/1     Running
   11      2d
   ocnssf-ocnssf-nrf-client-nfdiscovery-7646cfc547-5hpsd 1/1     Running
   5       2d
   ocnssf-ocnssf-nrf-client-nfdiscovery-7646cfc547-lc96j 1/1     Running
   5       2d
   ocnssf-ocnssf-nrf-client-nfmanagement-8f7bfb98-s4b4b  1/1     Running
   5       2d
   ocnssf-ocpm-config-5c475d4646-lxvzq                       1/1     Running
   5       2d
   ```

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

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

Example:

```
$ kubectl exec -it ocnssf-nsselection-67b7bd9bcf-2nv56 -c tools -n ocnssf
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>
   ```

   Example:

   ```
   $ kubectl cp -c tools ocnssf-nsselection-67b7bd9bcf-2nv56:/tmp/
   capture.pcap -n ocnssf /tmp/
   ```

Tools Tested in Debug Container

Following is the list of debug tools that are tested:

tcpdump
The details of tcpdump debug tool are as follows:

Table 3-1 tcpdump

Options Tested	Description	Details
`-D`	Print the list of the network interfaces available on the system and on which *tcpdump* can capture packets.	**Output:** `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] **Capabilities:** NET_ADMIN, NET_RAW

Table 3-1 (Cont.) tcpdump

Options Tested	Description	Details
`-i`	Listen on *interface*.	**Output:** `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) **Capabilities:** NET_ADMIN, NET_RAW
`-w`	Write the raw packets to file rather than parsing and printing them out.	**Output:** `tcpdump -w capture.pcap -i eth0` **Capabilities:** NET_ADMIN, NET_RAW
`-r`	Read packets from *file* (which was created with the **-w** option).	**Output:** `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 **Capabilities:** NET_ADMIN, NET_RAW

ip
The details of ip debug tool are as follows:

Table 3-2 ip

Options Tested	Description	Details
`addr show`	Look at protocol addresses.	**Output:** `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 forever 2. tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group defaultlink/ipip 0.0.0.0 brd 0.0.0.0 3. 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.	**Output:** `ip route show` default via 169.254.1.1 dev eth0 169.254.1.1 dev eth0 scope link
`addrlabel list`	List address labels	**Output:** `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 details of netstat debug tool are as follows:

Table 3-3 netstat

Options Tested	Description	Details

`-a`	Show both listening and non-listening sockets. For TCP, this means established connections.	**Output:**
		`netstat -a`
		
		`Active Internet connections (servers and established)`
		`Proto Recv-Q Send-Q Local Address Foreign Address State`
		`tcp 0 0 0.0.0.0:tproxy 0.0.0.0:* LISTEN`
		`tcp 0 0 0.0.0.0:websm 0.0.0.0:* LISTEN`
		`tcp 0 0 ocnssf-ingress:websm 10-178-254-194.ku:47292 TIME_WAIT`
		`tcp 0 0 ocnssf-ingress:46519 kubernetes.defaul:https ESTABLISHED`
		`tcp 0 0 ocnssf-ingress:websm 10-178-254-194.ku:47240 TIME_WAIT`
		`tcp 0 0 ocnssf-ingress:websm 10-178-254-194.ku:47347 TIME_WAIT`
		`udp 0 0 localhost:59351 localhost:ambit-lm ESTABLISHED`
		`Active UNIX domain sockets (servers and established)`
		`Proto RefCnt Flags Type State I-Node Path`
		`unix 2 [ ] STREAM CONNECTED 576064861`

`-l`	Show only listening sockets.	**Output:** `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

Table 3-3 (Cont.) netstat

Options Tested	Description	Details

`-s`	Display summary statistics for each protocol.	**Output:**
		`netstat -s`
		
		`Ip:`
		`4070 total packets received`
		`0 forwarded`
		`0 incoming packets discarded`
		`4070 incoming packets delivered`
		`4315 requests sent out`
		`Icmp:`
		`0 ICMP messages received`
		`0 input ICMP message failed.`
		`ICMP input histogram:`
		`2 ICMP messages sent`
		`0 ICMP messages failed`
		`ICMP output histogram:`
		`destination unreachable: 2`

`-i`	Display a table of all network interfaces.	**Output:** `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 details of curl debug tool are as follows:

Table 3-4 curl

Options Tested	Description	Details
`-o`	Write output to <file> instead of stdout.	**Output:** `curl -o file.txt http://abc.com/file.txt`
`-x`	Use the specified HTTP proxy.	**Output:** `curl -x proxy.com:8080 -o http://abc.com/file.txt`
--http2	HTTP/2	--

ping
The details of ping debug tool are as follows:

Table 3-5 ping

Options Tested	Description	Details

<ip>	Run a ping test to see whether the target host is reachable or not.	**Output:**
		`ping 10.178.254.194`
		
		`PING 10.178.254.194 (10.178.254.194) 56(84) bytes of data.`
		`64 bytes from 10.178.254.194: icmp_seq=1 ttl=64 time=0.044 ms`
		`64 bytes from 10.178.254.194: icmp_seq=2 ttl=64 time=0.048 ms`
		`64 bytes from 10.178.254.194: icmp_seq=3 ttl=64 time=0.047 ms`
		`64 bytes from 10.178.254.194: icmp_seq=4 ttl=64 time=0.057 ms`
		`64 bytes from 10.178.254.194: icmp_seq=5 ttl=64 time=0.046 ms`
		
		Capabilities:
		NET_ADMIN, NET_RAW

`-c`	Stop after sending '*c*' number of ECHO_REQUEST packets.	**Output:**
		`ping -c 5 10.178.254.194`
		
		`PING 10.178.254.194 (10.178.254.194) 56(84) bytes of data.`
		`64 bytes from 10.178.254.194: icmp_seq=1 ttl=64 time=0.051 ms`
		`64 bytes from 10.178.254.194: icmp_seq=2 ttl=64 time=0.036 ms`
		`64 bytes from 10.178.254.194: icmp_seq=3 ttl=64 time=0.037 ms`
		`64 bytes from 10.178.254.194: icmp_seq=4 ttl=64 time=0.037 ms`
		`64 bytes from 10.178.254.194: icmp_seq=5 ttl=64 time=0.049 ms`
		
		`--- 10.178.254.194 ping statistics ---`
		`5 packets transmitted, 5 received, 0% packet loss, time 3999ms`
		`rtt min/avg/max/mdev = 0.036/0.042/0.051/0.006 ms`
		
		Capabilities:
		NET_ADMIN, NET_RAW

Table 3-5 (Cont.) ping

Options Tested	Description	Details

`-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.	**Output:**
		`ping -f -i 2 10.178.254.194`
		
		`PING 10.178.254.194 (10.178.254.194) 56(84) bytes of data.`
		`--- 10.178.254.194 ping statistics --- 251 packets transmitted, 251 received, 0% packet loss, time 75ms`
		`rtt min/avg/max/mdev = 0.029/0.048/0.210/0.019 ms, ipg/ewma 300.001/0.044 ms`
		
		Capabilities:
		NET_ADMIN, NET_RAW

nmap
The details of nmap debug tool are as follows:

Table 3-6 nmap

Options Tested	Description	Details

<ip>	Scan for Live hosts, Operating systems, packet filters, and open ports running on remote hosts.	**Output:**
		`nmap 10.178.254.194`
		
		`Starting Nmap 6.40 ( http://nmap.org ) at 2020-09-29 05:54 UTC`
		`Nmap 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 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: 1 IP address (1 host up) scanned in 0.04 seconds`

Table 3-6 (Cont.) nmap

Options Tested	Description	Details

-v	Increase verbosity level	**Output:**
		`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	Details

`-iL`	Scan all the listed IP addresses in a file.
	Sample file
	
	`localhost`
	`10.178.254.194`
		Output:
		`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 details of dig debug tool are as follows:

Table 3-7 dig

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

Table 3-7 (Cont.) dig

Options Tested	Description	Output

`-x`	Query DNS Reverse Look-up.	**Output:**
		`dig -x 10.178.254.194`
		
		`; <<>> DiG 9.9.4-RedHat-9.9.4-61.el7`
		`<<>> -x 10.178.254.194`
		`;; global options: +cmd`
		`;; Got answer:`
		`;; ->>HEADER<<- opcode: QUERY, status:`
		`NOERROR, id: 59845`
		`;; flags: qr aa rd ra; QUERY: 1,`
		`ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1`
		
		`;; OPT PSEUDOSECTION:`
		`; EDNS: version: 0, flags:; udp: 4096`
		`;; QUESTION SECTION:`
		`;194.254.178.10.in-addr.arpa. IN PTR`
		
		`;; ANSWER SECTION:`
		`194.254.178.10.in-addr.arpa. 0 IN PTR`
		`master.openstack.internal.`
		
		`;; Query time: 1 msec`
		`;; SERVER:`
		`10.178.254.131#53(10.178.254.131)`
		`;; WHEN: Tue Sep 29 08:08:29 GMT 2020`
		`;; MSG SIZE rcvd: 95`

3.1 Debug Tool Configuration Parameters

Following are the parameters used to configure NSSF 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`	`spec` defines the policy enforced.
`spec.readOnlyRootFilesystem`	Controls whether the containers run with a read-only root filesystem (i.e. no writable layer).

Table 3-8 (Cont.) CNE Parameters

Parameter	Description
`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 allowed 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 versioned 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.
`rules`	Rules holds all the Policy Rules for this Role
`apiGroups`	`apiGroups` is the name of the API Group 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 allowlist of names that the rule applies to.

Table 3-10 Role Binding Creation

Parameter	Description
`apiVersion`	`apiVersion` defines the versioned 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.

Table 3-10 (Cont.) Role Binding Creation

Parameter	Description
`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 the resource being referenced
`subjects`	Subjects hold 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 `apiGroup` 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`	Container for debug
`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 describe the maximum amount of compute resources allowed
`resources.requests`	Requests describe the minimum amount of compute resources required
`resources.limits.cpu`	CPU limits
`resources.limits.memory`	Memory limits
`resources.limits.ephemeral-storage`	Ephemeral Storage limits
`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.readOnlyRootFilesystem`	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.

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

Parameter	Description
`securityContext.capabilities.drop`	Removed capabilities
`secuirtyContext.capabilities.add`	Added capabilities
`securityContext.runAsUser`	The UID to run the entry point of the container process. Debug Tool Container comes up with the default `<user-id>` as **7000**. If the operator wants to override this default value, it can be done using the `` `runAsUser` `` field. Otherwise, the field can be skipped. Default value: uid=7000(debugtool) gid=7000(debugtool) groups=7000(debugtool)
`volumeMounts.mountPath`	Indicates the path for volume mount.
`volumeMounts.name`	Indicates the name of the directory for debug tool logs storage.

4

Troubleshooting NSSF

This section provides information about how to identify problems and a systematic approach to resolve the identified issues. It also includes a generic checklist to help identify the problem.

Note

The performance and capacity of the NSSF system may vary based on the call model, Feature or Interface configuration, and underlying CNE and hardware environment.

4.1 Generic Checklist

The following sections provide a generic checklist for troubleshooting tips.

Deployment related tips

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

  Run the following command:

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

  Inspect the output, check the following columns:

  - AVAILABLE of deployment
  - READY, STATUS, and RESTARTS of pods
  - PORT(S) of service

- Is the correct image used and the correct environment variables set in the deployment? Run following the command:

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

  Inspect the output, check the environment and image.

  ```
  # kubectl -n nssf-svc get deployment ocnssf-nfregistration -o yaml
  apiVersion: extensions/v1beta1
  kind: Deployment metadata:
                annotations:
                deployment.kubernetes.io/revision: "1"
              kubectl.kubernetes.io/last-applied-configuration: |
              {"apiVersion":"apps/v1","kind":"Deployment","metadata":
              {"annotations":{},"name":"ocnssf-
  nfregistration","namespace":"nssfsvc"},
              "spec":{"replicas":1,"selector":{"matchLabels": {"app":"ocnssf-
  nfregistration"}},
              "template":{"metadata": {"labels":{"app":"ocnssf-
  ```

```
nfregistration"}},
            "spec": {"containers":[{"env":
[{"name":"MYSQL_HOST","value":"mysql"},
            {"name":"MYSQL_PORT","value":"3306"},
            {"name":"MYSQL_DATABASE","value":"nssfdb"},

{"name":"nssf_REGISTRATION_ENDPOINT","value":"ocnssfnfregistration"},

{"name":"nssf_SUBSCRIPTION_ENDPOINT","value":"ocnssfnfsubscription"},
            {"name":"NF_HEARTBEAT","value":"120"},
            {"name":"DISC_VALIDITY_PERIOD","value":"3600"}],
            "image":"dsrmaster0:5000/ocnssfnfregistration:latest",

"imagePullPolicy":"Always","name":"ocnssfnfregistration","ports":
            [{"containerPort":8080,"name":"server"}]}]}}}}
            creationTimestamp: 2018-08-27T15:45:59Z  generation: 1
            name: ocnssf-nfregistration  namespace: nssf-svc
            resourceVersion: "2336498"
            selfLink: /apis/extensions/v1beta1/namespaces/ nssf-svc/
deployments/ocnssf-nfregistration
            uid: 4b82fe89-aa10-11e8-95fd-fa163f20f9e2
            spec:  progressDeadlineSeconds: 600
            replicas: 1
            revisionHistoryLimit: 10
            selector:
            matchLabels:
            app: ocnssf-nfregistration
            strategy:
            rollingUpdate:
            maxSurge: 25%
            maxUnavailable: 25%
            type: RollingUpdate
            template:
            metadata:
            creationTimestamp: null
            labels:
            app: ocnssf-nfregistration
            spec:
            containers:
            - env:
            - name: MYSQL_HOST
            value: mysql
            - name: MYSQL_PORT
            value: "3306"
            - name: MYSQL_DATABASE
            value: nssfdb
            - name: nssf_REGISTRATION_ENDPOINT
            value: ocnssf-nfregistration
            - name: nssf_SUBSCRIPTION_ENDPOINT
            value: ocnssf-nfsubscription

            - name: NF_HEARTBEAT
            value: "120"
            - name: DISC_VALIDITY_PERIOD
            value: "3600"
            image: dsr-master0:5000/ocnssf-nfregistration:latest
            imagePullPolicy: Always
            name: ocnssf-nfregistration
            ports:        - containerPort: 8080
            name: server
            protocol: TCP
            resources: {}
            terminationMessagePath: /dev/termination-log
            terminationMessagePolicy: File
            dnsPolicy: ClusterFirst
            restartPolicy: Always
            schedulerName: default-scheduler
            securityContext: {}
            terminationGracePeriodSeconds: 30
            status:
            availableReplicas: 1
            conditions:  - lastTransitionTime: 2018-08-27T15:46:01Z
            lastUpdateTime: 2018-08-27T15:46:01Z
            message: Deployment has minimum availability.
            reason: MinimumReplicasAvailable
            status: "True"
            type: Available
            - lastTransitionTime: 2018-08-27T15:45:59Z
            lastUpdateTime: 2018-08-27T15:46:01Z
            message: ReplicaSet
            "ocnssf-nfregistration-7898d657d9" has successfully
progressed.
            reason: NewReplicaSetAvailable
            status: "True"
            type: Progressing
            observedGeneration: 1
            readyReplicas: 1
            replicas: 1
            updatedReplicas: 1
```

- Check if the microservices can access each other through a REST interface. Run the following command:

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

  Example:

  ```
  # kubectl -n nssf-svc exec $(kubectl -n nssf-svc get pods -o name|cut -
  d'/' -f2|grep nfs) -
          curl http://ocnssf-nfregistration:8080/nnssf-nfm/v1/nfinstances
  # kubectl -n nssf-svc exec $(kubectl -n nssf-svc get pods -o name|cut -
  d'/' -f2|grep nfr) -
  curl http://ocnssf-nfsubscription:8080/nnssf-nfm/v1/nfinstances
  ```

> **Note**
>
> These commands are in their simple form and display the logs only if there is 1 nssf<registration> and nssf<unscription> pod deployed.

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 patterns in the log output.

  Example:

  ```
  # kubectl -n nssf-svc logs -f $(kubectl -n nssf-svc get pods -o name|cut -
  d'/' -f2|grep nfr)
  # kubectl -n nssf-svc logs -f $(kubectl -n nssf-svc get pods -o name|cut -
  d'/' -f2|grep nfs)
  ```

Note

These commands are in their simple form and display the logs only if there is 1 nssf<egistration> and nfs<ubscription> pod deployed.

4.2 Deployment Related Issues

This section describes the most common deployment related issues and their resolution steps. Users are recommended to attempt the resolution steps provided in this guide before contacting Oracle Support.

4.2.1 Preinstallation

This section describes the common preinstallation issues and their resolution steps.

4.2.1.1 Debugging General CNE

Problem: The environment is not working as expected.

Solution:

Run the following command to get all the events:

```
kubectl get events -n <ocnssf_namespace>
```

4.2.1.1.1 The Environment is Not Working As Expected

Problem: The environment is not working as expected.

Solution:

1. Check if kubectl is installed and working as expected.
2. Check if `kubectl version` command works. This displays the Kubernetes client and server versions.
3. Check if `kubectl create namespace test` command works.
4. Check if `kubectl delete namespace test` command works.
5. Check if helm is installed and working as expected.

6. Check if `helm version` command works. This displays the helm client and server versions.

4.2.1.2 Curl HTTP2 Not Supported

Problem

The system does not support Curl HTTP2.

Error Code or Error Message

Unsupported protocol error is thrown or connection is established with HTTP/1.1 200 OK

Symptom

If unsupported protocol error is thrown or connection is established with HTTP1.1, it is an indication that Curl HTTP2 support is unavailable on your machine.

Solution

Following is the procedure to install Curl with HTTP2 support:

1. Make sure git is installed:

```
$ sudo yum install git -y
```

2. Install nghttp2:

```
$ git clone https://github.com/tatsuhiro-t/nghttp2.git
 $ cd nghttp2 $ autoreconf -i
$ automake
$ autoconf
$ ./configure
$ make
$ sudo make install
$ echo '/usr/local/lib' > /etc/ld.so.conf.d/custom-libs.conf
$ ldconfig
```

3. Install the latest Curl:

```
$ wget http://curl.haxx.se/download/curl-7.46.0.tar.bz2 (NOTE: Check for
latest version during Installation)
$ tar -xvjf curl-7.46.0.tar.bz2
$ cd curl-7.46.0
$ ./configure --with-nghttp2=/usr/local --with-ssl
$ make
$ sudo make install
$ sudo ldconfig
```

4. Run the following command to verify that HTTP2 is added in features:

```
$ curl --http2-prior-knowledge -v "<http://10.75.204.35:32270/nnrf
disc/v1/nf-instances?requester-nf-type=AMF&target-nf-type=SMF>"
```

4.2.2 Installation

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

4.2.2.1 Helm Install Failure

This section describes the various scenarios in which `helm install` might fail. Following are some of the scenarios:

- Incorrect image name in ocnssf-custom-values files
- Docker registry is configured incorrectly
- Continuous Restart of Pods

4.2.2.1.1 Incorrect image name in ocnssf-custom-values files

Problem

`helm install` might fail if an incorrect image name is provided in the `ocnssf_custom_values_25.2.200.yaml` file.

Error Code/Error Message

When `kubectl get pods -n <ocnssf_namespace>` is performed, the status of the pods might be ImagePullBackOff or ErrImagePull.

For example:

```
$ kubectl get pods -n ocnssf

NAME                                      READY STATUS RESTARTS AGE
ocnssf-appinfo-7969c9fbf7-4fmgj                1/1  Running   0   18m
ocnssf-config-server-54bf4bc8f9-s82cv          1/1  Running   0   18m
ocnssf-egress-6b6bff8949-2mf7b                 1/1  ImagePullBackOff   0
18m
ocnssf-ingress-68d76954f5-9fsfq                1/1  Running   0   18m
ocnssf-nrf-client-nfdiscovery-cf48cd8d8-l4q2q  1/1  Running   0   18m
ocnssf-nrf-client-nfdiscovery-cf48cd8d8-vmt5v  1/1  Running   0   18m
ocnssf-nrf-client-nfmanagement-7db4598fbb-672hc 1/1  Running   0   18m
ocnssf-nsavailability-644999bbfb-9gcm5         1/1  Running   0   18m
ocnssf-nsconfig-577446c487-dzsh6               1/1  Running   0   18m
ocnssf-nsdb-585f7bd7d-tdth4                    1/1  Running   0   18m
ocnssf-nsselection-5dfcc94bc7-q9gct            1/1  Running   0   18m
ocnssf-nssubscription-5c898fbbb9-fqcw6         1/1  Running   0   18m
ocnssf-performance-6d75c7f966-qm5fq            1/1  Running   0   18m
```

Solution

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

1. Check `ocnssf_custom_values_25.2.200.yaml` file has the release specific image name and tags.

   ```
   vi ocnssf_custom_values_25.2.200.yaml
   ```

For NSSF images details, see "**Customizing NSSF**" in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

2. Edit `ocnssf_custom_values_25.2.200.yaml` file in case the release specific image name and tags must be modified.
3. Save the file.
4. Run the following command to delete the deployment:

   ```
   helm delete --purge <release_namespace>
   ```

   Sample command:

   ```
   helm delete --purge ocnssf
   ```

5. In case the helm purge does not clean the deployment and Kubernetes objects completely, then see the "**Cleaning NSSF deployment**" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
6. Run `helm install` command. For helm install command, see the "**Customizing NSSF**" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
7. Run `kubectl get pods -n <ocnssf_namespace>` to verify if the status of all the pods is **Running**.
   For example:

   ```
   $ kubectl get pods -n ocnssf

   NAME                                    READY STATUS RESTARTS AGE
   ocnssf-appinfo-7969c9fbf7-4fmgj                   1/1  Running   0   18m
   ocnssf-config-server-54bf4bc8f9-s82cv             1/1  Running   0   18m
   ocnssf-egress-6b6bff8949-2mf7b                    1/1  Running   0   18m
   ocnssf-ingress-68d76954f5-9fsfq                   1/1  Running   0   18m
   ocnssf-nrf-client-nfdiscovery-cf48cd8d8-l4q2q     1/1  Running   0   18m
   ocnssf-nrf-client-nfdiscovery-cf48cd8d8-vmt5v     1/1  Running   0   18m
   ocnssf-nrf-client-nfmanagement-7db4598fbb-672hc 1/1  Running   0   18m
   ocnssf-nsavailability-644999bbfb-9gcm5            1/1  Running   0   18m
   ocnssf-nsconfig-577446c487-dzsh6                  1/1  Running   0   18m
   ocnssf-nsdb-585f7bd7d-tdth4                       1/1  Running   0   18m
   ocnssf-nsselection-5dfcc94bc7-q9gct               1/1  Running   0   18m
   ocnssf-nssubscription-5c898fbbb9-fqcw6            1/1  Running   0   18m
   ocnssf-performance-6d75c7f966-qm5fq               1/1  Running   0   18m
   ```

4.2.2.1.2 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/Error Message

When `kubectl get pods -n <ocnssf_namespace>` is performed, the status of the pods might be ImagePullBackOff or ErrImagePull.

For example:

```
$ kubectl get pods -n ocnssf

NAME                                      READY STATUS RESTARTS AGE
ocnssf-appinfo-7969c9fbf7-4fmgj                     1/1  Running   0   18m
ocnssf-config-server-54bf4bc8f9-s82cv               1/1  Running   0   18m
ocnssf-egress-6b6bff8949-2mf7b                      1/1  ImagePullBackOff   0
18m
ocnssf-ingress-68d76954f5-9fsfq                     1/1  Running   0   18m
ocnssf-nrf-client-nfdiscovery-cf48cd8d8-l4q2q       1/1  Running   0   18m
ocnssf-nrf-client-nfdiscovery-cf48cd8d8-vmt5v       1/1  Running   0   18m
ocnssf-nrf-client-nfmanagement-7db4598fbb-672hc     1/1  Running   0   18m
ocnssf-nsavailability-644999bbfb-9gcm5              1/1  Running   0   18m
ocnssf-nsconfig-577446c487-dzsh6                    1/1  Running   0   18m
ocnssf-nsdb-585f7bd7d-tdth4                         1/1  Running   0   18m
ocnssf-nsselection-5dfcc94bc7-q9gct                 1/1  Running   0   18m
ocnssf-nssubscription-5c898fbbb9-fqcw6              1/1  Running   0   18m
ocnssf-performance-6d75c7f966-qm5fq                 1/1  Running   0   18m
```

Solution

Configure docker registry on all primary and secondary nodes. For more information on configuring the docker registry, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

4.2.2.1.3 Continuous Restart of Pods

Problem

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

Error Code/Error Message

When `kubectl get pods -n <ocnssf_namespace>` is performed, the pods restart count increases continuously.

For example:

```
$ kubectl get pods -n ocnssf

NAME                                      READY STATUS RESTARTS AGE
ocnssf-appinfo-7969c9fbf7-4fmgj                     1/1  Running   0   18m
ocnssf-config-server-54bf4bc8f9-s82cv               1/1  Running   0   18m
ocnssf-egress-6b6bff8949-2mf7b                      1/1  Running   0   18m
ocnssf-ingress-68d76954f5-9fsfq                     1/1  Running   0   18m
ocnssf-nrf-client-nfdiscovery-cf48cd8d8-l4q2q       1/1  Running   0   18m
ocnssf-nrf-client-nfdiscovery-cf48cd8d8-vmt5v       1/1  Running   0   18m
ocnssf-nrf-client-nfmanagement-7db4598fbb-672hc     1/1  Running   0   18m
ocnssf-nsavailability-644999bbfb-9gcm5              1/1  Running   0   18m
ocnssf-nsconfig-577446c487-dzsh6                    1/1  Running   0   18m
ocnssf-nsdb-585f7bd7d-tdth4                         1/1  Running   0   18m
ocnssf-nsselection-5dfcc94bc7-q9gct                 1/1  Running   0   18m
ocnssf-nssubscription-5c898fbbb9-fqcw6              1/1  Running   0   18m
ocnssf-performance-6d75c7f966-qm5fq                 1/1  Running   0   18m
```

Solution

MySQL servers(s) may not be configured properly according to the pre-installation steps. For configuring MySQL servers, see the "**Configuring MySQL Database and User**" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

4.2.2.1.4 Tiller Pod Failure

Problem

Tiller Pod is not ready to run helm install.

Error Code/Error Message

The error 'could not find a ready tiller pod' message is received.

Symptom

When you run helm ls command and receive, 'could not find a ready tiller pod' message error.

Solution

Following is the procedure to install helm and tiller using the below commands:

1. Delete the preinstalled helm:

```
kubectl delete svc tiller-deploy -n kube-system kubectl delete deploy tiller-
deploy -n kube-system
```

2. Install helm and tiller using these commands:

```
helm init --client-only
helm plugin install https://github.com/rimusz/
helm-tiller helm tiller install
helm tiller start kube-system
```

4.2.2.2 Custom Value File Parsing Failure

This section explains troubleshooting procedure in case of failure while parsing `ocnssf_custom_values_25.2.200.yaml` file.

Problem

Not able to parse `ocnssf_custom_values_25.2.200.yaml`, while running `helm install`.

Error Code/Error Message

Error: failed to parse `ocnssf_custom_values_25.2.200.yaml`: error converting YAML to JSON: yaml

Symptom

While creating the `ocnssf_custom_values_25.2.200.yaml` file, if the aforementioned error is received, it means that the file is not created properly. The tree structure may not have been followed or there may also be tab spaces in the file.

Solution

Following the procedure as mentioned:

1. Download the latest NSSF templates zip file from MOS. For more information, see the "**Downloading the NSSF Package and Custom Template ZIP file**" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
2. Follow the steps mentioned in the "**Installation Tasks**" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

4.2.3 Post installation

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

4.2.3.1 Helm Test Error Scenarios

Identify error scenarios using the helm test as follows:

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

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

2. Check for the Helm Test pod that is in the error state.
3. Run the following command to get the logs:

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

   Example:

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

   Depending on the failure reasons, perform the resolution steps.

For further assistance, collect the logs and contact MOS.

4.3 Upgrade or Rollback Failure

When Oracle Communications Cloud Native Core, Network Slice Selection Function (NSSF) upgrade or rollback fails, perform the following procedure.

1. Check the pre or post upgrade logs or rollback hook logs in Kibana as applicable. Users can filter upgrade or rollback logs using the following filters:
   - For upgrade: lifeCycleEvent=9001
   - For rollback: lifeCycleEvent=9002

   ```
   {
      "time_stamp":"2021-08-23 06:45:57.698+0000",
      "thread":"main",
      "level":"INFO",
   ```

```
"logger":"com.oracle.cgbu.cne.ocnssf.hooks.releases.ReleaseHelmHook_1_14_1"
,
   "message":"{logMsg=Starting Pre-Upgrade hook Execution,
lifeCycleEvent=9001 | Upgrade, sourceRelease=101400,
targetRelease=101401}",

"loc":"com.oracle.cgbu.ocnssf.common.utils.EventSpecificLogger.submit(Event
SpecificLogger.java:94)"
}
```

2. Check the pod logs in Kibana to analyze the cause of failure.
3. After detecting the cause of failure, do the following:
   - For upgrade failure:
     – If the cause of upgrade failure is a database or network connectivity issue, then resolve the issue and rerun the upgrade command.
     – If the cause of failure is not related to a database or network connectivity issue and is observed during the preupgrade phase, then do not perform rollback because NSSF deployment remains in the source or older release.
     – If the upgrade failure occurs during the postupgrade phase, for example, post upgrade hook failure due to target release pod not moving to ready state, then perform a rollback.
   - For rollback failure: If the cause of rollback failure is a database or network connectivity issue, contact your system administrator. When the issue is resolved, rerun the rollback command.
4. If the issue persists, contact My Oracle Support.

4.3.1 Replication Channel Breaks While Rolling Back cnDBTier from 23.4.x to 23.3.x.

Scenario

Replication Channel has broken while doing a rollback of cnDBTier from 23.4.x to 23.3.x.

Problem

Intermittently, during rollback of cnDBTier from 23.4.x to 23.3.x in georedundant scenario, the replication is going down.

Solution

As a workaround, follow the recovery procedure explained in the sections, "Resolving Georeplication Failure Between cnDBTier Clusters in a Two Site Replication" and "Resolving Georeplication Failure Between cnDBTier Clusters in a Three Site Replication " in *Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide* to recover the replication.

4.3.2 Helm Hook Failure and NrfClient Discovery Restart

Scenario

Helm Hook Failure and NrfClient Discovery Restart

Problem

During the upgrade process, the NSSF hooks are responsible for updating the `common_configuration` table, which maintains one record per service per version. The hooks perform the following actions:

- **Preinstall/Preupgrade:** Each service adds a record to the table with its name, release, and current configuration.
- **Postupgrade:** Removes entries related to the previous release of the same service.

If a Helm hook fails due to issues such as connectivity problems or network glitches, subsequent hooks do not execute. This can result in the `postupgrade` hook failing to run, leaving the `common_configuration` table in an inconsistent state—such as having duplicate entries or missing records for the current release.

Impact:

When the `postupgrade` hook fails and the table is not updated correctly, it can cause the NrfClient Discovery service to restart. This occurs because the NrfClient relies on receiving a 200 OK response from the `config_server`, which is only returned when a valid record exists in the `common_configuration` table for the current release. The inconsistency leads to delays in the NrfClient's startup process.

Solution

Due to Helm's behavior, a failed hook prevents subsequent hooks from running. As a result, cleanup actions, such as removing old release entries and insertion of valid records for the current release may not occur. This prevents the NrfClient from starting correctly.

To ensure the NrfClient Discovery service starts as expected, perform the following steps:

1. **Manual Cleanup:** If a Helm hook failure occurs, manually remove any entries related to previous releases from the `common_configuration` table. Ensure there is only one entry per service for the current release.
2. **Verify Current Release Record:** Confirm that the `common_configuration` table contains a valid record for each service corresponding to the current release.

4.4 Database Related Issues

This section describes the most common database related issues and their resolution steps. It is recommended to attempt the resolution steps provided in this guide before contacting Oracle Support.

4.4.1 NSSF MySQL Database Access

Problem

Keyword - wait-for-db

Tags - "config-server" "database" "readiness" "init" "SQLException" "access denied"

Due to database accessibility issues from the NSSF service, pods will stay in the init state.

Even though some pods are up, they still keep receiving the following exception: " Cannot connect to database server java.sql.SQLException"

Reasons:

1. MySQL host IP address OR MySQL-service name[in case of occne-infra] is incorrect.
2. Few MySQL nodes are probably down.

3. The username or password given in the secrets are not created in the database or do not have proper grant or access to service databases.
4. Databases are not created correctly with the same name mentioned in the `ocnssf_custom_values_25.2.200.yaml` file while installing NSSF.

Resolution Steps

To resolve this issue, perform the following steps:

1. Check if the database IP is proper and pingable from worker nodes of the Kubernetes cluster. Update the database IP and service accordingly. If required, you can use floating IP as well. If the database connectivity persists, then update the correct IP address.
   In the case of OCCNE-infra, instead of mentioning IP address for MySQL connection, use FQDN for mysql-connectivity-service to connect to the database.
2. Manually log in to MySQL through the same database IP as mentioned in the `ocnssf_custom_values_25.2.200.yaml` file. In case of MySQL service name, run the following command to describe the service:

   ```
   kubectl describe svc <mysql-servicename> -n <namespace>
   ```

   Log in to the MySQL database with all sets of IPs described in the MySQL service. If any SQL node is down, it can lead to an intermittent database query failure. So, make sure that you can log in to MySQL from all the nodes mentioned in the IP list of MySQL service describe command.
   Make sure that all the MySQL nodes are up and running before installing NSSF.
3. Check the existing user list into the database using SQL query: "select user from mysql.user;"
   Check if all the mentioned users in the custom-value of NSSF installation are present in the database.

   Note

   Create the user with correct password as mentioned in the secret file of the NSSF.

4. Check the grants of all the users mentioned into the `ocnssf_custom_values_25.2.200.yaml` file by SQL query: "show grants for <username>;"
   If a username or password issue persists, then correctly create a user with the required password and also provide the grants as per the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
5. Check if the databases are created with the same name as mentioned in the `ocnssf_custom_values_25.2.200.yaml` file for the services.

   Note

   Create the database as per the `ocnssf_custom_values_25.2.200.yaml` file.

6. Check if problematic pods are getting created on any one unique worker node. If yes, then may be the cause of the error can be the worker node. Try draining the problematic worker node and allowing pods to move to another node.

4.4.2 "Communications link failure" Error

Problem

In two-site georedundant deployment, the "Communications link failure" error is observed in the logs of every Ns-Selection pod at both the sites.
This error indicates a connection problem between the NSSF (Network Slice Selection Function) pods and the cnDBTier (Cloud Native Database Tier), potentially disrupting data exchange and regular functionality.

The `com.mysql.cj.jdbc.exceptions.CommunicationsException` error typically signifies a failed connection attempt between NSSF and the MySQL database server.

Resolution Steps

When there is no data or messages loss, this error can be ignored. Here's why ignoring this error can be considered under certain circumstances.

Temporary Network Hiccups: Brief network interruptions, often caused by switch issues, router fluctuations, or temporary network congestion, can lead to connection failures without impacting data integrity. These interruptions result in the "CommunicationsException" error, but since the interruptions are momentary, and no data transfer occurs in this small amount of time, so no data is lost.

4.5 Troubleshooting TLS Related Issues

This section describes the TLS related issues and their resolution steps. It is recommended to attempt the resolution steps provided in this guide before contacting Oracle Support.

Problem: Handshake is not established between NSSFs.

Scenario: When the client version is TLSv1.2 and the server version is TLSv1.3.

Server Error Message

```
The client supported protocol versions[TLSv1.2] are not accepted by server
preferences [TLSv1.3]
```

Client Error Message

```
Received fatal alert: protocol_version
```

Scenario: When the client version is TLSv1.3 and the server version is TLSv1.2.

Server Error Message

```
The client supported protocol versions[TLSv1.3]are not accepted by server
preferences [TLSv1.2]
```

Client Error Message

```
Received fatal alert: protocol_version
```

Solution:

If the error logs have the SSL exception, do the following:

Check the TLS version of both NSSFs, if both support different and single TLS versions, (that is, NSSF 1 supports TLS 1.2 only and NSSF 2 supports TLS 1.3 only or vice versa), handshake fails. Ensure that the TLS version is same for both NSSFs or revert to default configuration for both NSSFs. The TLS version communication supported are:

Table 4-1 TLS Version Used

Client TLS Version	Server TLS Version	TLS Version Used
TLSv1.2, TLSv1.3	TLSv1.2, TLSv1.3	TLSv1.3
TLSv1.3	TLSv1.3	TLSv1.3
TLSv1.3	TLSv1.2, TLSv1.3	TLSv1.3
TLSv1.2, TLSv1.3	TLSv1.3	TLSv1.3
TLSv1.2	TLSv1.2, TLSv1.3	TLSv1.2
TLSv1.2, TLSv1.3	TLSv1.2	TLSv1.2

Check the cipher suites being supported by both NSSFs, it should be either the same or should have common cipher suites present. If not, revert to default configuration.

Problem: Pods are not coming up after populating the `clientDisabledExtension` or `serverDisabledExtension` Helm parameter.

Solution:

- Check the value of the `clientDisabledExtension` or `serverDisabledExtension` parameters. The following extensions should not be present for these parameters:
  - supported_versions
  - key_share
  - supported_groups
  - signature_algorithms
  - pre_shared_key

If any of the above values is present, remove them or revert to default configuration for the pod to come up.

Problem: Pods are not coming up after populating the `clientSignatureSchemes` Helm parameter.

Solution:

- Check the value of the `clientSignatureSchemes` parameter.
- The following values should be present for this parameter:
  - rsa_pkcs1_sha512
  - rsa_pkcs1_sha384
  - rsa_pkcs1_sha256

  If any of the above values is not present, add them or revert to default configuration for the pod to come up.

5
Alerts

This section provides information about alerts for Oracle Communications Cloud Native Core, Network Slice Selection Function (NSSF).

5.1 System Level Alerts

This section lists the system level alerts.

5.1.1 OcnssfNfStatusUnavailable

Table 5-1 OcnssfNfStatusUnavailable

Field	Details
Description	'OCNSSF services unavailable'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : All OCNSSF services are unavailable.'
Severity	Critical
Condition	All the NSSF services are unavailable, either because the NSSF is getting deployed or purged. These NSSF services considered are nssfselection, nssfsubscription, nssfavailability, nssfconfiguration, appinfo, ingressgateway and egressgateway.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9001
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-1 (Cont.) OcnssfNfStatusUnavailable

Field	Details

Recommended Actions	The alert is cleared automatically when the NSSF services 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 --sort-by=.metadata.creationTimestamp -n <namespace>`
	3. Refer to the application logs on Kibana and check for database related failures such as connectivity, invalid secrets, and so on. The logs can be filtered based on the services.
	4. Run the following command to check Helm status and make sure there are no errors:
	`helm status <helm release name of the desired NF> -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.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.2 OcnssfPodsRestart

Table 5-2 OcnssfPodsRestart

Field	Details
Description	'Pod <*Pod Name*> has restarted.
Summary	'kubernetes_namespace: {{$labels.namespace}}, podname: {{$labels.pod}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : A Pod has restarted'
Severity	Major
Condition	A pod belonging to any of the NSSF services has restarted.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9002
Metric Used	'kube_pod_container_status_restarts_total'Note: This is a Kubernetes metric. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-2 (Cont.) OcnssfPodsRestart

Field	Details

Recommended Actions	The alert is cleared automatically if the specific pod is up.
	Steps:
	1. Refer to the application logs on Kibana and filter based on the pod name. 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, cnDBTier User Guide*".
	4. If the issue persists, capture all the outputs from the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.3 OcnssfSubscriptionServiceDown

Table 5-3 OcnssfSubscriptionServiceDown

Field	Details
Description	'OCNSSF Subscription service <*ocnssf-nssubscription*> is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : NssfSubscriptionServiceDown service down'
Severity	Critical
Condition	NssfSubscription services is unavailable.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9003
Metric Used	"up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-3 (Cont.) OcnssfSubscriptionServiceDown

Field	Details

Recommended Actions	The alert is cleared when the NssfSubscription services is available.
	Steps:
	1. Check if NfService specific alerts are generated to understand which service is down.
	If the following alerts are generated based on which service is down
	OcnssfSubscriptionServiceDown
	2. Run the following command to check the orchestration log nfsubscription service and check 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 <specific desired full pod name> -n <namespace>`
	3. 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 events:
	`kubectl get events --sort-by=.metadata.creationTimestamp -n <namespace>`
	4. Refer to the application logs on Kibana and filter based on above service names. Check for ERROR WARNING logs for each of these services.
	5. Check the database status. For more information, see "*Oracle Communications Cloud Native Core, cnDBTier User Guide*".
	6. Refer to the application logs on Kibana and check for the service status of the nssfConfig service.
	7. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.4 OcnssfSelectionServiceDown

Table 5-4 OcnssfSelectionServiceDown

Field	Details
Description	'OCNSSF Selection service <ocnssf-nsselection> is down'**.**

Table 5-4 (Cont.) OcnssfSelectionServiceDown

Field	Details
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : OcnssfSelectionServiceDown service down'
Severity	Critical
Condition	None of the pods of the NSSFSelection microservice is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9004
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Recommended Actions	The alert is cleared when the nfsubscription service is available.
	Steps:
	1. Run the following command to check the orchestration logs of ocnssf-nsselection service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on ocnssf-nsselection service names. Check for ERROR WARNING logs.
	3. Check the database status. For more information, see "*Oracle Communications Cloud Native Core, cnDBTier User Guide*".
	4. Depending on the failure reason, take the resolution steps.
	5. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.5 OcnssfAvailabilityServiceDown

Table 5-5 OcnssfAvailabilityServiceDown

Field	Details
Description	'Ocnssf Availability service ocnssf-nsavailability is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : NssfAvailability service down'
Severity	Critical
Condition	None of the pods of the OcnssfAvailabilityServiceDown microservice is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9005

Table 5-5 (Cont.) OcnssfAvailabilityServiceDown

Field	Details
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Recommended Actions	The alert is cleared when the ocnssf-nsavailability service is available.
	Steps:
	1. Run the following command to check the orchestration logs of ocnssf-nsavailability service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on ocnssf-nsavailability service names. Check for ERROR WARNING logs.
	3. Check the database status. For more information, see "*Oracle Communications Cloud Native Core, cnDBTier User Guide*".
	4. Depending on the failure reason, take the resolution steps.
	5. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.6 OcnssfConfigurationServiceDown

Table 5-6 OcnssfConfigurationServiceDown

Field	Details
Description	'OCNSSF Config service *nssfconfiguration* is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : OcnssfConfigServiceDown service down'
Severity	Critical
Condition	None of the pods of the NssfConfiguration microservice is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9006
Metric Used	'up' **Note:** : This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-6 (Cont.) OcnssfConfigurationServiceDown

Field	Details

Recommended Actions	The alert is cleared when the nssfconfiguration service is available.
	Steps:
	1. Run the following command to check the orchestration logs of nssfconfiguration service and check for liveness or readiness probe failures:
	`kubectl get po -n <namespace>`
	Note the full name of the pod that is not running, and use in the following command:
	`kubectl describe pod <specific desired full pod name> -n <namespace>`
	2. Refer the application logs on Kibana and filter based on nssfconfiguration service names. Check for ERROR WARNING logs related to thread exceptions.
	3. Check the database status. For more information, see "*Oracle Communications Cloud Native Core, cnDBTier User Guide*".
	4. Depending on the reason of failure, take the resolution steps.
	5. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.7 OcnssfAppInfoServiceDown

Table 5-7 OcnssfAppInfoServiceDown

Field	Details
Description	OCNSSF Appinfo service *appinfo* is down'
Summary	kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : Appinfo service down'
Severity	Critical
Condition	None of the pods of the App Info microservice is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9025
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-7 (Cont.) OcnssfAppInfoServiceDown

Field	Details

Recommended Actions	The alert is cleared when the app-info service is available.
	Steps:
	1. Run the following command to check the orchestration logs of appinfo service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on appinfo service names. Check for ERROR WARNING logs related to thread exceptions.
	3. Depending on the failure reason, take the resolution steps.
	4. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.8 OcnssfIngressGatewayServiceDown

Table 5-8 OcnssfIngressGatewayServiceDown

Field	Details
Description	'Ocnssf Ingress-Gateway service *ingressgateway* is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : OcnssfIngressGwServiceDown service down'
Severity	Critical
Condition	None of the pods of the Ingress-Gateway microservice is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9007
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-8 (Cont.) OcnssfIngressGatewayServiceDown

Field	Details

Recommended Actions	The alert is cleared when the ingressgateway service is available.
	Steps:
	1. Run the following command to check the orchestration logs of ingress-gateway service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on ingress-gateway service names. Check for ERROR WARNING logs related to thread exceptions.
	3. Depending on the failure reason, take the resolution steps.
	4. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.9 OcnssfEgressGatewayServiceDown

Table 5-9 OcnssfEgressGatewayServiceDown

Field	Details
Description	'OCNSSF Egress service *egressgateway* is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : OcnssfEgressGwServiceDown service down'
Severity	Critical
Condition	None of the pods of the Egress-Gateway microservice is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9008
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-9 (Cont.) OcnssfEgressGatewayServiceDown

Field	Details

Recommended Actions	The alert is cleared when the egressgateway service is available.
	Note: The threshold is configurable in the alerts.yaml
	Steps:
	1. Run the following command to check the orchestration logs of egress-gateway service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on egress-gateway service names. Check for ERROR WARNING logs related to thread exceptions.
	3. Depending on the failure reason, take the resolution steps.
	4. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.10 OcnssfOcpmConfigServiceDown

Table 5-10 OcnssfOcpmConfigServiceDown

Field	Details
Description	'OCNSSF OCPM Config service is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : Ocnssf OCPM Config service down'
Severity	Critical
Condition	None of the pods of the ConfigService is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9027
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-10 (Cont.) OcnssfOcpmConfigServiceDown

Field	Details

Recommended Actions	The alert is cleared when the ConfigService is available.
	Note: The threshold is configurable in the alerts.yaml
	Steps:
	1. Run the following command to check the orchestration logs of ConfigService service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on PerfInfo service names. Check for ERROR WARNING logs related to thread exceptions.
	3. Depending on the failure reason, take the resolution steps.
	4. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.11 OcnssfPerfInfoServiceDown

Table 5-11 OcnssfPerfInfoServiceDown

Field	Details
Description	OCNSSF PerfInfo service is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : Ocnssf PerfInfo service down'
Severity	Critical
Condition	None of the pods of the PerfInfo service is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9026
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-11 (Cont.) OcnssfPerfInfoServiceDown

Field	Details

Recommended Actions	The alert is cleared when the PerfInfo service is available.
	Note: The threshold is configurable in the alerts.yaml
	Steps:
	1. Run the following command to check the orchestration logs of PerfInfo service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on PerfInfo service names. Check for ERROR WARNING logs related to thread exceptions.
	3. Depending on the failure reason, take the resolution steps.
	4. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.12 OcnssfNrfClientManagementServiceDown

Table 5-12 OcnssfNrfClientManagementServiceDown

Field	Details
Description	'OCNSSF NrfClient Management service is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : Ocnssf NrfClient Management service down'
Severity	Critical
Condition	None of the pods of the NrfClientManagement service is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9024
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-12 (Cont.) OcnssfNrfClientManagementServiceDown

Field	Details

Recommended Actions	The alert is cleared when the NrfClientManagement service is available.
	Note: The threshold is configurable in the alerts.yaml
	Steps:
	1. Run the following command to check the orchestration logs of NrfClientManagement service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on NrfClientManagement service names. Check for ERROR WARNING logs related to thread exceptions.
	3. Depending on the failure reason, take the resolution steps.
	4. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.13 OcnssfAlternateRouteServiceDown

Table 5-13 OcnssfAlternateRouteServiceDown

Field	Details
Description	'OCNSSF Alternate Route service is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : Ocnssf Alternate Route service down'
Severity	Critical
Condition	None of the pods of the Alternate Route service is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9023
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-13 (Cont.) OcnssfAlternateRouteServiceDown

Field	Details

Recommended Actions	The alert is cleared when the Alternate Route service is available.
	Note: The threshold is configurable in the alerts.yaml
	Steps:
	1. Run the following command to check the orchestration logs of Alternate Route service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on Alternate Route service names. Check for ERROR WARNING logs related to thread exceptions.
	3. Depending on the failure reason, take the resolution steps.
	4. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.14 OcnssfAuditorServiceDown

Table 5-14 OcnssfAuditorServiceDown

Field	Details
Description	'OCNSSF NsAuditor service is down'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : Ocnssf NsAuditor service down'
Severity	Critical
Condition	None of the pods of the NsAuditor service is available.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9022
Metric Used	'up' **Note:** This is a Prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.

Table 5-14 (Cont.) OcnssfAuditorServiceDown

Field	Details

Recommended Actions	The alert is cleared when the NsAuditor service is available.
	Note: The threshold is configurable in the alerts.yaml
	Steps:
	1. Run the following command to check the orchestration logs of NsAuditor service and check 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 <specific desired full pod name> -n <namespace>`
	2. Refer to the application logs on Kibana and filter based on NsAuditor service names. Check for ERROR WARNING logs related to thread exceptions.
	3. Depending on the failure reason, take the resolution steps.
	4. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support.
	Note: Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.1.15 OcnssfTotalIngressTrafficRateAboveMinorThreshold

Table 5-15 OcnssfTotalIngressTrafficRateAboveMinorThreshold

Field	Details
Description	'Ingress traffic Rate is above the configured minor threshold i.e. 64000 requests per second (current value is: {{ $value }})'
Summary	'timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }}: Traffic Rate is above 80 Percent of Max requests per second(80000)'
Severity	Minor
Condition	The total Ocnssf Ingress Message rate has crossed the configured minor threshold of 64000 TPS. Default value of this alert trigger point in NrfAlertValues.yaml is when Ocnssf Ingress Rate crosses 80 % of 80000 (Maximum ingress request rate).
OID	1.3.6.1.4.1.323.5.3.40.1.2.9009
Metric Used	'oc_ingressgateway_http_requests_total'

Table 5-15 (Cont.) OcnssfTotalIngressTrafficRateAboveMinorThreshold

Field	Details
Recommended Actions	The alert is cleared either when the total Ingress Traffic rate falls below the Minor threshold or when the total traffic rate crosses the Major threshold, in which case the OcnssfTotalIngressTrafficRateAboveMinorThreshold alert shall be raised. **Note:** The threshold is configurable in the alerts.yaml **Steps**: Reassess the reason why the NSSF is receiving additional traffic, for example, the mated site NSSF is unavailable in the georedundancy scenario. If this is unexpected, contact My Oracle Support. 1. Refer Grafana to determine which service is receiving high traffic. 2. Refer Ingress Gateway section in Grafana to determine an increase in 4xx and 5xx error codes. 3. Check Ingress Gateway logs on Kibana to determine the reason for the errors.

5.1.16 OcnssfTotalIngressTrafficRateAboveMajorThreshold

Table 5-16 OcnssfTotalIngressTrafficRateAboveMajorThreshold

Field	Details
Description	'Ingress traffic Rate is above the configured major threshold i.e. 72000 requests per second (current value is: {{ $value }})'
Summary	'timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }}: Traffic Rate is above 90 Percent of Max requests per second(80000)'
Severity	Major
Condition	The total Ocnssf Ingress Message rate has crossed the configured major threshold of 72000 TPS. Default value of this alert trigger point in NssfAlertValues.yaml is when Ocnssf Ingress Rate crosses 90 % of 80000 (Maximum ingress request rate).
OID	1.3.6.1.4.1.323.5.3.40.1.2.9010
Metric Used	'oc_ingressgateway_http_requests_total'
Recommended Actions	The alert is cleared when the total Ingress traffic rate falls below the major threshold or when the total traffic rate crosses the critical threshold, in which case the alert shall be raised. OcnssfTotalIngressTrafficRateAboveCriticalThreshold **Note:** The threshold is configurable in the alerts.yaml **Steps**: Reassess the reason why the NSSF is receiving additional traffic, for example, the mated site NSSF is unavailable in the georedundancy scenario. If this is unexpected, contact My Oracle Support. 1. Refer Grafana to determine which service is receiving high traffic. 2. Refer Ingress Gateway section in Grafana to determine an increase in 4xx and 5xx error codes. 3. Check Ingress Gateway logs on Kibana to determine the reason for the errors.

5.1.17 OcnssfTotalIngressTrafficRateAboveCriticalThreshold

Table 5-17 OcnssfTotalIngressTrafficRateAboveCriticalThreshold

Field	Details
Description	'Ingress traffic Rate is above the configured critical threshold i.e. 76000 requests per second (current value is: {{ $value }})'
Summary	'timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }}: Traffic Rate is above 95 Percent of Max requests per second(80000)'
Severity	Critical
Condition	The total Ocnssf Ingress Message rate has crossed the configured critical threshold of 76000 TPS. Default value of this alert trigger point in NrfAlertValues.yaml is when Ocnssf Ingress Rate crosses 95 % of 80000 (Maximum ingress request rate).
OID	1.3.6.1.4.1.323.5.3.40.1.2.9011
Metric Used	'oc_ingressgateway_http_requests_total'
Recommended Actions	The alert is cleared when the Ingress traffic rate falls below the critical threshold. **Note:** The threshold is configurable in the alerts.yaml **Steps**: Reassess the reason why the NSSF is receiving additional traffic, for example, the mated site NSSF is unavailable in the georedundancy scenario. If this is unexpected, contact My Oracle Support. 1. Refer Grafana to determine which service is receiving high traffic. 2. Refer Ingress Gateway section in Grafana to determine an increase in 4xx and 5xx error codes. 3. Check Ingress Gateway logs on Kibana to determine the reason for the errors.

5.1.18 OcnssfTransactionErrorRateAbove1Percent

Table 5-18 OcnssfTransactionErrorRateAbove1Percent

Field	Details
Description	Transaction Error rate is above 1 Percent of Total Transactions
Summary	Transaction Error Rate detected above 1 Percent of Total Transactions
Severity	Warning
Condition	The number of failed transactions has crossed the minor threshold of 1 percent of the total transactions.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9012
Metric Used	oc_ingressgateway_http_responses_total

Table 5-18 (Cont.) OcnssfTransactionErrorRateAbove1Percent

Field	Details
Recommended Actions	The alert is cleared when the number of failed transactions reduces from the 1% threshold of the total transactions or when the failed transactions crosses the 10% threshold in which case the OcnssfTransactionErrorRateAbove10Percent shall be raised. **Steps**: 1. Check the Service specific metrics to understand the specific service request errors. For example: ocnssf_nsselection_success_tx_total with statusCode ~= 2xx. 2. Verify the metrics per service, per method For example: Discovery requests can be deduced from the following metrics: Metrics="oc_ingressgateway_http_responses_total" Method="GET" NFServiceType="ocnssf-nsselection" Route_path="/nnssf-nsselection/v2/**" Status="503 SERVICE_UNAVAILABLE" 3. If guidance is required, contact My Oracle Support.

5.1.19 OcnssfTransactionErrorRateAbove10Percent

Table 5-19 OcnssfTransactionErrorRateAbove10Percent

Field	Details
Description	'Transaction Error rate is above 10 Percent of Total Transactions (current value is {{ $value }})'
Summary	'timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }}: Transaction Error Rate detected above 10 Percent of Total Transactions'
Severity	Minor
Condition	The number of failed transactions has crossed the minor threshold of 10 percent of the total transactions.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9013
Metric Used	'oc_ingressgateway_http_responses_total'

Table 5-19 (Cont.) OcnssfTransactionErrorRateAbove10Percent

Field	Details
Recommended Actions	The alert is cleared when the number of failed transactions reduces from the 10% threshold of the total transactions or when the failed transactions crosses the 25% threshold in which case the OcnssfTransactionErrorRateAbove25Percent shall be raised. **Steps**: 1. Check the Service specific metrics to understand the specific service request errors. For example: ocnssf_nsselection_success_tx_total with statusCode ~= 2xx. 2. Verify the metrics per service, per method For example: Discovery requests can be deduced from the following metrics: Metrics="oc_ingressgateway_http_responses_total" Method="GET" NFServiceType="ocnssf-nsselection" Route_path="/nnssf-nsselection/v2/**" Status="503 SERVICE_UNAVAILABLE" 3. If guidance is required, contact My Oracle Support.

5.1.20 OcnssfTransactionErrorRateAbove25Percent

Table 5-20 OcnssfTransactionErrorRateAbove25Percent

Field	Details
Description	'Transaction Error rate is above 25 Percent of Total Transactions (current value is {{ $value }})'
summary	'timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }}: Transaction Error Rate detected above 25 Percent of Total Transactions'
Severity	Major
Condition	The number of failed transactions has crossed the minor threshold of 25 percent of the total transactions.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9014
Metric Used	'oc_ingressgateway_http_responses_total'

Table 5-20 (Cont.) OcnssfTransactionErrorRateAbove25Percent

Field	Details
Recommended Actions	The alert is cleared when the number of failed transactions reduces from the 25% of the total transactions or when the number of failed transactions crosses the 50% threshold in which case the OcnssfTransactionErrorRateAbove50Percent shall be raised. **Steps**: 1. Check the Service specific metrics to understand the specific service request errors. For example: ocnssf_nsselection_success_tx_total with statusCode ~= 2xx. 2. Verify the metrics per service, per method For example: Discovery requests can be deduced from the following metrics: Metrics="oc_ingressgateway_http_responses_total" Method="GET" NFServiceType="ocnssf-nsselection" Route_path="/nnssf-nsselection/v2/**" Status="503 SERVICE_UNAVAILABLE" 3. If guidance is required, contact My Oracle Support.

5.1.21 OcnssfTransactionErrorRateAbove50Percent

Table 5-21 OcnssfTransactionErrorRateAbove50Percent

Field	Details
Description	'Transaction Error rate is above 50 Percent of Total Transactions (current value is {{ $value }})'
Summary	'timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }}: Transaction Error Rate detected above 50 Percent of Total Transactions'
Severity	Critical
Condition	The number of failed transactions has crossed the minor threshold of 50 percent of the total transactions.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9015
Metric Used	'oc_ingressgateway_http_responses_total

Table 5-21 (Cont.) OcnssfTransactionErrorRateAbove50Percent

Field	**Details**
Recommended Actions	The alert is cleared when the number of failed transactions is below 50 percent of the total transactions. **Steps**: 1. Check for service specific metrics to understand the specific service request errors. For example: ocnssf_nsselection_success_tx_total with statusCode ~= 2xx. 2. Verify the metrics per service, per method For example: Discovery requests can be deduced from the following metrics: Metrics="oc_ingressgateway_http_responses_total" Method="GET" NFServiceType="ocnssf-nsselection" Route_path="/nnssf-nsselection/v2/**" Status="503 SERVICE_UNAVAILABLE" 3. If guidance is required, contact My Oracle Support.

5.2 Application Level Alerts

This section lists the application level alerts.

5.2.1 OcnssfOverloadThresholdBreachedL1

Table 5-22 OcnssfOverloadThresholdBreachedL1

Field	**Details**
Description	'Overload Level of {{$labels.app_kubernetes_io_name}} service is L1'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, podname: {{$labels.kubernetes_pod_name}}: Overload Level of {{$labels.app_kubernetes_io_name}} service is L1'
Severity	Warning
Condition	NSSF Services have breached their configured threshold of Level L1 for any of the aforementioned metrics. Thresholds are configured for CPU, svc_failure_count, svc_pending_count, and memory.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9016
Metric Used	load_level

Table 5-22 (Cont.) OcnssfOverloadThresholdBreachedL1

Field	Details
Recommended Actions	The alert is cleared when the Ingress Traffic rate falls below the configured L1 threshold. **Note**: The thresholds can be configured using REST API. **Steps**: Reassess the reasons leading to NSSF receiving additional traffic. If this is unexpected, contact My Oracle Support. 1. Refer to alert to determine which service is receiving high traffic. It may be due to a sudden spike in traffic. For example: When one mated site goes down, the NFs move to the given site. 2. Check the service pod logs on Kibana to determine the reason for the errors. 3. If this is expected traffic, then the thresholds levels may be reevaluated as per the call rate and reconfigured as mentioned in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.2.2 OcnssfOverloadThresholdBreachedL2

Table 5-23 OcnssfOverloadThresholdBreachedL2

Field	Details
Description	'Overload Level of {{$labels.app_kubernetes_io_name}} service is L2'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, podname: {{$labels.kubernetes_pod_name}}: Overload Level of {{$labels.app_kubernetes_io_name}} service is L2'
Severity	Minor
Condition	NSSF Services have breached their configured threshold of Level L2 for any of the aforementioned metrics. Thresholds are configured for CPU, svc_failure_count, svc_pending_count, and memory.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9017
Metric Used	load_level
Recommended Actions	The alert is cleared when the Ingress Traffic rate falls below the configured L2 threshold. **Note**: The thresholds can be configured using REST API. **Steps**: Reassess the reasons leading to NSSF receiving additional traffic. If this is unexpected, contact My Oracle Support. 1. Refer to alert to determine which service is receiving high traffic. It may be due to a sudden spike in traffic. For example: When one mated site goes down, the NFs move to the given site. 2. Check the service pod logs on Kibana to determine the reason for the errors. 3. If this is expected traffic, then the thresholds levels may be reevaluated as per the call rate and reconfigured as mentioned in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.2.3 OcnssfOverloadThresholdBreachedL3

Table 5-24 OcnssfOverloadThresholdBreachedL3

Field	Details
Description	'Overload Level of {{$labels.app_kubernetes_io_name}} service is L3'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, podname: {{$labels.kubernetes_pod_name}}: Overload Level of {{$labels.app_kubernetes_io_name}} service is L3'
Severity	Major
Condition	NSSF Services have breached their configured threshold of Level L3 for any of the aforementioned metrics. Thresholds are configured for CPU, svc_failure_count, svc_pending_count, and memory.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9018
Metric Used	load_level
Recommended Actions	The alert is cleared when the Ingress Traffic rate falls below the configured L3 threshold. **Note**: The thresholds can be configured using REST API. **Steps**: Reassess the reasons leading to NSSF receiving additional traffic. If this is unexpected, contact My Oracle Support. 1. Refer to alert to determine which service is receiving high traffic. It may be due to a sudden spike in traffic. For example: When one mated site goes down, the NFs move to the given site. 2. Check the service pod logs on Kibana to determine the reason for the errors. 3. If this is expected traffic, then the thresholds levels may be reevaluated as per the call rate and reconfigured as mentioned in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.2.4 OcnssfOverloadThresholdBreachedL4

Table 5-25 OcnssfOverloadThresholdBreachedL4

Field	Details
Description	'Overload Level of {{$labels.app_kubernetes_io_name}} service is L4'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, podname: {{$labels.kubernetes_pod_name}}: Overload Level of {{$labels.app_kubernetes_io_name}} service is L4'
Severity	Critical
Condition	NSSF Services have breached their configured threshold of Level L4 for any of the aforementioned metrics. Thresholds are configured for CPU, svc_failure_count, svc_pending_count, and memory.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9019
Metric Used	load_level

Table 5-25 (Cont.) OcnssfOverloadThresholdBreachedL4

Field	Details
Recommended Actions	The alert is cleared when the Ingress Traffic rate falls below the configured L4 threshold. **Note**: The thresholds can be configured using REST API. **Steps**: Reassess the reasons leading to NSSF receiving additional traffic. If this is unexpected, contact My Oracle Support. 1. Refer to alert to determine which service is receiving high traffic. It may be due to a sudden spike in traffic. For example: When one mated site goes down, the NFs move to the given site. 2. Check the service pod logs on Kibana to determine the reason for the errors. 3. If this is expected traffic, then the thresholds levels may be reevaluated as per the call rate and reconfigured as mentioned in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.2.5 OcnssfScpMarkedAsUnavailable

Table 5-26 OcnssfScpMarkedAsUnavailable

Field	Details
Description	'An SCP has been marked unavailable'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : One of the SCP has been marked unavailable'
Severity	Major
Condition	One of the SCPs has been marked unhealthy.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9020
Metric Used	'oc_egressgateway_peer_health_status'
Recommended Actions	This alert get cleared when unavailable SCPs become available.

5.2.6 OcnssfAllScpMarkedAsUnavailable

Table 5-27 OcnssfAllScpMarkedAsUnavailable

Field	Details
Description	'All SCPs have been marked unavailable'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : All SCPs have been marked as unavailable'
Severity	Critical
Condition	All SCPs have been marked unavailable.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9021
Metric Used	'oc_egressgateway_peer_count and oc_egressgateway_peer_available_count'

Table 5-27 (Cont.) OcnssfAllScpMarkedAsUnavailable

Field	Details
Recommended Actions	NF clears the critical alarm when at least one SCP peer in a peer set becomes available such that all other SCP or SEPP peers in the given peer set are still unavailable.

5.2.7 OcnssfTLSCertificateExpireMinor

Table 5-28 OcnssfTLSCertificateExpireMinor

Field	Details
Description	'TLS certificate to expire in 6 months'.
Summary	'namespace: {{$labels.namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : TLS certificate to expire in 6 months'
Severity	Minor
Condition	This alert is raised when the TLS certificate is about to expire in six months.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9028
Metric Used	security_cert_x509_expiration_seconds
Recommended Actions	The alert is cleared when the TLS certificate is renewed. For more information about certificate renewal, see "Creating Private Keys and Certificate " section in the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

5.2.8 OcnssfTLSCertificateExpireMajor

Table 5-29 OcnssfTLSCertificateExpireMajor

Field	Details
Description	'TLS certificate to expire in 3 months.'
Summary	'namespace: {{$labels.namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : TLS certificate to expire in 3 months'
Severity	Major
Condition	This alert is raised when the TLS certificate is about to expire in three months.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9029
Metric Used	security_cert_x509_expiration_seconds
Recommended Actions	The alert is cleared when the TLS certificate is renewed. For more information about certificate renewal, see "Creating Private Keys and Certificate " section in the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

5.2.9 OcnssfTLSCertificateExpireCritical

Table 5-30 OcnssfTLSCertificateExpireCritical

Field	Details
Description	'TLS certificate to expire in one month.'
Summary	'namespace: {{$labels.namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : TLS certificate to expire in 1 month'
Severity	Critical
Condition	This alert is raised when the TLS certificate is about to expire in one month.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9030
Metric Used	security_cert_x509_expiration_seconds
Recommended Actions	The alert is cleared when the TLS certificate is renewed. For more information about certificate renewal, see "Creating Private Keys and Certificate " section in the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

5.2.10 OcnssfNrfInstancesInDownStateMajor

Table 5-31 OcnssfNrfInstancesInDownStateMajor

Field	Details
Description	'When current operative status of any NRF Instance is unavailable/unhealthy'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : Few of the NRF instances are in unavailable state'
Severity	Major
Condition	When sum of the metric values of each NRF instance is greater than 0 but less than 3.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9032
Metric Used	nrfclient_nrf_operative_status
Recommended Actions	This alert is cleared when operative status of all the NRF Instances is available/ healthy. **Steps**: 1. Check the nrfclient_nrf_operative_status metric value of each NRF instance. 2. The instances for which the metric value is '0' are down. 3. Bring up the NRF instances that are down. 4. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support. **Note:** Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.

5.2.11 OcnssfAllNrfInstancesInDownStateCritical

Table 5-32 OcnssfAllNrfInstancesInDownStateCritical

Field	Details
Description	'When current operative status of all the NRF Instances is unavailable/unhealthy'
Summary	'kubernetes_namespace: {{$labels.kubernetes_namespace}}, timestamp: {{ with query "time()" }}{{ . | first | value | humanizeTimestamp }}{{ end }} : All the NRF instances are in unavailable state'
Severity	Critical
Condition	When sum of the metric values of each NRF instance is equal to 0.
OID	1.3.6.1.4.1.323.5.3.40.1.2.9031
Metric Used	nrfclient_nrf_operative_status
Recommended Actions	This alert is cleared when current operative status of atleast one NRF Instance is available/healthy. **Steps**: 1. Bring up at least one NRF Instance. 2. If the issue persists, capture all the outputs for the above steps and contact My Oracle Support. **Note:** Use Cloud Native Core Network Function Data Collector tool for capturing the logs. For more information, see *Oracle Communications Cloud Native Core, Network Function Data Collector User Guide*.