Oracle® Communications

Cloud Native Core, Network Slice Selection Function User Guide

Release 25.2.200
G40404-01
February 2026

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

G40404-01

Contents

5 Configuring NSSF using CNC Console

6 NSSF Metrics, KPIs, and Alerts

7 Appendix A- HTTP Response Codes

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 used in the document:

Table 1 Acronyms

Field	Description
3GPP	3rd Generation Partnership Project
5GC	5G Core Network
5GS	5G System
Allowed NSSAI	NSSAI provided by the serving PLMN during a registration procedure, indicating the S-NSSAIs values the UE could use in the serving PLMN for the current registration area.
AMF	Access and Mobility Management Function
API	Application Programming Interface
ASM	Aspen Service Mesh
CA	Certificate Authority
CLI	Command Line Interface
CN	Common Name
CNC	Cloud Native Core
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
Configured NSSAI	NSSAI provisioned in the UE applicable to one or more PLMNs.
CSP	Communication Service Provider
DB	Database
DNN	Data Network Name
EANAN	Empty Authorized NSSAI Availability Notification
EGW	Egress Gateway
eMBB	enhanced Mobile Broadband
EPC	Evolved Packet Core. It is a framework for providing converged voice and data on a 4G Long-Term Evolution (LTE) network.
EPS	Evolved Packet System. It is a Mobility Management (EMM) protocol that provides procedures for the control of mobility when the User Equipment (UE) uses the Evolved UMTS Terrestrial Radio Access Network (E-UTRAN). EPS is a combination of E-UTRAN, EPC and UE.
FQDN	Fully Qualified Domain Name
GR	Georedundant
HNS	Hierarchical Namespace
H-NSSF	Home NSSF
HPLMN	Home Public Land Mobile Network
HTTPS	Hypertext Transfer Protocol Secure
IE	Information Element
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
KPI	Key Performance Indicator
MIoT	Massive Internet of Things
MOS	My Oracle Support

Table 1 (Cont.) Acronyms

Field	Description
MPS	Messages Per Second
NDB	Network Data Broker
NF	Network Function
NFs	Network Functions
NRF	Oracle Communications Cloud Native Core, Network Repository Function
NS	Network Slice. A logical network that provides specific network capabilities and network characteristics.
NSI	Network Slice Instance
NSI ID	Network Slice Instance Identifier
NSS	Network Switching Subsystem
NSSAI	Network Slice Selection Assistance Information
NSSF	Oracle Communications Cloud Native Core, Network Slice Selection Function
OCCM	Oracle Communications Cloud Native Core, Certificate Management
OCI	Oracle Cloud Infrastructure
OHC	Oracle Help Center
OKE	Container Engine for Kubernetes
ONSSAI	Optimized NSSAI Availability Data Encoding feature
OSDC	Oracle Service Delivery Cloud
PDU	Protocol Data Unit
PEI	Permanent Equipment Identifier
PLMN	Public Land Mobile Network
RAN	Radio Access Network
Requested NSSAI	NSSAI provided by the UE to the serving PLMN during registration.
Restricted S-NSSAI	This is an information element (IE) that contains restricted S-NSSAI(s) per PLMN for a Tracking Area(TA). If the restricted SNssai is not present, no restricted S-NSSAI is applicable to the TA. If present, this IE (restrictedSnssai) is included only by the NSSF.
SBA	Service Based Architecture
SBI	Service Based Interface
SCP	Oracle Communications Cloud Native Core, Service Communication Proxy
SD	Slice Differentiator
SEPP	Oracle Communications Cloud Native Core, Security Edge Protection Proxy
SMF	Session Management Function
S-NSSAI	Single Network Slice Selection Assistance Information
SSC	Session and Service Continuity
SST	Slice or Service type
Subscribed S-NSSAI	5G uses this as a default when the UE does not send a Requested NSSAI
SUMOD	Subscription Modification
SUPI	Subscription Permanent Identifier
SVC	Services
TA	Tracking Area
TAC	Tracking Area Code
TAI	Tracking Area Identifier
TLS	Transport Layer Security
UDM	Unified Data Management

Table 1 (Cont.) Acronyms

Field	Description
UDR	Oracle Communications Cloud Native Core, Unified Data Repository
UE	User Equipment
URI	Uniform Resource Identifier
URLLC	Ultra-Reliable Low Latency Communications
V-NSSF	Visited NSSF
VPLMN	Visited Public Land Mobile Network

What's New in This Guide

This section lists the documentation updates for release 25.2.2xx.

Release 25.2.200- G40404-01, February 2026

- **Feature Updates:**
  - **New Features:**
    - NRF Client Retry and Health Check: The NRF Client Health Check continuously and automatically assesses the health of NRF instances, ensuring high service availability and seamless failover, especially in georedundant environments. It features configurable intervals, intelligent health status updates, and a retry mechanism to minimize service disruption and manage network failures.
      - No new metrics have been added for this feature. However, it uses the existing "`nrfclient_nrf_operative_status`" metric from the NSSF Common metrics section.
      - No new alerts have been added for this feature. However, it uses the following Application Level Alerts:
        - OcnssfAllNrfInstancesInDownStateCritical
        - OcnssfNrfInstancesInDownStateMajor
    - Support for TLSv1.3 on Internal API Communication: Improved security for inter-service communication, reinforced by new metrics to monitor TLS certificate statuses and address resolution failures.
      - Added the following metrics in the NSSF Metrics section:
        - `oc_ingressgateway_ip_addresses_fetch_failure`
        - `oc_egressgateway_ip_addresses_fetch_failure`
        - `oc_certificatemanagement_tls_certificate_info`
        - `oc_certificatemanagement_tls_secret_status`
        - `oc_certificatemanagement_tls_certs_reload_failure`
        - `cgiu_jetty_ip_address_fetch_failure`
    - Support for Dual Stack: Added comprehensive handling for both IPv4 and IPv6, with new monitoring metrics to track ingress and egress IP types and rejections.
      - Added the following metrics in the NSSF Metrics section:
        - `oc_ingressgateway_incoming_ip_type`
        - `oc_ingressgateway_outgoing_ip_type`
        - `c_egressgateway_incoming_ip_type`
        - `oc_egressgateway_outgoing_ip_type`
        - `oc_egressgateway_dual stack_ip_rejected_total`
  - **Architectural Enhancements:**
    - The NSSF's core architecture has been redesigned to implement an in-memory, pod-level cache, reducing latency and dramatically boosting throughput.

NsSelection throughput has increased from 10k to 80k TPS (an 8x improvement), with CPU consumption scaling from 60 to 120 cores (a 2x increase).

    - The architecture now supports large-volume messages in NsAvailability PUT operations, allowing all TAIs to be transmitted in a single message.
    - The NSSF schema and REST APIs have been overhauled to support in-memory caching. Corresponding REST API configuration options and details are now available directly from the CNC Console.
  - **Deprecations and Removals (Driven by Architectural/Business Requirements)**
    - *Autopopulation of Configuration Using NRF Content*: Removed due to the introduction of new Candidate AMF selection logic and redundancy in AMF set configuration with updated functional behavior.
    - *Handover from EPS to 5G*: Deprecated, as call transfer use cases are now handled through independent selection requests per updated customer requirements.
    - *Optimized NSSAI Availability Data Encoding and TAI Range*: Removed because TAI ranges are no longer viable, given non-incremental TAC assignments reported by customers.
    - *Protection from Distributed Denial-of-Service (DDoS) Attack through Rate Limiting and Ingress Gateway Pod Protection*: Both features are now obsolete with the introduction of pod-level rate limiting and updated protection mechanisms in the gateway's latest releases.
    - *Time of the Day Based Network Slice Instance Selection:* Deprecated from this release.
    - *IPv6 Support:* Removed as a standalone feature and superseded by the "Support for Dual Stack feature.
    - *Ingress Gateway Pod Protection:* Removed as it is now considered obsolete due to the introduction of pod-level rate limiting in newer gateway releases.
    - Deprecated metrics and alerts have been cleaned up for a more focused monitoring and troubleshooting experience.
    - Details of several existing alerts and metrics have been updated and outdated alerts and metrics have been removed.
- **General Updates:**
  - Updated release number to 25.2.200 throughout the document.
  - Added a feature description section for Support for cnDBTier APIs in CNC Console. The corresponding UI details were also updated for it in the Configuring NSSF using CNC Console section.
  - Configuring NSSF using CNC Console: New and redesigned UIs simplify CNC-configured NSSF options, system settings, and profile management, while outdated UIs have been removed to streamline user interaction.
  - Updated Table 4-20 in Supports Compression Using Accept-Encoding or Content-Encoding gzip feature section to remove the unsupported POST method from the response.
  - Added new error scenarios for Subscription with HTTP Patch (Option ADD in the Subscription Modification Feature section.
  - Added the following new NSSF Metrics:
    - `ocnssf_indirect_communication_request_rx_total`

* ocnssf_indirect_communication_response_tx_success_total
* ocnssf_indirect_communication_response_tx_failure_total
* ocnssf_subscription_request_rx
* ocnssf_subscription_response_tx_success_total
* ocnssf_subscription_response_tx_failure_total
* ocnssf_nssaiavailability_error_tx_total
* ocnssf_nssaiavailability_options_tx_status_unsupportedmediatype_total
* ocnssf_nsavailability_unsupported_plmn_total
* ocnssf_state_data_write_error
* stale_records_computation_DQD_amf_tai_slice_availability_data_seconds_count
* stale_records_computation_JL_amf_tai_slice_availability_data_seconds_count
* total_stale_records_APD_amf_tai_slice_availability_data_seconds_count
* count_amf_tai_slice_availability_data_stale_records_deleted_total
* stale_records_computation_DQD_nssai_subscriptions_seconds_count
* stale_records_computation_JL_nssai_subscriptions_seconds_count
* total_stale_records_APD_nssai_subscriptions_seconds_count
* count_nssai_subscriptions_stale_records_deleted_total

1 Introduction

1.1 Overview

This section describes the role of Oracle Communications Network Slice Selection Function (NSSF) in the 5G Service Based Architecture (SBA).

Network slices enable the users to select customized networks with different functionalities (such as mobility) and performance requirements (such as latency, availability and reliability). Network slices differ in features supported and network function optimizations. In such cases, network slices may have different S-NSSAIs with different slice and service types. The user can deploy instances of multiple network slices delivering the same features but for different groups of User Equipments (UEs). These instances deliver different committed services as they are dedicated to a customer, the network slices may have different S-NSSAIs with the same slice or service type but different slice differentiators. The NSSF fulfills the requirement for determining the individual network function pertaining to a slice.

> **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.

NSSF is a functional element that supports the following functionalities:

- NSSF enables the Access and Mobility Management Function (AMF) to perform initial registration and Protocol Data Unit (PDU) session establishment.
- AMF can retrieve NRF, NSI ID, and target AMFs as part of UE initial registration and PDU establishment procedure.
- NSSF uses an NF Service Consumer (AMF) to update the S-NSSAI(s) that AMF supports and notifies of any changes in the status.
- NSSF selects the network slicing instance (NSI) and determines the authorized Network Slice Selection Assistance Information (NSSAIs) and AMF to serve the UE.
- NSSF interaction with NRF allows retrieving specific NF services to be used for registration request.

NSSF provides the following information when queried by the AMF:

- Allowed NSSAIs
- Configured NSSAIs
- Restricted NSSAIs
- Candidate AMF List (in case of registration)
- Network Slice instance ID (for PDU session establishment)
- Slice-level NRF information (for PDU Connectivity)

NSSF supports the above functions through the following NSSF services:

- **NSSelection service** (*Nnssf_NSSelection*): This service is used by an NF Service Consumer (AMF) to retrieve the information related to network slice. It enables network slice selection in the serving Home Public Land Mobile Network (HPLMN).
- **NSAvailability Service** (*Nnssf_NSAvailability*): This service stores and maintains list of supported S-NSSAIs per TA. It allows NF service Consumer (AMF) to update and subscribe the above data and get notifications for any addition or deletion of supported S-NSSAIs.

NSSF Availability

Oracle Communications Cloud Native Core, Network Slice Selection Function (NSSF) availability is dependent on many factors. NSSF applications are designed to achieve 99.999% availability, according to the applicable Telecommunications Industry Association TL9000 standards, with the following deployment requirements:

- Deploy on a Cloud Native Environment with at least 99.999% Availability.
- Deploy with n + k application redundancy, where k is greater than or equal to one.
- Maintain production software within n-3 software releases, where n is the current general availability release.
- Apply bug fixes, critical patches, and configuration recommendations provided by Oracle promptly.
- Maintain fault recovery procedures external to the applications for the reconstruction of lost or altered files, data, programs, or Cloud Native environment.
- Install, configure, operate, and maintain NSSF as per Oracle's applicable installation, operation, administration, and maintenance specifications.
- Maintain an active support contract and provide access to the deployed NSSF and your personnel to assist Oracle in addressing any outage.

NSSF availability is measured for each calendar year and is calculated as follows:

Table 1-1 Measuring NSSF Availability

Availability	Description
Planned Product Availability	(Product available time in each month) less (Excluded Time (defined below) in each month).
Actual Product Availability	(Planned Product Availability) less (any Unscheduled Outage).
Product Availability Level	(Actual Product Availability across all Production instances divided by Planned Product Availability across all Production instances) x 100.

Note

Excluded Time means:

- Scheduled maintenance time.
- Lack of power or backhaul connectivity, except to the extent that such lack of backhaul connectivity was caused directly by the CNC NF.
- Hardware failure.
- Issues arising out of configuration errors or omissions.
- Failures caused by third-party equipment or software not provided by Oracle.
- Occurrence of any event under Force Majeure.
- Any time associated with failure to maintain the recommended architecture and redundancy model requirements above.

1.2 References

- *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Network Slice Selection Function Network Impact Report*
- *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*
- *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Automated Testing Suite Guide*
- *Oracle Communications Cloud Native Core, cnDBTier User Guide*
- *Oracle Communications Cloud Native Core, Data Collector User Guide*

2
NSSF Supported Services

This chapter includes information about the services supported by NSSF.

> **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.

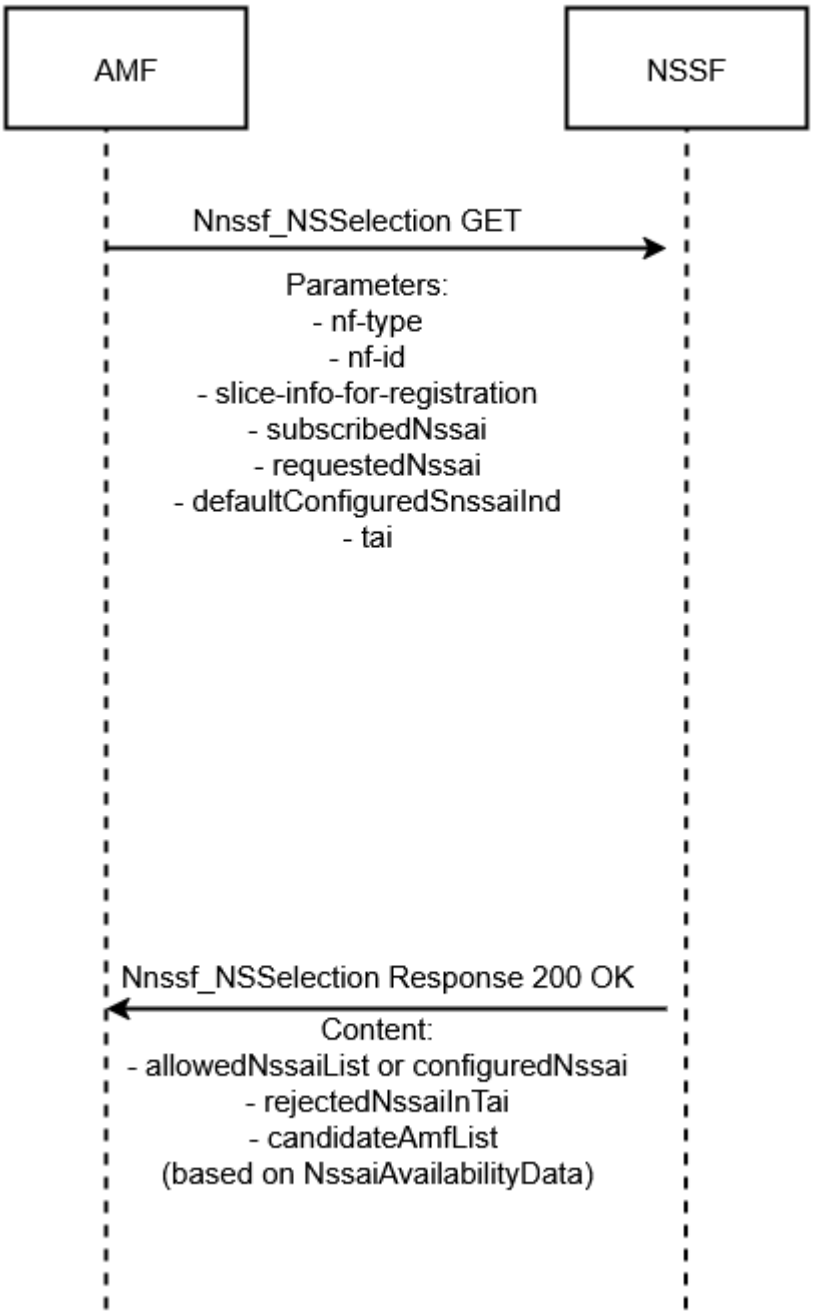
2.1 Network Slice Selection Service

The Network Slice Selection service is identified by the service operation name, *Nnssf_NSSelection*. This service supports the GET request during the following procedures by UE:

Initial Registration:

When NSSF is able to find authorized network slice information for the requested network slice, the response includes a payload containing at least the Allowed NSSAI, target AMF Set, or the list of candidate AMF(s).

Following diagram illustrates the procedure of initial registration:



- The AMF sends a GET request to the NSSF.

  The AMF GET request must include:

  – Subscribed S-NSSAIs (with an indication if marked as default S-NSSAI)
  – Any Allowed NSSAI

  The query parameters may also contain:

  – Requested NSSAI
  – Mapping of requested NSSAI to configured NSSAI for the HPLMN
  – Mapping to the Configured NSSAI for the HPLMN

  - PLMN ID of the Subscription Permanent Identifier (SUPI)
  - UE's current Tracking Area
  - NF type of the NF service consumer
  - AMF ID
- Based on the query parameters mentioned above, local configuration, and locally available information, including Radio Access Network (RAN) capabilities obtained by the current Tracking Area for the UE, NSSF does the following:
  - It selects the Network Slice instance(s) to serve the UE. When multiple Network Slice instances in the UE's Tracking Areas are able to serve a given S-NSSAI (based on operator's configuration), NSSF selects one slice to serve the UE, or defer the selection of the Network Slice instance until a NF or service within the Network Slice instance needs to be selected.
  - It determines the target AMF set to be used to serve the UE or based on configuration, the list of candidate AMF(s), possibly after querying the NRF.
  - The AuthorizedNetworkSliceInfo response for UE-registration must mandatorily include both AllowedNSSAI and Candidate AMF list or target amfset. The AllowedNSSAI is computed by taking into account the input request and applying operator policies as specified in 3GPP Spec 29.531 Release 15.5.
  - NSSF calculates ConfiguredNSSAI by determining the intersection of S-NSSAI(s) in ConfiguredNSSAI for the PLMN (operator configured) and S-NSSAI(s) in SubscribedNSSAI (from the message indicating SubscribedNSSAI by the UE).
  - It determines the Allowed NSSAI(s) for the applicable Access Type(s), taking also into account the availability of the Network Slice instances that are able to serve the S-NSSAI(s) in the Allowed NSSAI and the current UE's tracking areas.
  - Based on operator configuration, the NSSF determines the NRF(s) to be used to select NFs or services within the selected Network Slice instance(s).
- When the NSSF locates the authorized network slice information for the requested network, NSSF sends Discovery Request for AMF to NRF.
- The NRF responds with the list of all candidate AMFs to NSSF.
- The NSSF returns to the current AMF the Allowed NSSAI for the applicable Access Type(s), the target AMF Set, or the list of candidate AMF(s) based on configuration.
  - NSSF returns the NRF(s) to be used to select NFs/services within the selected Network Slice instance(s) and the NRF to be used to determine the list of candidate AMF(s) from the AMF Set.
  - NSSF returns NSI ID(s) to be associated to the Network Slice instance(s) corresponding to certain S-NSSAIs.
  - NSSF also returns the rejected S-NSSAI(s) and the Configured NSSAI for the Serving PLMN.
- Candidate AMF selection by NSSF:
  - In response to the Initial Registration request, NSSF responds with AMFs that support the Authorized NSSAI in the specified TAI. NSSF prioritizes runtime data, and if a suitable match is not found, it falls back to operator-configured data. Runtime data consists of NsAvailabilityData sent by the AMF.
  - This approach is chosen to ensure that NSSF responds with consideration of dynamic data. In scenarios where sufficient data is not present (for example, AMFs have not sent an Availability Update), NSSF relies on operator-configured data.

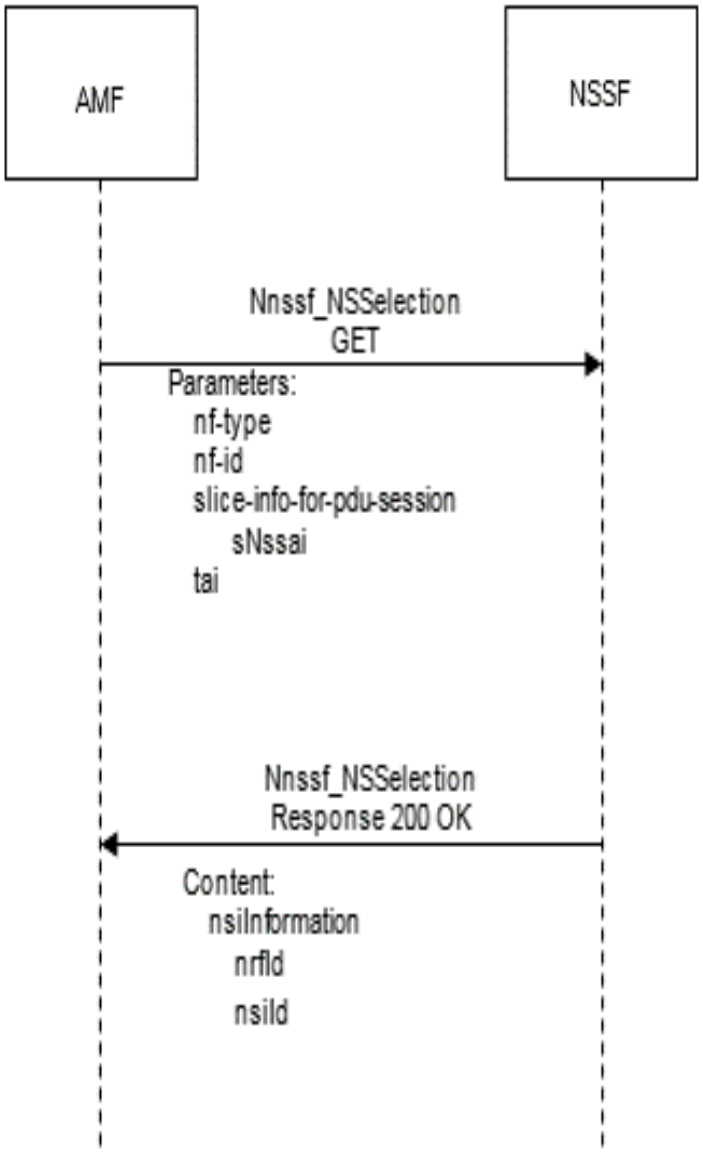
- This ensures that a response is provided when NSSF is just initialized and does not have Availability Data. The recommendation is that AMFs should have a configuration to update NsAvailability information with NSSF, so that NSSF can respond based on dynamic data.

PDU Session Establishment:

When the NSSF receives a PDU-Session establishment request from the NF consumer, it determines the network slice that can serve the requested S-NSSAI based on the user configured policies, and responds with the URL of the NRF that manages the Slice and Slice ID of the matching Network slice computed.

The PDU session establishment in a Network Slice to a Data Network (DN) allows data transmission in a Network Slice. A PDU Session is associated with a S-NSSAI and a Data Network Name (DNN). Following diagram illustrates the procedure of PDU Session Establishment:

Figure 2-1 PDU Session Establishment



The following is performed for PDU Session Establishment:

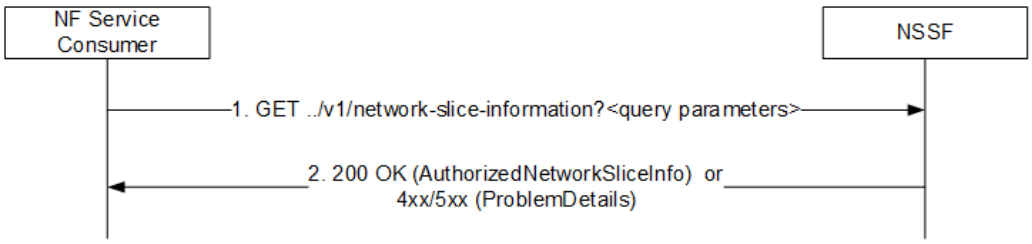
- If the AMF is not able to determine the appropriate NRF to query for the S-NSSAI provided by the UE, the AMF sends a GET request to the NSSF. The AMF queries the NSSF with this specific S-NSSAI, the NF type of the NF service consumer, Requester ID, PLMN ID of the SUPI, and the location information.
- The NSSF determines and returns the appropriate NRF to be used for selecting NFs or services within the selected Network Slice instance. The NSSF may also return an NSI ID identifying the Network Slice instance to use for this S-NSSAI.
  When a PDU Session for a given S-NSSAI is established using a specific Network Slice instance, the cloud native provides the RAN with S-NSSAI corresponding to this Network Slice instance, which enables the RAN to perform access specific functions.

UE-Config-Update:

When the UDM updates the Subscribed S-NSSAI(s) to the serving AMF, based on configuration in the AMF, the NSSF determines the mapping of the Configured NSSAI for the serving PLMN and Allowed NSSAI to the Subscribed S-NSSAI(s).

Following diagram illustrates the procedure of UE-Config-Update:

Figure 2-2 UE-Config-Update



The following is performed for UE-Config-Update:

- The AMF sends a UE-Config-Update (GET) request to NSSF. NSSF checks and validates the Subscribed S-NSSAI(s), Requested S-NSSAI(s), PLMN ID of the SUPI, TAI, NF type, and NF instance ID. If message is valid, NSSF searches for Allowed S-NSSAI list based on policy configuration and input parameters.
- NSSF responds with "200 OK with AuthorizedNetworkSliceInfo" if it finds a match.
- The AuthorizedNetworkSliceInfo response for UE-Config-Update must mandatorily include both AllowedNSSAI and ConfiguredNSSAI. The AllowedNSSAI is computed by taking into account the input request and applying operator policies as specified in 3GPP Spec 29.531 Release 15.5.
- NSSF calculates ConfiguredNSSAI by determining the intersection of S-NSSAI(s) in ConfiguredNSSAI for the PLMN (operator configured) and S-NSSAI(s) in SubscribedNSSAI (from the message indicating SubscribedNSSAI by the UE).
- NSSF responds with error code if it finds incorrect parameter validation.

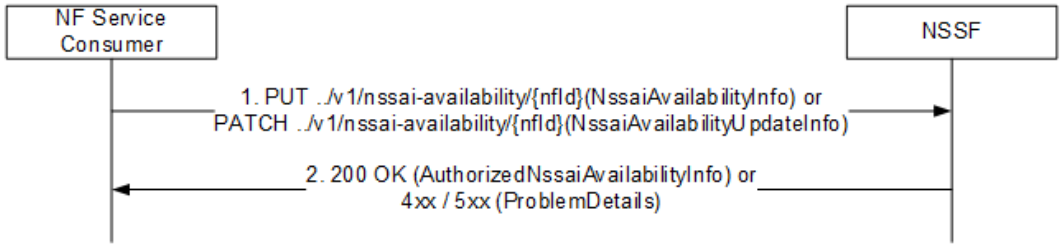
2.2 NSSAI Availability Service

The NSSAI Availability service is identified by the service name, `Nnssf_NSSAIAvailability`. The following operations are defined for this service:

- Update Service Operation
- Subscribe Service Operation
- Unsubscribe Service Operation
- Notify Service Operation
- Delete Service Operation

1. **Update Service Operation**
   The AMF uses this operation to update the NSSF with the supported S-NSSAI(s) on a per TA basis and to get informed on the S-NSSAIs available per TA (unrestricted) and the restricted S-NSSAI(s) per PLMN in that TA in the serving PLMN of the UE.

Figure 2-3 Update the S-NSSAIs the AMF supports per TA



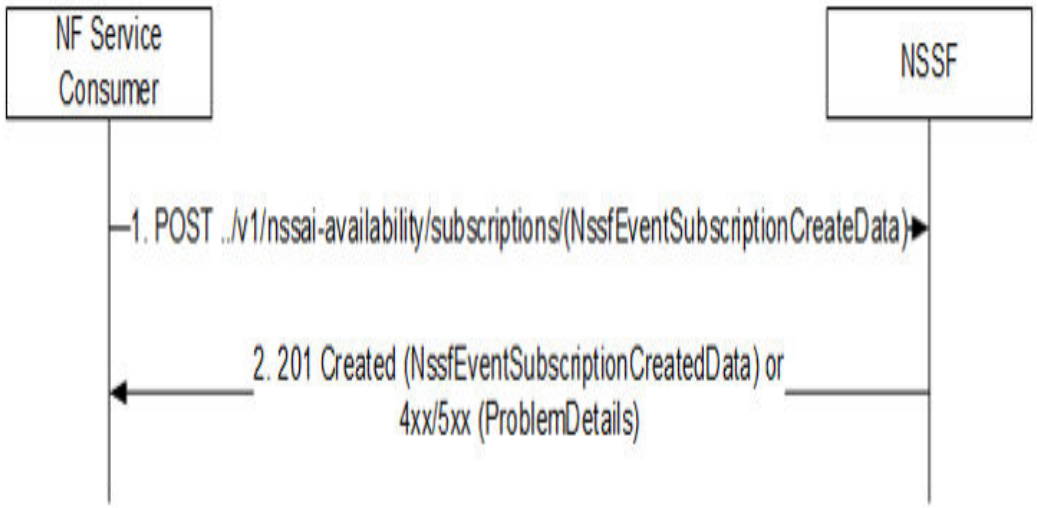
- The NF service consumer (for example, AMF) sends a `PUT` request to NSSF with NSSAI availability information, identified by `{nfId}`, using `NssaiAvailabilityInfo`. The message contains a list of S-NSSAIs supported by the AMF on a per-TA basis.
- NSSF checks (from operator configuration, specifically the PLMN Config managed object). If any S-NSSAI supported by the AMF is not configured (not part of `ConfiguredNSSAI`), NSSF responds with `403 Forbidden`.
  Since NSSF is the source for slices and S-NSSAIs in the network, if an NF (AMF) claims that it supports an S-NSSAI not known to NSSF (i.e., not present in `ConfiguredNSSAI` for the PLMN), NSSF responds with `403 Forbidden`, as the NF (AMF in this scenario) is forbidden to support an S-NSSAI not configured for that PLMN.
- NSSF checks (from operator system options configuration) that PLMNs are preconfigured in the NSSF System Option. If any PLMN is found to be not configured, NSSF responds with `403 Forbidden`.
  In case an NF (AMF), via `NssaiAvailabilityData`, claims that it supports a PLMN not known to NSSF (Central NF), NSSF rejects the request with `403 Forbidden`, since requests for unknown PLMNs cannot be accepted.
- NSSF checks (from operator configuration) if the S-NSSAIs are authorized in the TAI and responds with the S-NSSAIs that are authorized by NSSF and supported by the AMF for each TAI.
- If none of the S-NSSAIs in the request are authorized by NSSF (i.e., if all supported S-NSSAIs from the AMF in all TAIs are configured but restricted either at the PLMN level or at the TAI level), NSSF responds with a `204 No Content` message.
- NSSF supports `HTTP PATCH` for `NssaiAvailability` update.
- Upon receiving a `PUT` or `PATCH` message, NSSF stores or updates the list in the session database.

- In scenarios where all TAI-S-NSSAIs supported by the AMF are configured and some or all S-NSSAIs are authorized, NSSF responds with `200 OK` and `AuthorizedNssaiAvailabilityData` (the TAI-S-NSSAI mapping authorized by NSSF).

2. **Subscribe Service Operation**

   The Subscribe operation is used by NF Service Consumer (AMF) to get the notifications for any change in NSSAI availability information.

Figure 2-4 Subscription creation

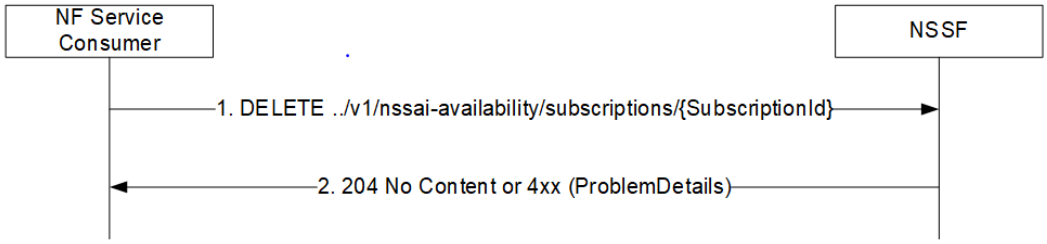


- AMF sends a `POST` request to NSSF with a notification URL and a list of `TAIs` as JSON body.
- NSSF stores the subscription request and responds with the list of authorized `S-NSSAI(s)` per `TAI` in the request.
- NSSF also returns a `subscription-id` and `expiry` (duration up to which NSSF sends notifications for any change in the status of `S-NSSAI` for subscribed `TAI(s)`).
- The expiry duration is optional:
  - If not provided, NSSF sets it to `maxExpiryDuration` (Helm configuration)
  - If greater than `maxExpiryDuration`, NSSF resets it to `maxExpiryDuration`
  - If less than `minExpiryDuration` (Helm parameter), the request is rejected
  - If a remove operation is received on the Expires header, NSSF refreshes the subscription to ensure continuity and avoid rejection caused by unexpected or unwarranted NF consumer behavior
- In case the `SUMOD` feature is enabled at NSSF and the request comes with `supportedFeatures` with `SUMOD` bit as true, NSSF accepts a `PATCH` request on the subscription.

3. **Unsubscribe Service Operation**

   The Unsubscribe service operation is used by AMF to unsubscribe to a notification of any previously subscribed changes to the NSSAI availability information.

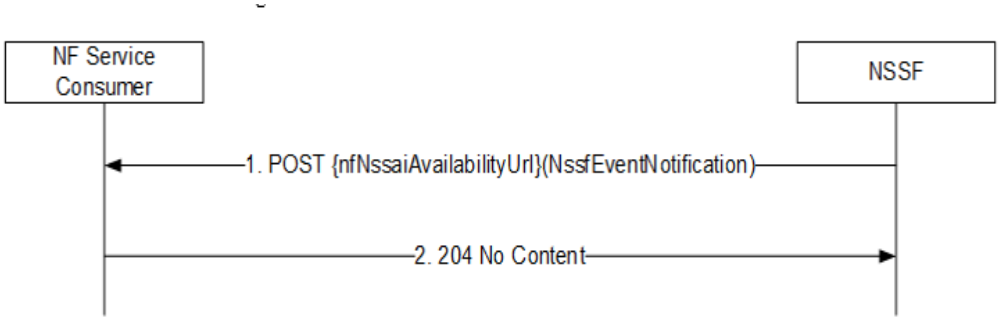
Figure 2-5 Unsubscribe a Subscription



- AMF sends a `DELETE` request to NSSF with `subscription-id`.
- NSSF checks for an active subscription with the ID and, if found, deletes the subscription and responds with the message `204`.
- If a subscription is not found with `subscription-id`, NSSF responds with `404 Not Found`.

4. **Notify Service Operation**
   The Notify service operation is used by the NSSF to update the AMF with any change in status, on a per TA basis, of the S-NSSAIs available per TA (unrestricted) and the S-NSSAIs restricted per PLMN in that TA in the serving PLMN of the UE.

Figure 2-6 Update the AMF with any S-NSSAI restricted per TA

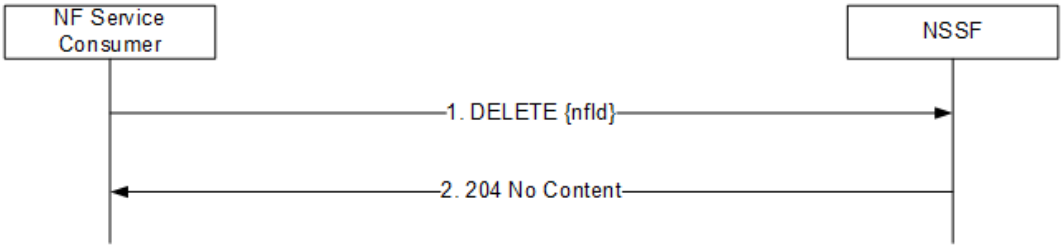


- NSSF sends notification to subscribed `AMF` when one or more of the following conditions are true
- If auto population of content based on `AMF` update is enabled:
  - An `S-NSSAI` has been barred or unbarred at `PLMN` level, for `PLMNs` associated with the subscribed `TAIs`
  - An `S-NSSAI` has been barred or unbarred for one or more subscribed `TAIs`
- If auto population of content based on `AMF` update is disabled:
  - An `S-NSSAI` has been added to or removed from the supported `S-NSSAI` for one or more subscribed `TAIs`

5. **Delete Service Operation**

   The AMF uses this operation to delete the NSSAI Availability information stored for that AMF in the NSSF.

Figure 2-7 Delete the NSSAI Availability Information at NSSF

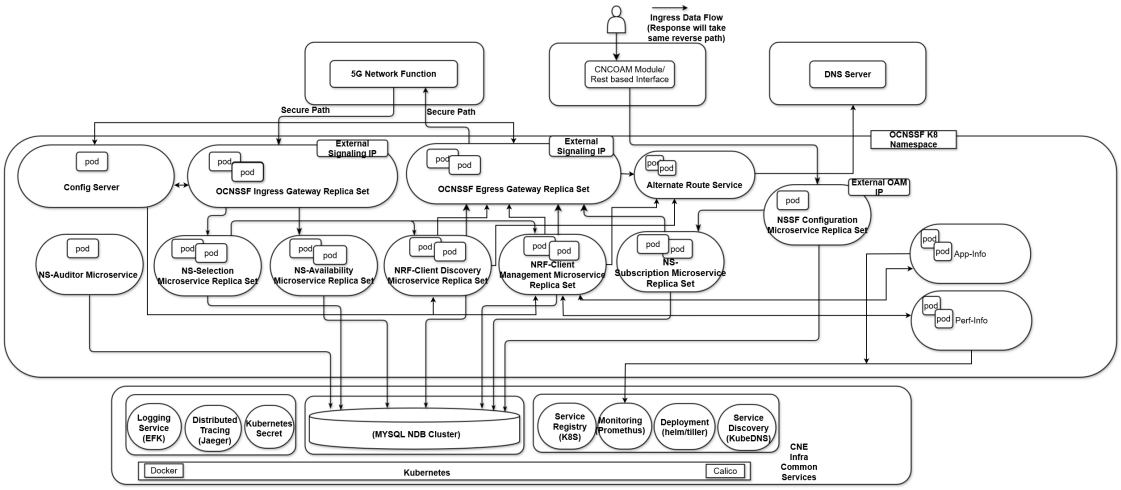


- The NF service consumer (For example: AMF) sends a DELETE request to NSSF with `nfId`.
- The NSSF searches in session database for the NSAvailability data corresponding to `nfId` and deletes them.

3
NSSF Architecture

NSSF comprises of various microservices deployed in Kubernetes based Cloud Native Environment (CNE), for example: Oracle Communications Cloud Native Core, Cloud Native Environment (CNE). CNE provides some common services like logs, metrics data collection, analysis, graphs, and charts visualization, etc. The microservices integrate with these common services and provide them with the necessary data.

The following diagram describes the overall architecture of the NSSF:



The architecture has the following components:

1. **NSSelection Service**
   The Network Slice Selection service is identified by the service operation name, Nnssf_NSSelection. This service supports the GET request during the following procedures by UE:

   a. **Initial Registration:**
      When the NSSF is able to find authorized network slice information for the requested network slice, the response body includes a payload body containing at least the Allowed NSSAI, target AMF Set, or the list of candidate AMF(s).

      - The AMF sends a GET request to the NSSF. The AMF GET request must include:
        - Subscribed S-NSSAIs (with an indication if marked as default S-NSSAI)
        - Any Allowed NSSAI
      - The query parameters may also contain:
        - Requested NSSAI
        - Mapping of requested NSSAI to configured NSSAI for the HPLMN
        - Mapping to the Configured NSSAI for the HPLMN
        - PLMN ID of the Subscription Permanent Identifier (SUPI)
        - UE's current Tracking Area

  - NF type of the NF service consumer
  - AMF ID
- Based on the query parameters mentioned above, local configuration, and other locally available information including Radio Access Network (RAN) capabilities made available by the current Tracking Area for the UE, the NSSF does the following:
  - It selects the Network Slice instance(s) to serve the UE. When multiple Network Slice instances in the UE's Tracking Areas are able to serve a given S-NSSAI, based on operator's configuration, the NSSF may select one of them to serve the UE, or the NSSF may defer the selection of the Network Slice instance until a NF or service within the Network Slice instance needs to be selected.
  - It determines the target AMF set to be used to serve the UE or based on configuration, the list of candidate AMF(s), possibly after querying the NRF.
  - The AuthorizedNetworkSliceInfo response for UE-registration must mandatory include both AllowedNSSAI and Candidate AMF list or target amfset. The AllowedNSSAI is computed by taking into account the input request and applying operator policies as specified in 3GPP Spec 29.531 Release 15.5.
  - NSSF calculates ConfiguredNSSAI by determining the intersection of S-NSSAI(s) in ConfiguredNSSAI for the PLMN (operator configured) and S-NSSAI(s) in SubscribedNSSAI (from the message indicating SubscribedNSSAI by the UE).
  - It determines the Allowed NSSAI(s) for the applicable Access Type(s), taking also into account the availability of the Network Slice instances that are able to serve the S-NSSAI(s) in the Allowed NSSAI in the current UE's tracking areas.
  - Based on operator configuration, the NSSF may determine the NRF(s) to be used to select NFs or services within the selected Network Slice instance(s).
- When the NSSF is able to find authorized network slice information for the requested network, NSSF sends Discovery Request for AMF to NRF.
- The NRF responds with list of candidate AMFs to NSSF.
- The NSSF returns to the current AMF the Allowed NSSAI for the applicable Access Type(s), the target AMF Set, or, based on configuration, the list of candidate AMF(s).
  - NSSF returns the NRF(s) to be used to select NFs/services within the selected Network Slice instance(s) and the NRF to be used to determine the list of candidate AMF(s) from the AMF Set.
  - NSSF returns NSI ID(s) to be associated to the Network Slice instance(s) corresponding to certain S-NSSAIs.
  - NSSF also returns the rejected S-NSSAI(s) and the Configured NSSAI for the Serving PLMN.

b. **PDU Session Establishment:**
When the NSSF receives PDU-Session establishment request from the NF consumer, NSSF determines the network slice which can serve the requested S-NSSAI, based on the user configured policies, and responds with the URL of NRF which manages to the Slice and/or Slice ID of the matching Network slice computed.

The PDU session establishment in a Network Slice to a Data Network (DN) allows data transmission in a Network Slice. A PDU Session is associated with a S-NSSAI and a Data Network Name (DNN).

The following is performed for PDU Session Establishment:

- If the AMF is not able to determine the appropriate NRF to query for the S-NSSAI provided by the UE, the AMF sends a GET request to the NSSF. The AMF queries the NSSF with this specific S-NSSAI, the NF type of the NF service consumer, Requester ID, PLMN ID of the SUPI, and the location information.
- The NSSF determines and returns the appropriate NRF to be used to select NFs or services within the selected Network Slice instance. The NSSF may also return an NSI ID identifying the Network Slice instance to use for this S-NSSAI.
  When a PDU Session for a given S-NSSAI is established using a specific Network Slice instance, the cloud native provides to the RAN the S-NSSAI corresponding to this Network Slice instance to enable the RAN to perform access specific functions.

c. **UE-Config-Update**:
   When the UDM updates the Subscribed S-NSSAI(s) to the serving AMF, based on configuration in this AMF, the NSSF determines the mapping of the Configured NSSAI for the serving PLMN and Allowed NSSAI to the Subscribed S-NSSAI(s).

   The following is performed for UE-Config-Update:

   - The AMF sends a UE-Config-Update (GET) request to NSSF. NSSF checks and validates the Subscribed S-NSSAI(s), Requested S-NSSAI(s), PLMN ID of the SUPI, TAI, NF type, and NF instance ID. If message is valid, NSSF searches for Allowed S-NSSAI list based on policy configuration and input parameters.
   - NSSF responds with 200 OK with AuthorizedNetworkSliceInfo in case NSSF finds a match.
   - The AuthorizedNetworkSliceInfo response for UE-Config-Update must mandatorily include both AllowedNSSAI and ConfiguredNSSAI. The AllowedNSSAI is computed by taking into account the input request and applying operator policies as specified in 3GPP Spec 29.531 Release 15.5.
   - NSSF calculates ConfiguredNSSAI by determining the intersection of S-NSSAI(s) in ConfiguredNSSAI for the PLMN (operator configured) and S-NSSAI(s) in SubscribedNSSAI (from the message indicating SubscribedNSSAI by the UE).
   - NSSF responds with error code in case of incorrect parameter validation.

2. **NS Availability Service**
   This microservice supports NSAvailability service of NSSF as per 29.531. This microservice stores subscriptions and AMF data.

   The NSSAI Availability service is identified by the service name, Nnssf_NSSAIAvailability. For the Nnssf_NSSAIAvailability service the following service operations are defined:

   - Update Service Operation
   - Subscribe Service Operation
   - Unsubscribe Service Operation
   - Delete Service Operation

3. **NS Subscription Service**

   - AMF sends a `POST` request to NSSF with a notification URL and a list of `TAIs` as JSON body.
   - NSSF stores the subscription request and responds with the list of authorized `S-NSSAI(s)` per `TAI` in the request.

- NSSF also returns a `subscription-id` and `expiry` (duration up to which NSSF sends notifications for any change in the status of `S-NSSAI` for subscribed `TAI(s)`).
- The expiry duration is optional:
  - If not provided, NSSF sets it to `maxExpiryDuration` (Helm configuration)
  - If greater than `maxExpiryDuration`, NSSF resets it to `maxExpiryDuration`
  - If less than `minExpiryDuration` (Helm parameter), the request is rejected
  - If a remove operation is received on the Expires header, NSSF refreshes the subscription to ensure continuity and avoid rejection caused by unexpected or unwarranted NF consumer behavior
- In case the `SUMOD` feature is enabled at NSSF and the request comes with `supportedFeatures` with `SUMOD` bit as true, NSSF accepts a `PATCH` request on the subscription.

4. **NS Auditor Service**
   This microservice is a timed auditor that removes stale and deleted records from NSSF.

   What is a stale record?

   In georedundant scenarios, tables in State Database (stateDB) maintain a column siteId, which identifies owner site of that record. There could be georedundancy scenarios when similar records can be owned by two sites, where the older record is termed as the stale record.

   NS Auditor is used in georedundancy scenarios where subscription is owned by one site, but the Patch is received on other site. This leads to creation of two records for same subscription owned by each site. Ns-Auditor detects this and removes the old stale record to ensure subscription is owned at a singe site only.

   For example:

   - Site-1 receives Subscription POST.
   - Site-1 creates a record, rec-1, for subscription with owner as site-1.
   - If AMF gets disconnected with the site-1, site-1 goes down, or SCP makes a routing decision based on congestion, the subscription PATCH is received on site-2.
   - Site-2 creates a new record, rec-2, for subscription with new owner as site-2.
   - Now, rec-2 for site-2 is same as rec-1 for site1.
   - NS Auditor detects this and deletes rec-1.
   - Site-2 becomes the owner of the subscription, and receives the latest patch.

   What is a Deleted Record?

   All database tables (`stateDB` and `provisionDB`) maintain a `state` column that indicates the current status of each record. The allowed values are `ACTIVE` and `DELETED`. When an operator or the AMF issues a delete request to the services, the corresponding record is updated and marked as `DELETED`. Records in the `DELETED` state are termed deleted records.

   The default auditor time to remove deleted records is every 12 hours (every 2 seconds).

   For example:

   a. When the operator sends a delete PLMN configuration request to the `NsConfig` service with a given `plmnId`, the service marks the corresponding PLMN record as `DELETED`. This same behavior applies to all managed objects.

b. When the AMF sends a delete availability request to the `NsAvailability` service with an `amfId`, the service marks the corresponding AMF record as `DELETED`.

c. When the AMF sends a delete subscription request to the `Subscription` service with a `subscriptionId`, the service marks the corresponding subscription record as `DELETED`.

5. **NS Configuration Service**
   This microservice is responsible for configuring policy rules. It implements a REST messaging server that receives configuration HTTP messages, validates them, and stores the configuration in the database.

6. **NRF Client Manegement**
   This microservice registers with the NRF and sends periodic heartbeats, also maintains subscriptions with NRF for AMF sets.

   - NRF Registration and Heartbeat: Once NSSF is registered with NRF, NSSF contacts the NRF periodically. First the registration profile is configured using helm. Then the performance service calculates load and capacity of NF. NS registration requests the load and capacity from performance service and sends it to NRF with heartbeat.
   - NRF Subscription: NSSF subscribes to NRF for AMF based on the Target AMF Set and Region ID for registration and deregistration and load update.

7. **NRF Client Discovery Microservice**
   This microservice plays a crucial role in managing discovery requests within a network. Its primary function is to handle on-demand service discovery and efficiently manage interactions with the Network Repository Function (NRF).

   - Discovery: Performs service discoveries as required. When a service or function needs to be identified or connected, the discovery component initiates the process.
   - Handling Requests: Handles requests directed towards the NRF, which is responsible for registering and providing information on network functions and their services.
   - Response Processing: Once a response is received from the NRF, the microservice processes this information and prepares it for the requesting service.

8. **App-Info**: This microservice monitors application (microservice) health and status.

9. **Perf-Info**: This microservice monitors application (microservice) capacity and load status.

10. **Configuration Server**: This service performs the database abstraction for storage and retrieval of NSSF configuration.

11. **Alternate Route Service**: Alternate Route Service (ARS) is a microservice designed to efficiently find and provide alternate network routes. It employs a tiered approach, beginning with a fast cache lookup. If the desired route isn't cached or the cached entry is outdated, ARS checks predefined static mappings. It then queries DNS-SRV (if configured), updates its cache, and returns a response indicating success or failure. On success, ARS provides a list of alternate FQDNs (Fully Qualified Domain Names). These alternate routes can be configured either statically (using Helm charts) or dynamically (through DNS-SRV). Essentially, ARS prioritizes speed and efficiency in finding alternate routes by leveraging caching and fallback mechanisms.

12. **Ingress Gateway Service**
    This microservice is an entry point for accessing NSSF supported service operations and provides the functionality of an OAuth validator.

13. **Egress Gateway Service**
    This microservice is responsible to route NSSF initiated egress messages to other NFs.

Note

For more information on Ingress and Egress Gateway, see *Oracle Communications Cloud Native Core, Cloud Native Environment User Guide*.

4 NSSF Supported Features

This section explains about the NSSF supported features.

> **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 NRF Client Retry and Health Check

The NRF Client Health Check provides robust monitoring and management capabilities for Network Repository Function (NRF) services within the application architecture. This feature enables automated health assessments of NRF instances, ensuring high service availability, particularly in georedundant deployments.

Key Highlights:

- **Continuous Monitoring:** The NRF client automatically sends `NFProfileRetrieval` requests to all configured NRF instances at configurable intervals to assess their health status.
- **Failure Detection & Traffic Redirection:** If the primary NRF at a local site becomes unavailable, the NRF client detects the failure and seamlessly redirects service traffic to a secondary NRF at a remote site.
- **Intelligent Health Status Updates:** All NRF instances are initially marked as HEALTHY. Upon receiving specific error responses (5xx, 4xx, or configurable 3xx errors) to profile retrieval requests, instance health status is updated to UNHEALTHY, reducing failed traffic attempts.
- **Customized Configuration:** Both the health check interval and failure response codes can be customized to suit deployment requirements.
- **Operational Efficiency:** Only NRF instances confirmed as healthy are used to process session requests.

If the connection to the NRF is lost or experiences errors, the NSSF client will automatically retry the connection and perform regular health checks. This minimizes service disruption and improves fault tolerance within the 5G core network.

Health Check: Details and Call Flow

NRF Client performs health checks on all configured NRFs and maintains a list of healthy NRFs to be used during service requests. The NRFs can be configured either statically or by performing a DNS query using a virtual FQDN.

As part of the health check procedure, the NRF Client sends one `NFProfileRetrieval` request to all configured NRFs at each `healthCheckInterval`.

An NRF is considered unhealthy:

- If it sends a failure response for successive `HealthCheckConfig.healthCheckCount` number of times. A response is considered a failure if its HTTP status code is configured in `HealthCheckConfig.errorReasonsForFailure`.
- If it sends a failure response with a `Retry-After` value in the response header; in this case, the NRF is marked as unhealthy immediately for the `Retry-After` period.

An NRF is considered healthy:

- If it sends a success response for successive `HealthCheckConfig.healthCheckCount` number of times. A response is considered a success if its HTTP status code is not configured in `HealthCheckConfig.errorReasonsForFailure`.

> **Note**
>
> - If the HealthCheck feature is disabled, all NRFs are considered HEALTHY by default.
> - If the highest priority NRF is down, requests are sent to the next highest priority healthy NRF. When the highest priority NRF is marked as healthy again, requests are redirected back to it.

Retry Mechanism (`nrfRetryConfig`)

The Retry Mechanism is a fault-tolerance pattern that enables an application to automatically retry a failed operation. In this NRF Client implementation, it systematically handles temporary network communication failures by attempting to perform the same operation multiple times before considering it a complete failure. The feature allows NFs to attempt their service requests to alternate NRFs in case of failures.

General Call Flow

1. NRF Client triggers a service request to the highest priority and healthy NRF.
2. If a failure response or request timeout is received, the NRF Client retries the request to the same NRF for `NrfRetryConfig.primaryNrfRetryCount` times. On receiving a success response within these attempts, the response is consumed, and no further requests are sent.
3. If no success response is received, the NRF Client sends the service request to the next higher priority healthy alternate NRF.
   - If HealthCheck is disabled, the previous NRF is flagged as unhealthy for a period of `retryAfterTime`.
   - If HealthCheck is enabled, it flags the NRF as unhealthy based on its algorithm.
4. If a failure response or request timeout occurs again, the NRF Client retries the request to the same alternate NRF for `NrfRetryConfig.nonPrimaryNrfRetryCount` times. On receiving a success response, the response is consumed.
5. If a `Retry-After` time value is received in the response header from the NRF, the NRF Client halts further attempts to that NRF and flags it as unhealthy for the `Retry-After` period.
6. Steps 3–5 are repeated until one of the following occurs:
   - NRF Client receives a success response,
   - `NrfRetryConfig.alternateNRFRetryCount` is exhausted,

- All attempts to available healthy NRFs are exhausted.

7. If any of the above conditions are met, the NRF Client consumes the response and proceeds.

- Lower numbers indicate higher priority NRFs.
- A response is considered a failure if the response code or exception matches `NrfRetryConfig.errorReasonsForFailure/NrfRetryConfig.gatewayErrorCodes`.
- The primary and non-primary NRFs are georedundant.

Autonomous Procedures

For procedures such as `NFRegistration`, `NfHeartbeat`, `NfStatusSubscribe`, and `NFDiscovery`, the NRF Client performs the General Call Flow steps periodically until a success response is received:

- **NfRegistration:** Retried every `registrationRetryInterval`.
- **NfStatusSubscribe:** Retried every `subscriptionRetryInterval`.
- **NFDiscovery:** Retried every `discoveryRetryInterval`.
- **NfHeartbeat:** Retried every `heartbeatTimer` interval as received in the `NfRegistration` response.

> **Note**
>
> If a response code or exception is not configured in `NrfRetryConfig.errorReasonsForFailure/NrfRetryConfig.gatewayErrorCodes`, no retry will occur.

Limitations

1. The state of the NRFs is maintained in memory and not in a database. Hence, if the NRF Client pod goes down, the state information is lost.
2. The state of the NRFs is not shared across pods. Each NRF Client pod maintains its own NRF state. As a result, some subsequent requests may be sent to an unavailable NRF before being sent to an available one.

Managing NRF Client Retry and Health Check

This section provides information about Helm, REST API, and Cloud Native Configuration Console (CNC Console) configurations required to configure this feature.

Enable:

You can enable this feature using Helm configuration.

To enable it:

1. Open the `ocnssf_custom_values_25.2.200.yaml` file.
2. In the `Nrf-Client Micro service attributes` section, update the following parameters as shown below:

```
{
  "healthCheckConfig": {
    "healthCheckCount": 2,
```

```
    "healthCheckInterval": 5,
    "requestTimeout": 10,
    "errorReasonsForFailure": [
      "500",
      "502",
      "503",
      "504",
      "400",
      "403",
      "404",
      "405",
      "408",
      "411",
      "SocketTimeoutException",
      "JsonProcessingException",
      "UnknownHostException",
      "NoRouteToHostException",
      "IOException"
    ],
    "gatewayErrorCodes": [
      "503",
      "429"
    ]
  }
}
```

> **Note**
>
> Here, `healthCheckCount` > `0` enables proactive health checks, with a scheduled check every 5 seconds and NRF health status updated after 2 consecutive identical responses.

3. For detailed descriptions of these parameters, see the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
4. Save the file.

Description of the Enable Case

In the example above, within the `healthCheckConfig` section:

- The `healthCheckCount` value is set to `2`, which enables the health check feature.
- The `healthCheckInterval` value is `5`, meaning that every 5 seconds a scheduler evaluates whether the routes defined in the `nrfRouteList` (in this case, `primaryNrfApiRoot` and `secondaryNrfRoot`) have changed their status.

To justify a change in route status, the `healthCheckCount` must also be considered. This parameter defines the number of consecutive times the same status must be received before replacing the previous status with a new one.

A route evaluation is considered unhealthy under the following conditions:

- If the `NFProfileRetrieval` request breaches the `requestTimeout` condition
- If the `NFProfileRetrieval` request throws an exception or returns an error code defined in `errorReasonForFailure` or `gatewayErrorCode`.

- If the `NFProfileRetrieval` response contains a `Retry-After` header, the route is marked as unhealthy for the duration specified by that header, regardless of whether this is the first occurrence or whether the required number of consecutive evaluations has been completed.

Disable

You can disable this feature using Helm configuration.

To disable it:

1. Open the `ocnssf_custom_values_25.2.200.yaml` file.
2. In the `Nrf-Client Micro service attributes` section, update the following parameters as shown below:

```
{
  "healthCheckConfig": {
    "healthCheckCount": -1,
    "healthCheckInterval": 5,
    "requestTimeout": 10,
    "errorReasonsForFailure": [
      "500",
      "502",
      "503",
      "504",
      "400",
      "403",
      "404",
      "405",
      "408",
      "411",
      "SocketTimeoutException",
      "JsonProcessingException",
      "UnknownHostException",
      "NoRouteToHostException",
      "IOException"
    ],
    "gatewayErrorCodes": [
      "503",
      "429"
    ]
  }
}
```

> **Note**
>
> - With `healthCheckCount: -1`, health checks are disabled, and all NRFs are considered healthy by default.
> - The only difference between this example and the previous one is that when Health Check (HC) is disabled, the behavior depends entirely on the NRF Client Feature: Retry Mechanism (`nrfRetryConfig`). In this case, NRF requests are retried based solely on the result of the first request sent.
> - If a response code or exception is received that is not configured in the `healthCheckConfig`, the peer will be marked as healthy.

3. For detailed descriptions of these parameters, see the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
4. Save the file.

Configuration

Helm

Apart from the above configuration to enable or disable Health check, the following configurations should also be done for NRF Client Retry Mechanism.

```
{
  "nrfRetryConfig": [
    {
      "serviceRequestType": "ALL_REQUESTS",
      "primaryNRFRetryCount": 0,
      "nonPrimaryNRFRetryCount": 0,
      "alternateNRFRetryCount": -1,
      "errorReasonsForFailure": [
        "500",
        "502",
        "503",
        "504",
        "400",
        "403",
        "404",
        "405",
        "408",
        "411"
      ],
      "gatewayErrorCodes": [
        "503"
      ],
      "requestTimeout": 3
    },
    {
      "serviceRequestType": "AUTONOMOUS_NFREGISTER",
      "primaryNRFRetryCount": 1,
      "nonPrimaryNRFRetryCount": 1,
      "alternateNRFRetryCount": -1,
      "errorReasonsForFailure": [
        "500",
```

```
        "502",
        "503",
        "504",
        "400",
        "403",
        "404",
        "405",
        "408",
        "411"
      ],
      "gatewayErrorCodes": [
        "503"
      ],
      "requestTimeout": 3
    },
    {
      "serviceRequestType": "AUTONOMOUS_NFHEARTBEAT",
      "primaryNRFRetryCount": 0,
      "nonPrimaryNRFRetryCount": 0,
      "alternateNRFRetryCount": -1,
      "errorReasonsForFailure": [
        "500",
        "502",
        "503",
        "504",
        "400",
        "403",
        "404",
        "405",
        "408",
        "411"
      ],
      "gatewayErrorCodes": [
        "503"
      ],
      "requestTimeout": 3
    },
    {
      "serviceRequestType": "AUTONOMOUS_NFPATCH",
      "primaryNRFRetryCount": 0,
      "nonPrimaryNRFRetryCount": 0,
      "alternateNRFRetryCount": -1,
      "errorReasonsForFailure": [
        "500",
        "502",
        "503",
        "504",
        "400",
        "403",
        "404",
        "405",
        "408",
        "411"
      ],
      "gatewayErrorCodes": [
        "503"
```

```
      ],
      "requestTimeout": 3
    },
    {
      "serviceRequestType": "NFDEREGISTER",
      "primaryNRFRetryCount": 0,
      "nonPrimaryNRFRetryCount": 0,
      "alternateNRFRetryCount": -1,
      "errorReasonsForFailure": [
        "500",
        "502",
        "503",
        "504",
        "400",
        "403",
        "404",
        "405",
        "408",
        "411"
      ],
      "gatewayErrorCodes": [
        "503"
      ],
      "requestTimeout": 3
    }
  ]
}
```

For detailed descriptions of these parameters, see the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*

REST API

There are no REST API configurations required for this feature.

CNC Console

There is no option to enable or disable this feature using CNC Console.

Observe

Metrics

No new metrics have been added for this feature. However, it uses the following existing metric from the NSSF Common metrics section:

- `nrfclient_nrf_operative_status`

For more information about other metrics, see NSSF Metrics section.

KPIs

There are no new KPIs for this feature.

Alerts

No new alerts have been added for this feature. However, it uses the following existing Application Level Alerts:

- OcnssfAllNrfInstancesInDownStateCritical
- OcnssfNrfInstancesInDownStateMajor

For more information about these alerts, see NSSF Alerts section.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.2 Support for TLSv1.3 on Internal API Communication

In a Kubernetes-based 5G Core deployment, the Network Slice Selection Function (NSSF) interacts with the Kubernetes API server (Kube-API-Server) to retrieve Secrets and ConfigMaps.

With this enhancement, NSSF now supports TLSv1.3 in addition to TLSv1.2 for secure communication with the Kubernetes API server. The Support for TLS for Consumer NFs and Producer NFs is already there in NSSF.

Note

- **Exclusive TLS version support**: NSSF does not support TLSv1.2 and TLSv1.3 concurrently. All pods must use the same version (either TLSv1.2 or TLSv1.3) when communicating with the Kubernetes API server.
- **CNE version requirement**: An application TLS server configured with TLSv1.3 requires a corresponding CNE version that also supports TLSv1.3.
- **Secret volume mount latency**: When Secrets are mounted as volumes, changes may not be detected immediately within the configured reload period. Kubelet may take several seconds to a few minutes (typically 1–3 minutes) to propagate updates to the volumes. Once the updates are reflected, they are detected in the next scheduler run, triggering the necessary actions.
- **Behavior when `tlsVersionSupportForKubeApiServer.enabled` is enabled**:
  - **Helm installation or upgrade fails if:**
    - An invalid TLS version is configured.
    - Configured cipher suites are incompatible with the selected TLS version, or no cipher suites are configured.
    - Validation of Helm configuration fails because secrets required by enabled features are missing in `featureSecrets`.
  - Mounting Secrets across namespaces is not supported. All Secrets must reside in the same namespace as the Ingress Gateway, Egress Gateway, and Alternate-Route microservices.
  - For the CCA Header feature, any Secret added using REST API that was not defined in the Helm chart at deployment is considered unavailable.
  - Perform a Helm upgrade whenever adding or updating Secrets in `featureSecrets`.

Managing Support for TLSv1.3 on Internal API Communication

This section provides information about Helm, REST API, and Cloud Native Configuration Console (CNC Console) configurations required to configure this feature.

Enable:

This feature is disabled by default when NSSF is deployed.

To enable it using Helm:

1. Open the `ocnssf_custom_values_25.2.200.yaml` file.
2. In the `Global parameters` section, set:
   - `global.tlsVersionSupportForKubeApiServer.enabled` to `true`
3. Configure the following parameters as required:
   - **TLS version**
     - `global.tlsVersionSupportForKubeApiServer.kubeApiServerTlsVersion` – specifies the TLS version to be used.
   - **Cipher suites**
     - `global.tlsVersionSupportForKubeApiServer.cipherSuites` – defines the cipher suites for the selected TLS version.
   - **Ingress Gateway Secrets**
     - `ingressgateway.tlsVersionSupportForKubeApiServer.featureSecrets` – lists all Secrets required for Ingress Gateway volume mounting (for example, TLS, Message Copy (SASL/SSL), CCA header).
   - **Egress Gateway Secrets**
     - `egressgateway.tlsVersionSupportForKubeApiServer.featureSecrets` – lists all Secrets required for Egress Gateway volume mounting (for example, HTTPS for Egress Gateway, Message Copy (SASL/SSL)).
4. For detailed descriptions of these parameters, see the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
5. Save the file.
6. Install NSSF. For the installation procedure, see the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
7. If enabling the feature after deployment, run a Helm upgrade. For upgrade steps, see the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

REST API

There are no REST API configurations required for this feature.

CNC Console

There is no option to enable or disable this feature using CNC Console.

Observe

Metrics

The following metrics are added in the Ingress Gateway Metrics section:

- `oc_ingressgateway_ip_addresses_fetch_failure`

The following metrics are added in the Egress Gateway Metrics section:

- `oc_egressgateway_ip_addresses_fetch_failure`

The following metrics are added in the NSSF Common metrics section:

- `oc_certificatemanagement_tls_certificate_info`
- `oc_certificatemanagement_tls_secret_status`
- `oc_certificatemanagement_tls_certs_reload_failure`
- `cgiu_jetty_ip_address_fetch_failure`

For more information about metrics, see NSSF Metrics section.

KPIs

There are no new KPIs for this feature.

Alerts

There are no new alerts for this feature.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.3 Support for Dual Stack

Using the dual stack mechanism, applications or NFs can establish connections with pods and services in a Kubernetes cluster using IPv4 or IPv6 or both simultaneously. Dual stack provides:

- coexistence strategy that allows hosts to reach IPv4 and IPv6 simultaneously.
- IPv4 and IPv6 allocation to the Kubernetes clusters during cluster creation. This allocation is applicable for all the Kubernetes resources unless explicitly specified during the cluster creation.

With this feature, NSSF can be deployed on a dual stack Kubernetes infrastructure. Using the dual stack mechanism, NSSF establishes and accepts connections within pods and services in a Kubernetes cluster using either IPv4 or IPv6. NSSF communicates with entities that support both IPv4 and IPv6.

NSSF supports:

- Dual stack communication with external entities, such as other NFs.
- Only IPv6 communication within the NSSF microservices.
- Only IPv6 communication with cnDBTier, CNC Console, and Kubernetes APIs.

The dual stack architecture for NSSF within a Kubernetes cluster can be visualized as follows:

Edge Microservices (Ingress Gateway):

- Handles incoming traffic from external Network Functions (NFs) and supports dual stack communication.
- Can accept IPv4 and IPv6 addresses based on the incoming request.
- Establishes connections to NSSF backend microservices using single stack IPv6 in IPv6-preferred configurations.

Backend Microservices:

- Operate as single stack, either IPv4 or IPv6, based on the deployment mode.
- Communicate with other backend services through the configured IP family policy.
- IP families and policies are aligned with deployment mode settings to ensure compatibility.

Egress Gateway Microservices:

- Routes outgoing requests using either IPv4 or IPv6.
- Configurable using the `global.egressRoutingMode` Helm parameter to support preferred routing modes.

> **Note**
>
> The Ingress Gateway microservice establishes connections with NSSF backend microservices using a single stack, with IPv6 preferred.
>
> - **Backend Microservice**s: All other microservices, except the Ingress Gateway microservice, are considered backend microservices. NSSF supports single stack IPv4 or IPv6 for backend microservices. All backend microservices interact with each other using a single stack.
> - **Egress Gateway Microservice**: Provides options for sending outgoing requests using either IPv4 or IPv6. The preferred routing mode can be configured using the `global.egressRoutingMode` Helm parameter.

Pods and Services

The services are broadly categorized into two types: **external pods and services** and **internal pods and services**. Each category has specific networking requirements for handling Ingress Gateway and Egress Gateway traffic.

External Pods and Services

- **Ingress Gateway Traffic**: External pods and services handling Ingress Gateway traffic must be configured as dual stack. This means both pods and services must be exposed on both IPv6 and IPv4 interfaces, with IPv6 being the preferred IP.
- **Egress Gateway Traffic**: For Egress Gateway traffic, external pods must also be dual stack, exposed on both IPv6 and IPv4 interfaces, with IPv6 again set as the preferred IP.

Internal Pods and Services

- **Ingress Gateway Traffic**: Internal pods and services for Ingress Gateway traffic are expected to be single stack, meaning they should be exposed only on IPv6 interfaces.
- **Egress Gateway Traffic**: Similarly, for Egress Gateway traffic, internal pods must be single stack and exposed only on IPv6 interfaces, with IPv6 as the preferred IP.

Deployment Modes

The feature introduces five deployment modes to accommodate different network scenarios:

Table 4-1 Deployment Modes

Deployment Mode	Internal Microservices	External Services	Behavior
IPv4 Only	Single stack (IPv4)	Single stack (IPv4)	Only single stack and IPv4 preferred for all services. If the deployment mode is set to IPv4, then all communication between microservices is through IPv4. In this case, both internal microservices and external services' IP family policy is single stack, and IP families will be IPv4.
IPv6 Only	Single stack (IPv6)	Single stack (IPv6)	Only single stack and IPv6 preferred for all services. If the deployment mode is set to IPv6, then all communication between microservices is through IPv6. In this case, both internal microservices and external services' IP family policy is single stack, and IP families will be IPv6.
IPv4 over IPv6	Single stack (IPv4)	Dual stack (IPv4 and IPv6)	Dual stack and IPv4 preferred for all services. If the deployment mode is set to IPv4_IPv6, then all communication between microservices is through IPv4, since IPv4 is preferred. In this case, internal microservices' IP family policy is single stack, and IP families will be IPv4. The IP family policies of the external-facing services require dual stack, and IP families will be IPv4 and IPv6.
IPv6 over IPv4	Single stack (IPv6)	Dual stack (IPv4 and IPv6)	Dual stack and IPv6 preferred for all services. If the deployment mode is set to IPv6_IPv4, then all communication between microservices is through IPv6, since IPv6 is preferred. In this case, internal microservices' IP family policy is single stack, and IP families will be IPv6. The IP family policies of the external-facing services require dual stack, and IP families will be IPv4 and IPv6.
Cluster Preferred	Single stack (cluster IP-based)	Single stack (cluster IP-based)	There will be no changes in the service file. IPs are assigned depending on the cluster. In this case, both internal microservices' and external-facing services' IP family policies are single stack, and IP families will be the cluster IP address.

IP Address Allocation to Pods

IP address allocation to pods depends on the IP address preference set in the Kubernetes cluster. Pods do not have the capability to select their IP address. For example:

IPv4 Preferred Configuration: If the Kubernetes cluster is configured with IPv4 as the preferred address family, both IPv4 and IPv6 addresses are allocated to the pod, but the primary address will be IPv4.

Example of a pod with a primary IPv4 address in an IPv4-preferred infrastructure:

```
Status: Running
IP: 10.xxx.xxx.xxx
IPs:
 IP: 10.xxx.xxx.xxx
 IP: fd00::1:1xxx:9xxx:bxxx:fxxx
```

IPv6 Preferred Configuration: If the Kubernetes cluster is configured with IPv6 as the preferred address family, both IPv4 and IPv6 addresses are allocated to the pod, but the primary address will be IPv6.

Example of a pod with a primary IPv6 address in an IPv6-preferred infrastructure:

```
Status: Running
IP: fd00::1:1xxx:9xxx:bxxx:fxxx
IPs:
 IP: fd00::1:1xxx:9xxx:bxxx:fxxx
 IP: 10.xxx.xxx.xxx
```

IP Address Allocation to Services

IP address allocation for all NSSF services depends on the deployment mode specified in the Helm configuration, as defined in the table below. When the Helm parameters are configured appropriately, NSSF automatically sets the IP Family Policy and IP Families attributes for its services.

Table 4-2 IP Allocation As Per Supported Deployment Modes

Supported Mode	Deployment Mode (global.deploymentMode)	NSSF Microservices	Egress Routing Mode (global.egressRoutingMode)	Description
IPv4 Only	IPv4	IPv4	IPv4	The Ingress Gateway service will be single stack with IPv4 only. It uses IPv4 pod IPs to establish connections.
IPv6 Only	IPv6	IPv6	IPv6	The Ingress Gateway service will be single stack with IPv6 only. It uses IPv6 pod IPs to establish connections.
IPv4 over IPv6	IPv4_IPv6	IPv4	Any	The Ingress Gateway service will be dual stack with both IPv4 and IPv6 addresses. It uses IPv4 pod IPs to establish connections.
IPv6 over IPv4	IPv6_IPv4	IPv6	Any	The Ingress Gateway service will be dual stack with both IPv4 and IPv6 addresses. It uses IPv6 pod IPs to establish connections.
Cluster Preferred	ClusterPreferred	ClusterPreferred	None	This is the default value configured for `global.deploymentMode`. All NSSF services will be single stack with IPv4 or IPv6 based on the infrastructure preference.

Customizing IP Address Allocation

You can customize IP address allocation for services using the Helm parameters outlined above. Services route traffic to destination endpoints based on these configurations. For example:

- If `global.deploymentMode` is set to IPv4, only IPv4 addresses are allocated to services, and services use IPv4 pod IPs to send traffic to their endpoints.
- If `global.deploymentMode` is set to ClusterPreferred, IPs are assigned based on the cluster's IP family configuration.

Upgrade Impact

In a dual stack environment, all NSSF backend services are configured as single stack. Due to Kubernetes limitations, an NSSF upgrade is not supported when there is a mismatch between the `ipFamilies` configured for NSSF backend microservices and the deployment mode's `ipFamilies` configuration. In such cases, NSSF recommends performing a fresh installation to change the preferred `ipFamilies`. However, this does not apply to the edge microservice (Ingress Gateway microservice), as it can expand the `ipFamilies` list by setting it to dual stack based on cluster preferences.

Managing Support for Dual Stack

This section provides information about Helm, REST API, and Cloud Native Configuration Console (CNC Console) configurations required to configure this feature.

Enable:

The support for dual stack is a core functionality of NSSF. You do not need to enable or disable this feature. However, configuration for the Edge and Egress Gateway microservices is required for this feature to work.

Configuration for Edge and Egress Gateway Microservices

- Edge Microservices: IP address allocation for edge microservices (such as the Ingress Gateway) is determined by the `global.deploymentMode` Helm parameter.
- Egress Gateway Microservices: IP selection and connection preferences for outgoing traffic are managed by the `global.egressRoutingMode` Helm parameter. The preferred routing mode can be configured to prioritize either IPv4 or IPv6, depending on network requirements.

Configure

This section lists the configuration details for this feature.

Helm

Use the following Helm parameters to configure dual stack behavior:

1. Open the `ocnssf_custom_values_25.2.200.yaml` file in a text editor.
2. Update the following parameters:

```
global:
  deploymentMode: clusterPreferred  # Indicates the edge deployment mode.
  egressRoutingMode: None           # Controls IP address selection and
connections for the Egress Gateway microservice.
```

3. Save the changes to the file.
4. Install NSSF. Follow the installation procedure provided in the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide* to install NSSF using the updated configuration file.

REST API

There are no REST API configurations required for this feature.

CNC Console
There is no option to enable or disable this feature using CNC Console.

Observe

Metrics

The following metrics are added in the Ingress Gateway Metrics section:

- `oc_ingressgateway_incoming_ip_type`
- `oc_ingressgateway_outgoing_ip_type`

The following metrics are added in the Egress Gateway Metrics section:

- `oc_egressgateway_incoming_ip_type`
- `oc_egressgateway_outgoing_ip_type`
- `oc_egressgateway_dualstack_ip_rejected_total`

For more information about metrics, see NSSF Metrics section.

KPIs

There are no new KPIs for this feature.

Alerts

There are no new alerts for this feature.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

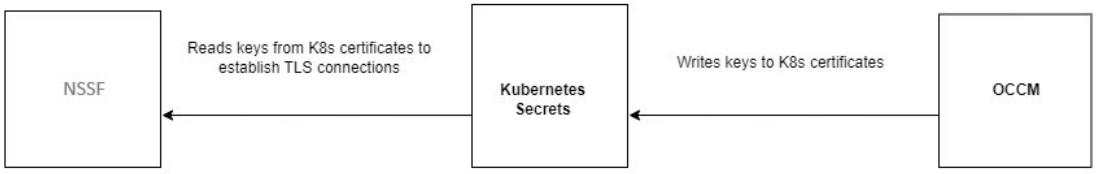
1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.4 Support for Automated Certificate Lifecycle Management

NSSF uses secure protocols, such as HTTPS and Secure Socket Layer (SSL) or Transport Layer Security (TLS), to establish and manage secure connections. This is achieved using Public and Private Keys and the presence of trusted authorities such as Certificate Authorities (CA), which create and issue certificates. These certificates have validity. You must renew these certificates before they expire. These certificates can be revoked when the CA or its keys are compromised

Starting with NSSF 25.1.2xx, you can integrate NSSF with Oracle Communications Cloud Native Core, Certificate Management (OCCM) to support automation of certificate lifecycle management. OCCM manages TLS certificates stored in Kubernetes secrets by integrating

with Certificate Authority (CA) using the Certificate Management Protocol Version 2 (CMPv2) protocol in the Kubernetes secret. OCCM obtains and signs TLS certificates within the NSSF namespace. For more information about OCCM, see *Oracle Communications Cloud Native Core, Certificate Management User Guide*.



The above diagram indicates that OCCM writes the keys to the certificates and NSSF reads these keys to establish a TLS connection with other NFs.

OCCM can automatically manage the following TLS certificates:

- 5G Service Based Architecture (SBA) client TLS certificates
- 5G SBA server TLS certificates
- Message Feed TLS certificates

Install Guide Considerations

- **Upgrade**: When NSSF is deployed with OCCM, follow the specific upgrade procedure. For information about the upgrade strategy, see "Upgrading NSSF" in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
- **Rollback**: For more information on migrating the secrets from NSSF to OCCM and removal of Kubernetes secrets from the yaml file, see "Postupgrade Task" in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

Managing DNS SRV Based Selection of NRF in NSSF

This section provides information about Helm, REST API, and Cloud Native Configuration Console (CNC Console) configurations required to configure this feature.

Configure

There are no additional configuration changes required at NSSF.

Observe

Metrics

The following metric is available for this feature:

- `oc_egressgateway_connection_failure_total`
- `oc_ingressgateway_connection_failure_total`

For more information about metrics, see NSSF Metrics section.

KPIs

There are no new KPIs for this feature.

Alerts

There are no new alerts for this feature.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.5 DNS SRV Based Selection of NRF in NSSF

Currently, the NSSF selects the NRF for communication based on static configurations set by the operator. However, there are scenarios where the NRF may go down for various reasons. In such cases, the NSSF, which relies on the NRF for specific communications, may experience service disruptions.

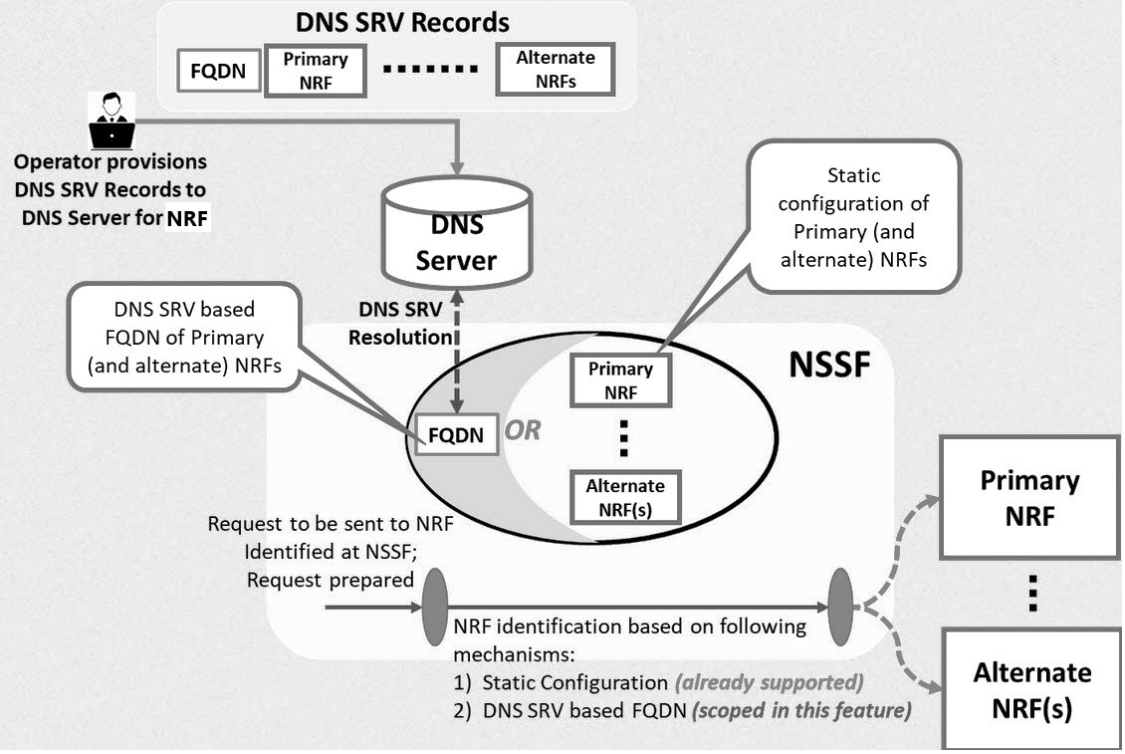
The DNS SRV based selection of NRF in NSSF feature enhances network resilience by utilizing the NRF's georedundancy capabilities to handle potential site failures. This setup enables Network Functions (including NSSF) to continue operating smoothly by redirecting traffic from a primary NRF to a secondary NRF when a failure occurs at the primary site.

This feature allows the NSSF to dynamically select NRF instances based on real-time availability and site redundancy through DNS SRV configurations. In addition to the static configurations by operators, the NSSF can now resolve NRFs using DNS SRV-based Fully Qualified Domain Names (FQDNs). The NSSF is configured with a primary NRF and multiple fallback NRFs, which take over if the primary NRF becomes unreachable.

The nrf-client uses the Alternate Route Service, which helps the Network Slice Selection Function (NSSF) find and select different Network Repository Functions (NRFs) by using DNS SRV-based lookups. This service allows the NSSF to translate Fully Qualified Domain Names (FQDNs) or virtual FQDNs into alternate NRF addresses. This setup enables the NSSF to prioritize and adjust connections to different NRFs based on specific service needs.

How Alternate Routing Works

Figure 4-1 Call Flow of DNS SRV Based Selection of NRF in NSSF



1. **DNS SRV Record Lookup**: The NSSF starts by using DNS SRV records to fetch information about the FQDN of the primary NRF and alternate NRFs.
2. **DNS Query**: The DNS server receives a query from the NSSF to resolve the FQDN of the primary NRF or alternate NRFs.
3. **DNS Resolution**: The DNS server resolves the FQDN and returns the resolved address for the primary NRF or one of the alternate NRFs.
4. **NRF Identification by NSSF**: Based on the FQDN provided by DNS, the NSSF identifies the primary NRF to communicate with. If the primary NRF is unavailable, the NSSF uses one of the alternate NRFs.
5. **Static Configuration Option**: Alternatively, the NSSF can rely on a static configuration of the primary and alternate NRFs instead of querying DNS.
6. **Request Preparation**: Once an NRF (primary or alternate) is identified, the NSSF prepares the request to be sent to the selected NRF.
7. **Communication with NRF**: The NSSF sends the request to the identified NRF, enabling further processing or service access.

The Alternate Route Service utilizes DNS SRV records to look up virtual FQDNs. The Egress Gateway queries this service to retrieve a list of alternate NRFs along with their priorities. Based on these priorities, the Egress Gateway reroutes requests to other NRF instances as required.

Key Operations of nrf-client for High Availability

The NSSF uses a component called nrf-client to facilitate communication with NRFs. The nrf-client performs critical functions to ensure NRFs remain available and responsive:

1. **Traffic Routing**: The nrf-client directs all requests (for example, registration, updates, heartbeats, and discovery) to the primary NRF.
2. **Failure Detection**: If the primary NRF cannot be reached, the nrf-client detects the failure either through routing errors or by receiving a "503 Service Unavailable" response, which may include a "Retry-After" interval.
3. **Failover and Retry**: If the primary NRF fails, the nrf-client temporarily marks it as unavailable and reroutes traffic to a secondary NRF based on DNS SRV priority. The retry interval, specified in the "Retry-After" header or a preset interval, determines when the nrf-client reattempts contact with the primary NRF.
   - During this interval, all requests are routed to the secondary NRF.
   - Once the retry interval expires, the nrf-client resumes attempts to communicate with the primary NRF.

> **Note**
>
> The nrf-client only tries to connect with each NRF once. If it cannot send a request to an NRF, it returns a "503 Service Unavailable" response to the NSSF. This one-time attempt helps avoid repeated failures and ensures alternate routing to secondary NRFs when needed.

Managing DNS SRV Based Selection of NRF in NSSF

This section provides information about Helm, REST API, and Cloud Native Configuration Console (CNC Console) configurations required to configure this feature.

Enable:

You can enable this feature using Helm configurations. To enable this feature, it is mandatory to configure the following Helm parameters as given below:

```
enableVirtualNrfResolution=true
virtualNrfFqdn=nrfstub.changeme-ocats.svc
virtualNrfScheme=http
```

> **Note**
>
> If `enableVirtualNrfResolution` is set to `false`, the nrf-client uses `primaryNrfApiRoot` and `secondaryNrfApiRoot` configurations to register with the NRF. However, if `enableVirtualNrfResolution` is set to `true` and an incorrect `virtualNrfFqdn` is configured, the nrf-client does not fallback to `primaryNrfApiRoot` and `secondaryNrfApiRoot` for registration, resulting in a registration failure.

Apart from this, for more information on other Helm parameters supported for this feature, see Helm section below.

Configure

Helm

To enable this feature, configure the following `nrf-client` Helm parameters in `ocnssf_custom_values_25.2.200.yaml` file.

```
nrf-client:
  # This config map is for providing inputs to NRF-Client
  configmapApplicationConfig:
    &configRef
    profile: |-
      [appcfg]
      primaryNrfApiRoot=nrf-stubserver.changeme-ocats:8080
      secondaryNrfApiRoot=nrf-stubserver.changeme-ocats:8080
      nrfScheme=http
      retryAfterTime=PT120S
      nrfClientType=NSSF
      nrfClientSubscribeTypes=
      appProfiles=[{
  "nfInstanceId": "9faf1bbc-6e4a-4454-a507-aef01a101a01",
  "nfType": "NSSF",
  "nfStatus": "REGISTERED",
  "heartBeatTimer": 30,
  "fqdn": "ocnssf-nsgateway.ocnssf.svc",
  "priority": 1,
  "capacity": 1,
  "load": 2,
  "plmnList": [
    {
      "mcc": "311",
      "mnc": "480"
    }
  ],
  "nfSetIdList": [
    "setEast.nssfset.5gc.mnc480.mcc311"
  ],
  "locality": "rcnltxekloc1",
  "nfServices": [
    {
      "serviceInstanceId": "92d59bfc-e5d6-47f5-a26b-3a03facdebcc",
      "serviceName": "nnssf-nsselection",
      "versions": [
        {
          "expiry": null,
          "apiFullVersion": "1.0.0",
          "apiVersionInUri": "v1"
        }
      ],
      "scheme": "http",
      "nfServiceStatus": "REGISTERED",
      "fqdn": "ocnssf1-ingress-gateway.ocnssf.svc",
      "interPlmnFqdn": null,
      "ipEndPoints": [
        {
          "ipv4Address": "10.224.45.178",
          "transport": "TCP",
          "port": 80
        }
```

```
        ],
        "allowedNfTypes": [
          "AMF",
          "NSSF"
        ],
        "priority": 1,
        "capacity": 1,
        "load": 2
      },
      {
        "serviceInstanceId": "d33728cd-6e21-434b-bc5a-ed69bc612377",
        "serviceName": "nnssf-nssaiavailability",
        "versions": [
          {
            "expiry": null,
            "apiFullVersion": "1.0.0",
            "apiVersionInUri": "v1"
          }
        ],
        "scheme": "http",
        "nfServiceStatus": "REGISTERED",
        "fqdn": "ocnssf2-ingress-gateway.ocnssf.svc",
        "interPlmnFqdn": null,
        "ipEndPoints": [
          {
            "ipv4Address": "10.224.45.179",
            "transport": "TCP",
            "port": 80
          }
        ],
        "allowedNfTypes": [
          "AMF",
          "NSSF"
        ],
        "priority": 1,
        "capacity": 1,
        "load": 2
      }
    ]
  }]
```


```
        enableF3=true
        enableF5=true
        renewalTimeBeforeExpiry=3600
        validityTime=30
        enableSubscriptionAutoRenewal=true
        nfHeartbeatRate=80
        acceptAdditionalAttributes=false
        retryForCongestion=5
        enableVirtualNrfResolution=true
        virtualNrfFqdn=nrfstub.changeme-ocats.svc
        virtualNrfScheme=http
```

> **Note**
>
> This is a sample configuration. Modify the parameters as per your setup. For detailed information about the Helm parameters, see the "Customizing NSSF " section in the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

REST API

There are no REST API configurations required for this feature.

CNC Console
There is no option to enable or disable this feature using CNC Console.

Observe

Metrics

The following metric is available for this feature:

- `nrfclient_nrf_operative_status`
- `nrfclient_dns_lookup_request_total`

For more information about metrics, see NSSF Metrics section.

KPIs

There are no new KPIs for this feature.

Alerts

The following alerts are available for this feature:

- OcnssfNrfInstancesInDownStateMajor
- OcnssfAllNrfInstancesInDownStateCritical

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.6 Deleting All Slices in a TAI Using PATCH Remove Operation

In the context of 5G networks, network slicing is a crucial feature. It allows the network to be divided into multiple virtual slices, each optimized for a specific type of service or use case. These slices are defined by S-NSSAIs (Single Network Slice Selection Assistance Information) and are associated with specific geographical areas, referred to as Tracking Areas (TAIs).

The Access and Mobility Function (AMF) is responsible for managing these slices and ensuring that these devices connect to the appropriate slices. The Network Slice Selection Function (NSSF) assists the AMF in managing and selecting these slices.

Currently, when the AMF needs to update or delete slices for specific areas, there are limitations in how the NSSF responds to such requests. This new feature addresses those limitations, improving flexibility and compliance with certain operational requirements.

This feature enhances the NSSF to support specific behaviors for managing network slices through the PATCH operation. This includes the ability to:

- delete all slices for specific areas (TAI) using PATCH.
- improve responses for better clarity and compliance, including the use of `204 No Content` when all slices in an area are removed.
- allow slices to be added back later through PATCH operations.

Old Behavior

- **Slice Management with PATCH Operations:**
  - The NSSF responded with `200 OK` to all PATCH requests, even if slices were only partially deleted.
  - `204 No Content` responses were not supported for PATCH operations.
  - If no authorized slices were found in a PATCH request, the NSSF returned a `403 Forbidden` response.
- **Deletion Constraints:**
  - The AMF could only delete all slices across all areas using a DELETE request, which was not always practical.
  - The NSSF required at least one slice to remain in a TAI, in accordance with existing specifications.

Enhanced Behavior (When the feature is enabled):

- **Enhanced Deletion Using PATCH:**
  - The AMF can now issue a PATCH request to delete all slices for a specific TAI.
    - If all slices are successfully deleted for the specified area(s), the NSSF responds with `204 No Content`.
    - If some slices remain after the deletion, the NSSF responds with `200 OK`, including the list of remaining slices.
- **Support for Full Deletion Scenarios**:
  - If a `NssaiAvailability` PATCH request results in deletion of all slices within a specific TAI:
    - The response can be `200 OK` (if other TAIs still have SNSSAIs that are supported by the AMF and authorized by NSSF), or
    - `204 No Content` (if none of the SNSSAIs supported by the AMF are authorized by NSSF).
  - If all slices are deleted across all TAIs, the NSSF responds with `204 No Content`.
- **Adding Back Slices:** After a full deletion, the AMF can re-add slices for the same area using a PATCH operation.
- **Oracle NSSF Specific Handling:**
  - When processing a PATCH request that removes all slices, the NSSF verifies compliance with the 3GPP specification, which requires at least one slice in an area.
  - To support the new feature, the NSSF uses a flag:

* If the flag is enabled, the NSSF can handle and store areas with no slices (for example, storing a null value or a placeholder).
* This ensures the system behaves as expected while complying with 3GPP standards and supporting this feature.

Enhanced Behavior (When the feature is disabled):

- **Enhanced Deletion Using PATCH**:
  - The system first checks the size of the `supportedSnssais` list. If it is greater than 1, NSSF responds with `200 OK`, including the list of remaining slices.
  - Otherwise, deletion is not allowed, and NSSF returns a `400 Bad Request` response.

This feature provides flexibility for the AMF to manage slices with greater granularity, targeting specific areas without affecting others, thereby improving efficiency and control. It aligns with 3GPP standard operations for PATCH while introducing flexibility for specific use cases (e.g., FOA). Additionally, returning `204 No Content` for full deletions offers a clearer indication that all slices were successfully removed, enhancing system transparency.

Managing Deletion of All Slices in a TAI Using PATCH Remove Operation

This section provides information about Helm, REST API, and Cloud Native Configuration Console (CNC Console) configurations required to configure this feature.

Enable:

You can enable this feature using REST API or CNC Console.

Configure

Helm

There are no Helm configurations required for this feature.

REST API

A new boolean parameter, `enhancedPatchBehaviourEnable`, is added to enable or disable this feature using **SystemOptions** API. Update this parameter value as `true` or `false` to enable or disable this feature, respectively. By default, it is set to `false`.

For more information about the REST API configuration, see "SystemOptions" section in "NSSF REST Specifications" chapter of *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

CNC Console

The value of `EnhancedPatchBehaviourEnable` parameter can also be updated using CNC Console interface for NSSF System Option.

Observe

Metrics

There are no new Metrics for this feature. However, this feature uses the following metric to count the success response messages sent by NSSF for requests for the Nnssf_NSSAIAvailability service.

`ocnssf_nssaiavailability_success_tx_total`

For more information about metrics, see NSSF Metrics section.

KPIs

There are no new KPIs for this feature.

Alerts

There are no new alerts for this feature.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

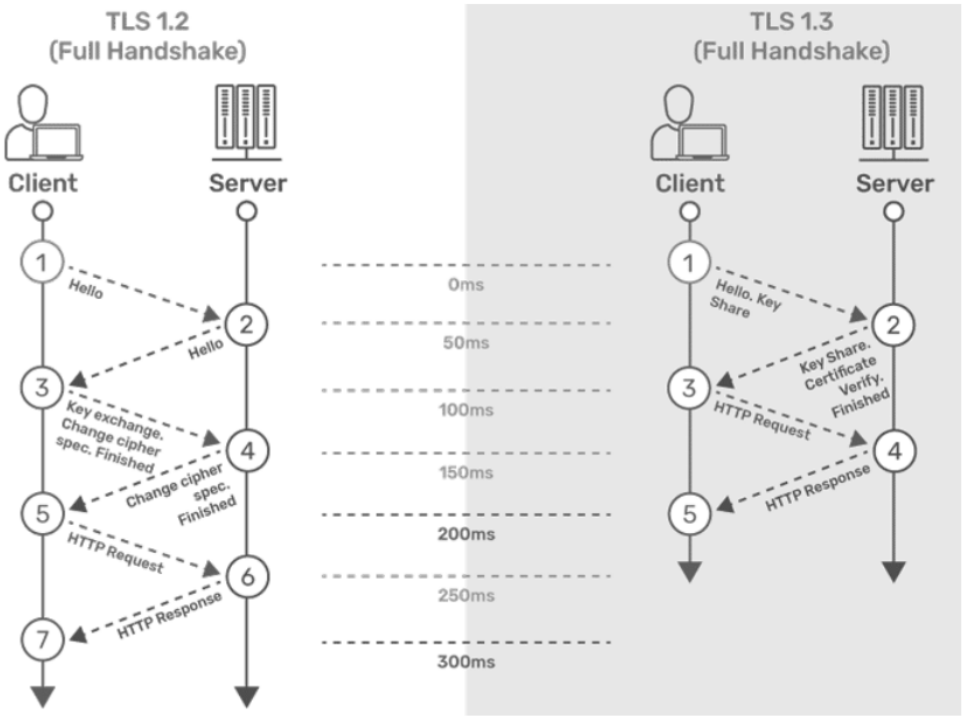
4.7 Support for TLS

NSSF uses Hypertext Transfer Protocol Secure (HTTPS) to establish secure connections with Consumer NFs and Producer NFs, respectively. These communication protocols are encrypted using Transport Layer Security (TLS). TLS comprises the following components:

- **Handshake Protocol**: Exchanges the security parameters of a connection. Handshake messages are supplied to the TLS record layer.
- **Record Protocol**: Receives the messages to be transmitted, fragments the data into multiple blocks, secures the records, and then transmits the result. Received data is delivered to higher-level peers.

TLS Handshake

This section describes the differences between TLSv1.2 and TLSv1.3 and the advantages of TLSv1.3 over TLSv1.2 and earlier versions.



TLSv1.2

1. The connection or handshake starts when the client sends a “client hello” message to the server. This message consists of cryptographic information such as supported protocols and supported cipher suites. It also contains a random value or random byte string.
2. To respond to the “client hello” message, the server sends the “server hello” message. This message contains the CipherSuite that the server has selected from the options provided by the client. The server also sends its certificate along with the session ID and another random value.
3. The client verifies the certificate sent by the server. When the verification is complete, it sends a byte string and encrypts it using the public key of the server certificate.
4. When the server receives the secret, both the client and server generate a master key along with session keys (ephemeral keys). These session keys are used to symmetrically encrypt the data.
5. The client sends an “HTTP Request” message to the server to enable the server to switch to symmetric encryption using the session keys.
6. To respond to the client’s “HTTP Request” message, the server does the same and switches its security state to symmetric encryption. The server concludes the handshake by sending an HTTP response.
7. The client-server handshake is completed in two round-trips.

TLSv1.3

1. The connection or handshake starts when the client sends a “client hello” message to the server. The client sends the list of supported cipher suites. The client also sends its key share for that particular key agreement protocol.
2. To respond to the “client hello” message, the server sends the key agreement protocol that it has chosen. The “Server Hello” message comprises the server key share, server certificate, and the “Server Finished” message.

3. The client verifies the server certificate, generates keys as it has the key share of the server, and sends the “Client Finished” message along with an HTTP request.
4. The server completes the handshake by sending an HTTP response.

The following digital signature algorithms are supported in TLS handshake:

Table 4-3 Digital Signature Algorithms

Algorithm	Key Size (Bits)	Elliptic Curve (EC)
RS256 (RSA)	2048	NA
	4096 This is the recommended value.	NA
ES256 (ECDSA)	NA	SECP384r1 This is the recommended value.

Comparison Between TLSv1.2 and TLSv1.3

The following table provides a comparison of TLSv1.2 and TLSv1.3:

Table 4-4 Comparison of TLSv1.2 and TLSv1.3

Feature	TLS v1.2	TLS v1.3
TLS Handshake	• The initial handshake was carried out in clear text. • A typical handshake in TLSv1.2 involves the exchange of 5 to 7 packets.	• The initial handshake is carried out along with the key share. • A typical handshake IN TLSv1.3 involves the exchange of up to 3 packets.
Cipher Suites	• Less secure Cipher suites. • Use SHA-256 and SHA-384 hashing – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 – TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	• More secure Cipher suites. • Apart from all the ciphers supported for TLSv1.2, the following additional ciphers are supported for only TLSv1.3: – TLS_AES_128_GCM_SHA256 – TLS_AES_256_GCM_SHA384 – TLS_CHACHA20_POLY1305_SHA256
Round-Trip Time (RTT)	This has a high RTT during the TLS handshake.	This has low RTT during the TLS handshake.
Perfect Forward Secrecy (PFS)	This doesn't support PFS.	TLSv1.3 supports PFS. PFS ensures that each session key is completely independent of long-term private keys, which are keys that are used for an extended period to decrypt encrypted data.
Privacy	This is less secure, as the ciphers used are weak.	This is more secure, as the ciphers used are strong.
Performance	This has high latency and a less responsive connection.	This has low latency and a more responsive connection.

Advantages of TLSv1.3

The TLSv1.3 handshake offers the following improvements over earlier versions:

- All handshake messages after the ServerHello are encrypted.
- It improves efficiency in the handshake process by requiring fewer round trips than TLSv1.2. It also uses cryptographic algorithms that are faster.
- It provides better security than TLSv1.2, addressing known vulnerabilities in the handshake process.
- It eliminates data compression.

The following table describes the TLS versions supported on the client and server sides. The last column indicates which version will be used.

TLS Version Used

When NSSF is acting as a client or a server, it can support different TLS versions.

The following table provides information about which TLS version will be used when various combinations of TLS versions are present between the server and the client.

Table 4-5 TLS Version Used

Client Support	Server Support	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	TLS v1.3	TLSv1.3
TLSv1.2	TLSv1.2, TLSv1.3	TLSv1.2
TLSv1.2, TLSv1.3	TLSv1.2	TLSv1.2
TLS v1.3	TLSv1.2	Sends an error message. For more information about the error message, see "Troubleshooting TLS Version Compatibilities" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
TLSv1.2	TLSv1.3	Sends an error message. For more information about the error message, see "Troubleshooting TLS Version Compatibilities" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.

Note

- If Egress Gateway is deployed with both the versions of TLS that is TLSv1.2 and TLSv1.3, then Egress Gateway as client will send both versions of TLS in the client hello message during the handshake and the server needs to decide which version to be used.
- If Ingress Gateway is deployed with both the version of TLS that is with TLSv1.2 and TLSv1.3, then Ingress Gateway as the server will use the TLS version received from the client in the server hello message during the handshake.
- This feature does not work in ASM deployment.

Managing Support for TLSv1.2 and TLSv1.3

Enable:

This feature can be enabled or disabled at the time of NSSF deployment using the following Helm parameters:

- **enableIncomingHttps**: This flag is used for enabling/disabling HTTPS/2.0 (secured TLS) in the Ingress Gateway. If the value is set to false, NSSF will not accept any HTTPS/2.0 (secured) traffic. If the value is set to true, NSSF will accept HTTPS/2.0 (secured) traffic. **Note**: Do not change the `&enableIncomingHttpsRef` reference variable. For more information on enabling this flag, see the "Enabling HTTPS at Ingress Gateway" section in the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
- **enableOutgoingHttps**: This flag is used for enabling/disabling HTTPS/2.0 (secured TLS) in the Egress Gateway. If the value is set to false, NSSF will not accept any HTTPS/2.0 (secured) traffic. If the value is set to true, NSSF will accept HTTPS/2.0 (secured) traffic. For more information on enabling this flag, see the "Enabling HTTPS at Egress Gateway" section in the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

Configure

You can configure this feature using Helm parameters.

The following parameters in the Ingress Gateway and Egress Gateway microservices must be customized to support TLSv1.2 or TLSv1.3:

1. **Generate HTTPS certificates** for both the ingress and egress gateways. Ensure that the certificates are correctly configured for secure communication. After generating the certificates, create a Kubernetes secret for each gateway (egress and ingress). Then, configure these secrets to be used by the respective gateways. For more information about HTTPS configuration, generating certificates, and creating secrets, see the "Configuring Secrets for Enabling HTTPS" section in the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
2. After configuring the secret and applying it to the namespace where NSSF is deployed, perform the following Helm changes for Ingress and Egress gateways in the `ocnssf_custom_values_25.2.200.yaml` file:
   - **Parameters required to support TLSv1.2**:
     - `service.ssl.tlsVersion` indicates the TLS version.
     - `cipherSuites` indicates supported cipher suites.
     - `allowedCipherSuites` indicates allowed cipher suites.
   - **Parameters required to support TLSv1.3**:
     - `service.ssl.tlsVersion` indicates the TLS version.
     - `cipherSuites` indicates the supported cipher suites.
     - `allowedCipherSuites` indicates the allowed cipher suites.
     - `clientDisabledExtension` is used to disable the extension sent by messages originating from clients during the TLS handshake with the server.
     - `serverDisabledExtension` is used to disable the extension sent by messages originating from servers during the TLS handshake with the client.

- `tlsNamedGroups` is used to provide a list of values sent in the `supported_groups` extension. These are comma-separated values.
- `clientSignatureSchemes` is used to provide a list of values sent in the `signature_algorithms` extension.

For more information about configuring the values of the above-mentioned parameters, see the "Ingress Gateway Microservice" and "Egress Gateway Microservice" sections in the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

3. Save the `ocnssf_custom_values_25.2.200.yaml` file.
4. Install NSSF. For more information about the installation procedure, see the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.
5. Run Helm upgrade if you are enabling this feature after NSSF deployment. For more information about the upgrade procedure, see the *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

Note

- NSSF does not prioritize cipher suites based on priority. To select a cipher based on priority, you must list the cipher suites in decreasing order of priority.
- NSSF does not prioritize supported groups based on priority. To select a supported group based on priority, you must list the supported group values in decreasing order of priority.
- If you want to provide values for the `signature_algorithms` extension using the `clientSignatureSchemes` parameter, the following comma-separated values must be provided to deploy the services:
  - rsa_pkcs1_sha512
  - rsa_pkcs1_sha384
  - rsa_pkcs1_sha256

  Note

  By default, it is `null`.
- The mandatory extensions as listed in RFC 8446 cannot be disabled using the `clientDisabledExtension` attribute on the client or using the `serverDisabledExtension` attribute on the server side. The following is the list of the extensions that cannot be disabled:
  - supported_versions
  - key_share
  - supported_groups
  - signature_algorithms
  - pre_shared_key

Observe

Metrics

The following metrics are available for this feature:

- `oc_ingressgateway_incoming_tls_connections`
- `oc_egressgateway_outgoing_tls_connections`
- `security_cert_x509_expiration_seconds`

For more information about metrics, see NSSF Metrics section.

KPIs

There are no new KPIs for this feature.

Alerts

The following alerts are available for this feature:

- OcnssfTLSCertificateExpireMinor
- OcnssfTLSCertificateExpireMajor
- OcnssfTLSCertificateExpireCritical

> **Note**
>
> Alert gets raised for every certificate that will expire in the above time frame. For example, NSSF supports both RSA and ECDSA. So, we have configured two certificates. Accordingly, let us suppose RSA certificate is about to expire in 6 months in this situation only one alert will be raised and if both are about to expire then two alerts will be raised.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.8 Traffic Segregation

This feature provides end-to-end traffic segregation to NSSF based on traffic types. Within a Kubernetes cluster, traffic segregation can divide applications or workloads into distinct sections such as OAM, SBI, Kubernetes control traffic, etc. The Multus CNI container network interface (CNI) plugin for Kubernetes enables attaching multiple network interfaces to pods to help segregate traffic from each NSSF microservice.

This feature addresses the challenge of logically separating IP traffic of different profiles, which are typically handled through a single network (Kubernetes overlay). The new functionality ensures that critical networks are not cross-connected or sharing the same routes, thereby preventing network congestion.

With traffic segregation, operators can segregate traffic to external feeds and applications more effectively. Previously, all external traffic was routed through the same external network, but now, egress traffic from the NSSF pods can be directed through non-default networks to third-party applications. This separation is achieved by leveraging cloud-native infrastructure and the load balancing algorithms in OCCNE.

The feature supports the configuration of separate networks, Network Attachment Definitions (NADs), and the Cloud Native Load Balancer (CNLB). These configurations are crucial for enabling cloud native load balancing, facilitating ingress-egress traffic separation, and optimizing load distribution within NSSF.

> **Note**
>
> The Traffic Segregation feature is only available in NSSF if OCCNE is installed with CNLB.

Cloud Native Load Balancer (CNLB)

CNE provides Cloud Native Load Balancer (CNLB) for managing the ingress and egress network as an alternate to the existing LBVM, lb-controller, and egress-controller solutions. You can enable or disable this feature only during a fresh CNE installation. When this feature is enabled, CNE automatically uses CNLB to control ingress traffic. To manage the egress traffic, you must preconfigure the egress network details in the `cnlb.ini` file before installing CNE.

For more information about enabling and configuring CNLB, see *Oracle Communications Cloud Native Core, Cloud Native Environment User Guide*, and *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Network Attachment Definitions for CNLB

A Network Attachment Definition (NAD) is a resource used to set up a network attachment, in this case, a secondary network interface to a pod. NSSF supports two types of CNLB NADs:

1. **Ingress Network Attachment Definitions**
   Ingress NADs are used to handle inbound traffic only. This traffic enters the CNLB application through an external interface service IP address and is routed internally using interfaces within CNLB networks.
   - **Naming Convention:**`nf-<service_network_name>-int`
2. **Egress Only Network Attachment Definitions**
   Egress Only NADs enable outbound traffic only. An NF pod can initiate traffic and route it through a CNLB application, translating the source IP address to an external egress IP address. An egress NAD contains network information to create interfaces for NF pods and routes to external subnets.
   - **Requirements:**
     - Ingress NADs are already created for the desired internal networks.
     - Destination (egress) subnet addresses are known beforehand and defined under the `cnlb.ini` file's `egress_dest` variable to generate NADs.
     - The use of an Egress NAD on a deployment can be combined with Ingress NADs to route traffic through specific CNLB apps.
   - **Naming Convention:**`nf-<service_network_name>-egr`

Managing Ingress and Egress Traffic Segregation

Enable:

This feature is disabled by default. To enable this feature, you must configure the network attachment annotations in the `ocnssf_custom_values_25.2.200.yaml` file.

Configuration

For more information about Traffic Segregation configuration, see " Configuring Traffic Segregation" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide.*.

Observe

There are no Metrics, KPIs, or Alerts available for this feature.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.9 Support for cnDBTier APIs in CNC Console

NSSF earlier fetched the status of cnDBTier using the CLI-based mechanism.

With the implementation of this feature, cnDBTier APIs are integrated into the CNC Console. NSSF users can view the specific cnDBTier APIs directly on the CNC Console as mentioned in the following tables:

Table 4-6 cnDBTier APIs

Console Parameter	cnDBTier API name	Description
Backup List	`http://<base-uri>/ocdbtier/list/backups`	This is a read-only API. This API displays the details of completed backups along with backup ID, backup creation timestamp, and backup size.
Backup Manager Health Status	`http://<base-uri>/{unique_prefix}/ocdbtier/health-info/backup-mgr-svc/status`	This is a read-only API. This API displays the health status of the backup manager service. It checks the following: • if the backup manager service is up or not • if the service can connect to database or not

Table 4-6 (Cont.) cnDBTier APIs

Console Parameter	cnDBTier API name	Description
cnDBTier Backup Status	`http://<base-uri>/ocdbtier/backup/status`	This is a read-only API. This API displays the current backup status of the cnDBTier. It displays the following: • current system time • current cnDBTier backup status • next scheduled backup time
cnDBTier Health	`http://<base-uri>/{unique_prefix}/ocdbtier/health-info`	This API lists the health info of cnDBTier pods.
cnDBTier Version	`http://<base-uri>/ocdbtier/version`	This is a read-only API. This API displays the cnDBTier version.
Database Statistics Report	`http://<base-uri>/ocdbtier/statistics/report/dbinfo`	This is a read-only API. This API displays the number of available database.
Georeplication Recovery Status	`http://<base-uri>/ocdbtier/faultrecovery/status`	This is a read-only API. This API is used to monitor the recovery status of georeplication for both FAILED and ACTIVE cnDBTier sites.
Georeplication Status	`http://<base-uri>/ocdbtier/status/replication/realtime`	This API displays the georeplication status.
Georeplication Status Across All Sites	`http://<base-uri>/ocdbtier/replication/status/realtime`	This is a read-only API. This API is used to retrieve the georeplication status across all the configured sites, offering a consolidated view of the overall replication status.
Local Cluster Status	`http://<base-uri>/ocdbtier/status/cluster/local/realtime`	This is a read-only API. This API displays the status of the local cluster.
Monitor Health Status	`http://<base-uri>/{unique_prefix}/ocdbtier/health-info/monitor-svc/status`	This is a read-only API. This API displays the health status of the monitor service. It checks the following: • if the monitor service is up or not • if the service can connect to database or not • if the metrics are fetched or not (the metrics are fetched when the service is up and vice versa)

Table 4-6 (Cont.) cnDBTier APIs

Console Parameter	cnDBTier API name	Description
NDB Health Status	`http://<base-uri>/{unique_prefix}/ocdbtier/health-info/ndb-svc/status`	This is a read-only API. This API displays the health status of the NDB service pods such as (data pods, sql pods, app-my-sql pods, mgmt pods). It checks if the pods status is up or not.
On Demand Backup	`http://<base-uri>/ocdbtier/on-demand/backup/initiate`	This API is used to initiate a new backup. This API displays the status of initiated on-demand backups and helps to create a new backup.
Real Time Overall Replication Status	`http://<base-uri>/ocdbtier/status/replication/realtime`	This is a read-only API. This API displays the overall replication status in multisite deployments. For example, in a four-site deployment, it provides the replication status between the following sites: site1-site2, site1-site3, site1-site4. This is applicable for all other sites, such as, site 2, site 3, and site 4.
Replication Health Status	`http://<base-uri>/{unique_prefix}/ocdbtier/health-info/replication-svc/status`	This is a read-only API. This API displays the health status of the replication service. It checks the following: • if the replication service is up or not • if the replication service can connect to database or not
Replication HeartBeat Status	`http://<base-uri>/ocdbtier/heartbeat/status`	This is a read-only API. This API displays the connectivity status between the local site and the remote site name to which NSSF is connected.
Site Specific Real Time Replication Status	`http://<base-uri>/ocdbtier/status/replication/realtime/{remoteSiteName}`	This is a read-only API. This API displays the site-specific replication status.
Start Georeplication Recovery	`http://<base-uri>/ocdbtier/faultrecovery/start`	This API is used to start the georeplication recovery process.
Update Cluster As Failed	`http://<base-uri>/ocdbtier/markcluster/failed`	This API is used to mark a disrupted cnDBTier cluster as failed.

For more information about the above-mentioned cnDBTier APIs, see *Oracle Communications Cloud Native Core, cnDBTier User Guide*.

Managing cnDBTier APIs in CNC Console

Enable

The CNC console section for accessing the cnDBTier APIs is available in NSSF if the cnDBTier is configured as an instance during the CNC Console deployment. For more information about integrating cnDBTier APIs in CNC Console, see the "NF Single Cluster Configuration With cnDBTier Menu Enabled" section in *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*.

There is no option to disable this feature.

Configure

cnDBTier APIs can be accessed or configured using the CNC Console. For more information, see cnDBTier APIs in the CNC Console.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.10 Support for Common Service APIs in CNC Console

The configuration for common service APIs was supported only using REST. With the implementation of this feature, NSSF now supports the configuration of Ingress Gateway and Egress Gateway parameters using the CNC Console. You can perform HTTP methods such as GET and PUT using the Console.

For more information about the common service APIs, see "Common Services REST APIs" section in the *Oracle Communication Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

Managing common service APIs in the CNC Console

Enable

This feature is enabled automatically along with the NRF instance deployment.

Configure

You can configure the common services APIs in the CNC Console. For more information, see Common Services Configuration section.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.

2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.11 Enhanced Computation of AllowedNSSAI in NSSF

As per 3GPP specification, NSSF utilizes AMF (Access and Mobility Management Function) to manage the slice selection criteria. Here, if a newly configured NSSAI is not supported by the Radio Access Network (RAN), the AMF designates it as the Rejected-NSSAI. To allow the newly configured NSSAI, UE needs to request for a re-registration. It allows the AMF to disregard a UE's Requested NSSAI and formulate the AllowedNSSAI based on subscription details and support from the RAN

The objective of this feature is to maintain a consistent approach to handling slice subscription changes between AMF and NSSF. Recognizing the need for adaptive and subscriber centric configurations of NSSAI, this feature allows NSSF to generate AllowedNSSAI based on individual user equipment (UE) subscriptions and Tracking Area characteristics.

When this feature is enabled, NSSF generates AllowedNSSAI based on User Equipment (UE) subscription details and the supported configuration within the Tracking Area. Conversely, when the feature is disabled, the NSSF adheres to the 3GPP specification 29.531, constructing AllowedNSSAI based on the intersection of Requested and Subscribed NSSAI and also operator policy.

NSSAI Configuration (When the feature is enabled):

- **Subscriber-Driven Configuration**: The NSSF will dynamically build AllowedNSSAI based on the UE subscription information. This ensures that network slices are aligned with the specific requirements and preferences of individual subscribers.
- **Tracking Area Considerations**: The NSSF takes into account the supported configuration within the Tracking Area, optimizing the network slice selection based on the geographical location and network capabilities in real time.

Static NSSAI Configuration (When the feature is disabled):

- **Compliance with 3GPP Specification 29.531**: When the feature is disabled, the NSSF follows the guidelines outlined in the 3GPP specification 29.531. This involves creating AllowedNSSAI based on the intersection of Requested and Subscribed NSSAI, providing a standardized approach to network slice configuration.
- **Maintaining Compatibility**: Disabling the dynamic NSSAI configuration ensures compatibility with industry standards and facilitates interoperability across different network elements and vendors.

Managing Enhanced Computation of AllowedNSSAI in NSSF

Enable:

You can enable this feature using REST API or CNC Console.

Enable using REST API

1. Use the following API path:
   {apiRoot}/nnsf-configuration/v1/systemoptions/
2. To enable the feature for a specific PLMN, set the `enhancedAllowedNssaiEnable` parameter to `true`. The default value for this parameter is `false`.
3. Run GET service method to get the `enhancedAllowedNssaiEnable` value for the PLMN.
4. Run PUT service method to update the `enhancedAllowedNssaiEnable` value for the PLMN.

For more information about API path, see "SystemOptions" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

Enable using CNC Console

For more information, see NSSF System Option section.

Observe

Metrics

There are no new metrics for this feature.

KPIs

There are no new KPIs for this feature.

Alerts

There are no alerts generated for this feature.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.12 LCI and OCI Headers

Within the complex 5G architecture, network overload scenarios are common. The exchanges of data between producer and consumer Network Functions (NFs) often involve significant message and notification volumes, necessitating a precise approach to load balancing. This is imperative to prevent network failures triggered by overload conditions.

In such scenarios, it becomes crucial for consumer NFs to be promptly notified when the producer NF approaches an overloaded state. This enables consumer NFs to implement corrective actions proactively.

To address these challenges, the introduction of LCI and OCI Headers plays a pivotal role in optimizing communication between NSSF and its consumer NFs. They provide consumer NFs with real-time insights into the operational status of the NSSF resources, facilitating efficient traffic management.

These headers provide essential load and overload information for consumer NFs to optimize traffic distribution and take proactive measures during network overload scenarios.

The headers are integrated into outgoing responses, based on load levels at the Ingress Gateway. They server as communication tools and allow Network Functions (NFs) to share crucial load information, ensuring an architecture where 5G Core Network remains stable and high performing even during heavy load conditions.

LCI Header

The LCI header comprises overall load related information such as the timestamp of load data generation, the current load of the NF, and the scope of the load information. For example, there are two LCI headers:

- NF scope LCI header
- NF-service scope LCI header

Examples of LCI Headers

NF Scope LCI Header:

```
3gpp-sbi-lci: Timestamp: "Tue, 19 Sep 2023 13:47:41 UTC"; Load-Metric: 1%; NF-Instance: 9faf1bbc-6e4a-4454-a507-aef01a101a01
```

Service Scope LCI Header:

```
3gpp-sbi-lci: Timestamp: "Mon, 25 Sep 2023 11:30:17 UTC"; Load-Metric: 0%; NF-Service-Instance: ae870316-384d-458a-bd45-025c9e748976
```

OCI Header

The OCI header communicates information about overload conditions. This information encompasses the timestamp when the overload condition was detected, Overload-Reduction-Metric, and Overload Validity period.

Examples of OCI Headers

NF Scope OCI Header:

```
3gpp-Sbi-Oci:Timestamp: "Mon, 02 May 2022 07:43:48 UTC"; Period-of-Validity: 30s; Overload-Reduction-Metric: 5.0%; NF-Instance: 5a7bd676-ceeb-44bb-95e0-f6a55a328b03
```

Service Scope OCI Header:

```
3gpp-Sbi-Oci:Timestamp: "Mon, 02 May 2022 07:43:48 UTC"; Period-of-Validity: 30s; Overload-Reduction-Metric: 5.0%; NF-Service-Instance: 5a7bd676-ceeb-44bb-95e0-f6a55a328b03
```

Both the LCI and OCI headers are incorporated in HTTP response messages without triggering additional signaling, ensuring a more efficient communication process. Here is how they help:

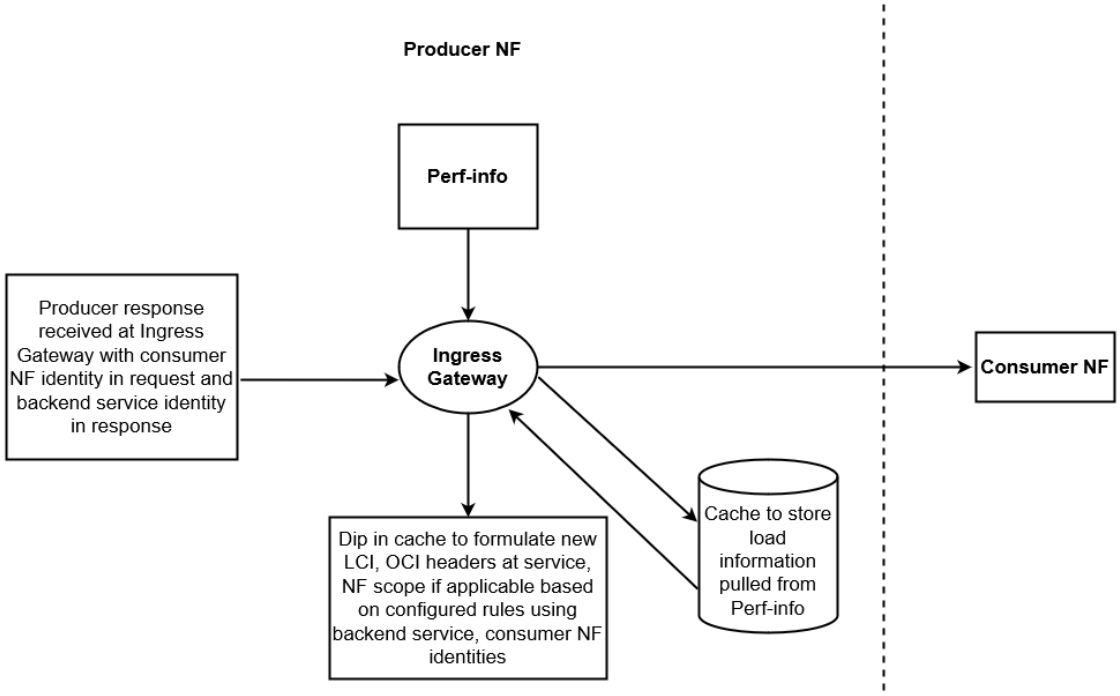
- The LCI header conveys the overall load of an NF, assisting in decisions regarding the acceptance or rejection of new requests to prevent further overload.
- In contrast, the OCI header communicates specific overload conditions, helping NFs take informed actions to mitigate these conditions.
- The LCI and OCI headers complement each other, allowing an NF to reduce the number of requests it sends to other NFs in response to an OCI header, even if it's not yet overloaded.
- This proactive measure prevents overload conditions from replicating.

Use Cases

- As of now, LCI and OCI Headers are supported at the Ingress Gateway only; not at the Egress Gateway.
- NSSF can utilize the LCI header to share its load information with the AMF (Access and Mobility Management Function). This information allows the AMF to make informed decisions about directing new User Equipment (UE) connections to the NSSF.
- Conversely, the NSSF can employ the OCI header to notify the Radio Access Network (RAN) of overload conditions, enabling the AMF to reduce the number of UEs it routes to NSSF.

> **Note**
>
> Currently, this feature supports LCI-OCI headers at Ingress Gateway. The support for LCI and OCI headers in outgoing messages from Egress Gateway will be enabled in the future.

This following diagram depicts the LCI and OCI workflow:



LCI Headers and Their Workflow

LCI Headers are dedicated to providing real-time load information, using the identifier "3gpp-Sbi-Lci." This information provides consumer NFs with the essential knowledge needed to make informed decisions about distribution of network traffic efficiently. When specific conditions are met, LCI headers are included in the messages and notifications (notifications will be supported in future releases; not supported yet), extending the scope to network function, backend service, or both. The condition states that user-agent/via/oauth header should be present in the request so that LCI is included in the response header.

When NSSF receives a request from an NF (for example, AMF), Ingress Gateway at NSSF checks if LCI is enabled. If it is not enabled, LCI header is not added as part of response. If it is enabled, LCI header is computed as follows:

- Ingress Gateway fetches the previous LCI Header from the cache.

> **Note**
>
> If no data is fetched regarding LCI Header from the cache, it indicates that LCI Header was not sent more than N seconds ago. In this case, a fresh LCI Header with the current information is included.

- If the current load metric is within the load metric threshold, LCI Header is not included in the message.
- If it is beyond the threshold, LCI Header is included in the message. The LCI Header information is updated in the coherence cache for the current destination and further processing is done.

Table 4-7 LCI header fields

Fields	Description
Load Control Timestamp	Human readable timestamp (date and time) indicating time when LCI header is sent.
Load Metric	Current load level for the scope of LCI, in terms of percentage, ranging from 0 to 100. 0 indicated 0% or no load, and 100 indicates 100% or maximum load.
Scope of LCI	The scope of LCI are NF Instance ID, NF Set ID, NF Service Instance ID or NF Service Set ID. This scope depends on the scope of load info received from perf-info. The same scope is conveyed to the consumer NF.

OCI Headers and Their Workflow

OCI Headers operate under the identifier "3gpp-Sbi-Oci" and are designed to convey current overload information. This information is critical for consumer NFs, enabling them to take preemptive actions to reduce the traffic directed toward overloaded NFs. Like LCI headers, OCI headers are included in messages and notifications when specific conditions are met, spanning the NF scope, the backend service, or both.

When NSSF receives a request from an NF (for example, AMF), Ingress Gateway at NSSF checks if OCI is enabled. If it is not enabled, OCI header is not added as part of response. If it is enabled, OCI header is computed as follows:

- Ingress Gateway checks if the NF is overloaded as per the configured range. Then, it fetches the previous OCI Header sent from the cache.

> **Note**
>
> If no data is fetched regarding OCI Header, however, the NF is overloaded, even then OCI Header is prepared.

- If the overload is different from the previously computed value, the Ingress Gateway changes the overload reduction metric value and prepares a new OCI Header.
- If the overload is the same as the previous reported value, but the period of validity has expired, the OCI Header is included in the message.
- If the period of validity has not expired, the Ingress Gateway continues with further processing. The OCI Header information is updated in the coherence cache for the current destination and further processing is done.

Overload Control based on OCI Header
Overload information need to be sent from producer NF to consumer NF in "3gpp-Sbi-Oci" header. The fields included in this header are described below:

Table 4-8 OCI header fields

Fields	Description
Overload Control Timestamp	Human readable timestamp (date and time) indicating time when OCI header is sent.
Overload Reduction Metric	Indicates the percentage of traffic reduction that this NF expects from the receiver of OCI Header. The value ranges from 0-100, 0 indicating no traffic reduction towards sender of OCI.
Overload Control Period of Validity	Indicates a timer within which the information conveyed in OCI Header shall be considered valid (unless overridden by a newer OCI Header)
Scope of OCI	The scope of OCI can be one of below: • NFInstanceID or NF-SET or NF-Service-instance or NF-Service-Set (if NF is a producer) • NFInstanceID or NF-SET or NF-Service-instance or NF-Service-Set or Call-Back URI (if NF is a consumer) The scope depends on the scope of load info received from perf-info. The same scope is conveyed to the destination NF.

Load Computation

The Ingress Gateway features a configurable polling interval used to retrieve service-level load information from perf-info. The Ingress Gateway conducts load aggregation for supported NSSF services at the NF (Network Function) level, using a straightforward averaging logic. The supported services are decided based on the NF service to instance ID mapping, which is done through Helm. For example, in NSSF, the mapping is done for NsSelection and NsAvailability services. Hence, only these two services are supported in NSSF.

For instance, when the Ingress Gateway receives two pieces of load information for a particular service, it adds these values together and then divides the sum by two to calculate the average load at the NF level.

Peer Identity

If multiple fields are available to extract peer identity, the priority to extract this identity will be in below order:

1. OAuth token
2. User Agent header
3. VIA header

Validity Period

If the same peer sends multiple requests within the validity period and there no breach of configured thresholds, then NSSF will not add LCI or OCI headers.

Managing LCI and OCI Headers

Enable:

You can enable LCI and OCI Headers by performing the following Helm configurations globally and at the Ingress Gateway:

- **Global Helm Configuration**

- **Enable**: You can enable LCI and OCI Headers globally at the Network Function (NF) level by setting the values of `nssfLciEnabled` and `nssfOciEnabled` parameters as `true`, respectively.

```
global:
#======================LCI/OCI header Global
Values==========================================
  #Enabling LCI
  nssfLciEnabled: &lcienable true
  #Enabling OCI
  nssfOciEnabled: &ocienable true
 #====================================================================
```

- **Ingress Gateway Helm Configuration**
  - **Enable**: You can enable LCI and OCI Headers globally at Ingress Gateway level by setting the `lciHeaderConfig.enabled` and `ociHeaderConfig.enabled` parameters as `true`, respectively.
  - **Configure**: You can configure LCI and OCI Headers at Ingress Gateway using the Helm based configuration:

```
## This is mandatory for LCI and OCI feature as this is required by
perf-info service to get the load information of the services from
prometheus
perf-info:
  configmapPerformance:
      prometheus: http://occne-prometheus-server.occne-infra:80


ingress-gateway:
 #To remove the Producer header from Ingress Response when LCI is
enabled
  globalRemoveResponseHeader:
  - name: *producer
  # ********  Sub-Section Start: LCI/OCI Ingress Gateway Parameters
************

#*********************************************************************
***

  # Engineering Parameter Start
  global:
    lciHeaderConfig:
      enabled: *lcienable
      # difference between previous threshold and current threshold for
lci header will be added when the difference crosse the mentioned value
      loadThreshold: 30
      # Validity period after which lci header will be added to reponse
header if delta of threshold is not breached
      localLciHeaderValidity: 60000 #(value in milliseconds)
      ## This header needs to be same which is being sent along with
request in microservice
      producerSvcIdHeader: *producer

    ociHeaderConfig:
      enabled: *ocienable
```

```
      ## This header needs to be same which is being sent along with
reuest  microservice
      producerSvcIdHeader: *producer
      validityPeriod: 10000 #(value in milliseconds)
      ## The range of the cpu load for which the ingress gateway will
get notified regarding the criticality of the load.
      overloadConfigRange: #Note - minor, major and critical conditions
should cover complete range of 0 to 100 both inclusive for it to be a
valid config
        minor: "[75-80]"
        major: "[81-90]"
        critical: "[91-100]"
      ## The range of the cpu load which needs to be decreased from the
consumer when a particular criticality has reached.
      reductionMetrics:
        minor: 5  #(Possible values 1 to 9 both inclusive)
        major: 15  #(Possible values 5 to 15 both inclusive)
        critical: 25 #(Possible values 10 to 50 both inclusive)

    nfInstanceId:  "9faf1bbc-6e4a-4454-a507-aef01a101a01"
      ## This is a mapping for service name to service instance id for
which service the LCI and OCI headers needs to be enabled. Default
values are given with 'ocnssf' as release name. It must be  configured
by operator in case release name is changed.
      ## only nsselection and nsavailability services should be
configured. As these are the two services for which  LCI/OCI headers is
supported.
      ## Format is <releaseName>-<serviceName> . Eg: if release name is
given 'ocnssf-test' then svc name should be 'ocnssf-test-nsselection'
and 'ocnssf-test-nsavailability'
    svcToSvcInstanceIdMapping:
      - svcName: ocnssf-nsselection
        serviceInstanceId: "ae870316-384d-458a-bd45-025c9e748976"
      - svcName: ocnssf-nsavailability
        serviceInstanceId: "ae870316-384d-458a-bd45-025c9e748996"

    perfInfoConfig:
      ## the interval when perf-info will fetch the cpu load from
prometheus
      pollingInterval: 5000 #(value in milliseconds)
      serviceName: "ocnssf-perf-info"
      port: 5905
      perfInfoRequestMap: "/load"
  # Engineering Parameter End

  # ********  Sub-Section End: LCI/OCI Ingress Gateway Parameters
************

  # ********  Sub-Section Start: DB credentials Ingress Gateway
Parameters ************

#*********************************************************************
***
  dbConfig:
    dbHost: *dbHost
    dbPort: *dbPort
```

```
    secretName: *privDbSecret
    dbName: *provDB
    # Name of the Key configured for "DB Username" in Secret with
following name: "<dbConfig.secretName>"
    dbUNameLiteral: mysql-username
    # Name of the Key configured for "DB Password" in Secret with
following name: "<dbConfig.secretName>"
    dbPwdLiteral: mysql-password
    # Default is NDBCLUSTER
    dbEngine: *dbEngine

  # ********  Sub-Section End: DB credentials Ingress Gateway
Parameters ************

#*************************************************************************
***
```

For more information about the Helm parameters, see "Customizing NSSF" section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

Observe

Metrics

There are no new metrics for this feature.

KPIs

There are no new KPIs for this feature.

Alerts

There are no alerts generated for this feature.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.13 Server Header in NSSF

One of the core functionalities of the Network Slice Selection Function (NSSF) is to manage the security aspects of the 5G network. As part of this role, NSSF handles various requests from other network functions (NFs) and network entities over the HTTP protocol. On receiving these requests, NSSF validates and processes them before issuing responses to the requesting NFs or network entities.

In such scenarios, NFs and other network entities may encounter issues resulting in error responses. It becomes imperative for consumer NFs to pinpoint the source of the error, so they can undertake troubleshooting and corrective measures. The integration of this feature at NSSF helps to determine the originator of the error response.

This feature offers the support for Server Header in NSSF responses, which contain crucial information about the origin of an error response and the type of the error encountered. Thus, the Server Header enhances the behavior of NSSF while responding to requests, particularly the error responses.

> **Note**
>
> This feature is applicable in scenarios where NSSF generates error responses. It does not affect normal response behavior.

A Server Header starts with the value of NF Type, followed by a "-" and any other specific information, if needed, afterward. It is expected to be present in all NSSF responses in the following format:

`<NF Type>-<Instance-Id>`
Where,

- `<NF Type>` is the type of the NF.
- `<NF Instance-Id>` is the unique identifier of the NF instance generating the response.

For example, the following combinations are applicable to NSSF:

`NSSF-<NSSF's Instance-Id>`

Where,

- `NSSF` is the <NF Type>.
- `<NSSF's Instance-Id>` is the unique identifier of the NSSF instance generating the response.

Managing Server Header in NSSF

Enable:

You can enable this feature using REST API or CNC Console.

Enable using REST API

By default, this feature is disabled. To enable it, REST API needs to be invoked and the `enabled` flag needs to be updated to `true` in the following URI:

/{nfType}/nf-common-component/v1/{serviceName}/serverheaderdetails

Example

Example of Request or Response Body to Enable Server Header:

```
{
  "enabled": true,
  "errorCodeSeriesId": "E1",
  "configuration": {
    "nfType": "NSSF",
    "nfInstanceId": "9faf1bbc-6e4a-4454-a507-aef01a101a01"
  }
}
```

Example of Request or Response Body to Disable Server Header:

```
{
  "enabled": false,
  "errorCodeSeriesId": "E1",
  "configuration": {
    "nfType": "NSSF",
    "nfInstanceId": "9faf1bbc-6e4a-4454-a507-aef01a101a01"
  }
}
```

Note

- `enabled` is used to enable or disable the feature.
- `nfType` and `nfInstanceId` are used to form Server Header.
- In the mentioned configuration, when sending a response to AMF, the Server Header will be appended by the NSSF with the value "`NSSF-9faf1bbc-6e4a-4454-a507-aef01a101a01`"
- The values in the above example are samples. Ensure that you update the values of the following parameters according to your deployment:
  - `nfType` must be NSSF.
  - `errorCodeSeriesId`: A valid configured value.
  - `nfInstanceId`: NSSF's valid instance value. It must be same as NSSF's instance ID.

Enable using CNC Console

For more information, see Server Header Details.

Configure

Perform the REST API or CNC Console configurations in the following sequence to configure this feature:

1. Configure **errorcodeserieslist** to update the **errorcodeserieslist** that are used to list the configurable exception or error for an error scenario in Ingress Gateway.
2. Configure **serverheaderdetails** to enable the feature.
3. <Optional>Configure **routesconfiguration** to map route ID and its corresponding route-level configuration.

Note

If this configuration is done for Server Header, then for this particular route, it will take precedence over the **serverheaderdetails** configuration.

> **Note**
>
> - If **routesconfiguration** is done without **errorCodeSeriesId** then **errorCodeSeriesId** configured at **serverheaderdetails** is picked up, if Server Header is enabled there.
> - If Server Header is enabled without configuring **errorCodeSeriesId** then it will be applied for all error codes. This is not a recommended configuration.

For more details about REST APIs, see "REST API Configurations for Server Header Feature" in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

Observe

Metrics

There are no new metrics for this feature.

KPIs

There are no new KPIs for this feature.

Alerts

There are no alerts generated for this feature.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.14 Support for User-Agent Header

In 5G networks, producer Network Functions (NFs) cannot identify or validate a consumer on their own. To overcome this, 3GPP has introduced User-Agent headers, which are added to consumer service requests. This field is included in the HTTP (Hypertext Transfer Protocol) request that a consumer sends to the producer to identify itself and provide information about the NF making the request.

This feature enables the usage of the User-Agent Header in NSSF.

NSSF support the following inter-NF communication and service request functionalities:

- NSSF sends notifications to AMF.
- NSSF sends registration and heartbeat request to NRF.

This enhancement enables NSSF to include the User-Agent Header in every HTTP/2 request that it sends over any Service Based Interface (SBI) to a producer NF (for example, AMF and NRF). The User-Agent Header in NSSF's HTTP/2 requests helps a producer NF to identify NF type of client that has sent a request. Here, it helps:

- AMF in identifying the NSSF that sent the notification.

- NRF in identifying the NSSF that sent the subscription, registration, or heartbeat request to NRF.

Structure of an User-Agent Header

An User-Agent Header starts with the value of NF type, followed by a "-" and any other specific information, if needed afterwards. It is expected to be present in all the service requests and notification in the following formats:

- <NF Type>
- <NF Type>-<Instance-Id>
- <NF Type>-<Instance-Id> <FQDN>

Where,

- <NF Type> is the type of the NF.
- <Instance-Id> is the instance ID of the NF.
- <FQDN> is the FQDN of the NF.

For example: The following combinations are applicable to NSSF:

NSSF

NSSF-<NSSF's Instance-Id>

NSSF-<NSSF's Instance-Id> <NSSF's FQDN>

When the User-Agent Header is not included in the incoming requests sent to AMF or NRF, the corresponding metric cannot gather information about the origin of the service request. Nevertheless, the request is still processed successfully without any problems, but the AMF or NRF are not able to identify the NSSF from which the request has originated.

> **Note**
>
> The onus is on operator to configure the values correctly as defined in the syntax explained above.

Managing the Support for User-Agent Header

Enable:

You can enable this feature using REST API or CNC Console.

Enable using REST API

1. Use the following API path:
   /{nfType}/nf-common-component/v1/{serviceName}/useragentheader
2. Set enabled as `true`.
3. Run the API using PUT method with the proposed values given in the Rest API. For more information about API path, see "Configurations to Enable or Disable User-Agent Header" section of "Egress Gateway REST APIs" in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.
   Given below is a sample REST API configuration to enable this feature:

```
{
   "enabled": true,
```

```
   "nfType": "NSSF",
   "nfInstanceId": "9faf1bbc-6e4a-4454-a507-aef01a101a01",
"nfFqdn": "nssf.oracle.com",
"addFqdnToHeader": true,
 "overwriteHeader": true
}
```

> **Note**
>
> - In the mentioned configuration, when sending notifications to AMF, the User-Agent Header will be appended by the NSSF with the value `NSSF-9faf1bbc-6e4a-4454-a507-aef01a101a01 nssf.oracle.com`.
> - The `nfInstanceId` and `nfFqdn` values in the above example are samples. Ensure that you update the values of the `nfInstanceId` and `nfFqdn` parameters accordingly.

Enable using CNC Console

For more information, see User Agent Header Generation.

Observe

Metrics

The following metric is used to provide information about this feature:

- `oc_egressgateway_user_agent_consumer_total`: This metric is applicable whenever the feature is enabled and User-Agent Header is getting generated.

For information about the metrics, see Egress Gateway Metrics.

Alerts

There are no alerts generated for this feature.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.15 Monitoring the Availability of SCPs using SCP Health APIs

With the introduction of this feature, NSSF determines the availability and reachability status of all the SCPs configured either statically by the operator or through DNS SRV based selection of the SCP sets. With this feature, NSSF determines the availability and reachability status of all SCPs irrespective of the configuration types. This feature is an enhancement to the existing SBI routing functionality. Egress Gateway microservice interacts with SCP on their health API endpoints using HTTP2 OPTIONS method. It monitors the health of configured SCP peers to ensure that the traffic is routed directly to the healthy peers. This enhancement avoids routing or rerouting towards unhealthy peers, thus minimizing the latency time.

Egress Gateway microservice maintains the health status of all available and unavailable SCPs. It maintains the latest health of SCPs by periodically monitoring and uses this data to route egress traffic to the healthy SCP.

> **Note**
>
> - This is not a standalone feature but an add-on to the existing SBI Routing feature, which means this feature is activated only if the SBI Routing feature is enabled.
> - Health monitoring can only be enabled for the peers which belong to a peerset associated with a SBI Routing filter.

Managing Monitoring the Availability of SCPs using SCP Health APIs

Prerequisites

During the installation, `peermonitoringconfiguration` is set to `false` by default. Since this feature is an add-on to the existing SBI Routing feature, it will be activated if the `sbirouteconfig` is enabled. To enable this feature, perform the following:

Configure Using REST API

You can also enable this feature using the REST API configurations at Egress Gateway in the following sequence:

1. Configure `peerconfiguration` to define the list of peers to which Egress Gateway can send request.

   > **Note**
   >
   > `peerconfiguration` must consist of `healthApiPath` even though `peermonitoringconfiguration` is set to `false` by default. Configure `virtualHost` under `peerconfiguration` where the AMF query is sent.

   Here is a sample configuration:

   PUT Request

   ```
   curl -v -X PUT "http://{{host}}:{{port}}/nssf/nf-common-component/v1/egw/
   peerconfiguration" -H  "Content-Type: application/json"  --data-raw
   '[{"id": "peer1","host": "scp1","port": "8080","apiPrefix":
   "/","healthApiPath":"/health/v1"},{"id": "peer2","host": "scp2","port":
   "8080","apiPrefix":"/","healthApiPath":"/health/v2"},{"id":
   "peer3","host": "scp3","port": "8080","apiPrefix": "/","healthApiPath":"/
   health/v3"},{"id": "peer4","host": "scp4","port": "8080","apiPrefix":
   "/","healthApiPath":"/health/v4"},{"id": "peer5","virtualHost":
   "xyz.test.com","apiPrefix": "/","healthApiPath":"/health/v5"},{"id":
   "peer6","virtualHost": "abc.test.com","apiPrefix": "/","healthApiPath":"/
   health/v6"}]'
   ```

2. Configure `peersetconfiguration` to logically group the peers into sets.

> **Note**
>
> - `peerIdentifier` must be the value of SCP peer configured in `peerConfiguration`.
> - You cannot configure multiple virtual hosts as peers in the same peer set.
> - Configure the `priority` for each SCP peer in the set. Depending on the priority, it selects the primary, secondary, or tertiary SCP peers to route requests.

Here is a sample configuration:

PUT Request

```
curl -v --http2-prior-knowledge -X PUT "http://{{host}}:{{port}}/nssf/nf-
common-component/v1/egw/peersetconfiguration" -H  "Content-Type:
application/json" -d '[{"id":"set0","httpConfiguration":[{"priority":
1,"peerIdentifier": "peer1"},{"priority": 2,"peerIdentifier": "peer2"},
{"priority": 3,"peerIdentifier": "peer3"},{"priority": 4,"peerIdentifier":
"peer4"}],"httpsConfiguration":[{"priority": 1,"peerIdentifier": "peer1"},
{"priority": 2,"peerIdentifier": "peer2"},{"priority": 3,"peerIdentifier":
"peer3"},{"priority": 4,"peerIdentifier": "peer4"}]},
{"id":"set1","httpConfiguration":[{"priority": 1,"peerIdentifier":
"peer5"}],"httpsConfiguration":[{"priority": 1,"peerIdentifier":
"peer6"}]}]'
```

3. Configure or update `errorcriteriasets`.
   Here is a sample configuration:

   PUT Request

   ```
   curl -v --http2-prior-knowledge -X PUT "http://{{host}}:{{port}}/nssf/nf-
   common-component/v1/egw/sbiroutingerrorcriteriasets" -H  "Content-Type:
   application/json" -d '[{"id":"criteria_0","method":
   ["GET","POST","PUT","DELETE","PATCH"],"exceptions":
   ["java.util.concurrent.TimeoutException","java.net.UnknownHostException"]},
   {"id":"criteria_1","method":
   ["GET","POST","PUT","DELETE","PATCH"],"response":{"statuses":
   [{"statusSeries":"4xx","status":[400,404]},{"statusSeries":"5xx","status":
   [500,503]}]}}]'
   ```

4. Configure or update `erroractionsets`.
   Here is a sample configuration:

   PUT Request

   ```
   curl -v --http2-prior-knowledge -X PUT "http://{{host}}:{{port}}/nssf/nf-
   common-component/v1/egw/sbiroutingerroractionsets" -H  "Content-Type:
   application/json" -d
   '[{"id":"action_0","action":"reroute","attempts":3,"blacklist":
   {"enabled":false,"duration":60000}},
   {"id":"action_1","action":"reroute","attempts":3,"blacklist":
   {"enabled":false,"duration":60000}}]'
   ```

5. Configure `routesconfiguration` to define the route and reroute parameters.

> **Note**
>
> - The configuration under `sbiRoutingConfiguration` corresponds to the SBI-Routing specific configuration.
> - If SBIRouting functionality is required, then configure `SBIRoutingFilter`. If reroute mechanism is required for that route, then configure `SBIReroute` filter with retries, methods, and statuses.
> - `peerSetIdentifier` must be the value configured during `peersetconfiguration`.

Here is a sample configuration:

PUT Request

```
curl -v --http2-prior-knowledge -X PUT "http://{{host}}:{{port}}/nssf/nf-
common-component/v1/egw/routesconfiguration" -H  "Content-Type:
application/json" -d '[{"id":"egress_scp_proxy1","uri":"http://
localhost:32068/","order":0,"metadata":
{"httpsTargetOnly":false,"httpRuriOnly":false,"sbiRoutingEnabled":true},"pr
edicates":[{"args":{"pattern":"/notification/
amf2/"},"name":"Path"}],"filters":[{"name":"SbiRouting","args":
{"peerSetIdentifier":"set0","customPeerSelectorEnabled":true,"errorHandling
":[{"errorCriteriaSet":"criteria_1","actionSet":"action_1","priority":1},
{"errorCriteriaSet":"criteria_0","actionSet":"action_0","priority":2}]}}]},
{"id": "default_route","uri": "egress://request.uri","order":
100,"filters": [{"name": "DefaultRouteRetry"}],"predicates": [{"args":
{"pattern": "/**"},"name": "Path"}]}]'
```

6. Set the value of `enabled` under `sbiRoutingConfiguration` to `true` to route the AMF queries through SCP configured in the `id` attribute.

> **Note**
>
> `peerconfiguration` and `peersetconfiguration` can be either set to empty list or populated with values. These attributes are used for routing only if `sbiRoutingConfiguration` is enabled for a particular route.

7. After above configurations, configure `enable` in `peermonitoringconfiguration` as `true` to enable peer monitoring. By default, `enable` is set to `false`.

> **Note**
>
> - Peer Monitoring can be enabled to use this feature, where Egress Gateway dynamically monitors the health of the peers configured.
> - It is mandatory to configure `peerconfiguration` with `healthApiPath` if `peermonitoringconfiguration` is enabled.

Here is a sample configuration:

PUT Request

```
curl -v --http2-prior-knowledge -X PUT "http://{{host}}:{{port}}/nssf/nf-
common-component/v1/egw/peermonitoringconfiguration" -H  "Content-Type:
application/json"  --data-raw ' {"enabled": true,"timeout":
1000,"frequency": 2000,"failureThreshold": 3,"successThreshold" : 3}'
```

> **Note**
>
> The IPs and parameter values in the examples are just placeholders. Replace them with your own settings for the cURLs to function correctly.

For detailed information about the REST APIs and parameters, see "Egress Gateway REST APIs" in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

Configure Using CNC Console

Taking into account the prerequisites criteria and recommended sequence explained in sections above, you can refer to the Egress Gateway Configurations section in Configuring NSSF using CNC Console for configuring this feature using CNC Console.

Observe

Metrics

The following metrics are used to provide information about this feature:

- `oc_egressgateway_peer_health_status`
- `oc_egressgateway_peer_health_ping_request_total`
- `oc_egressgateway_peer_health_ping_response_total`
- `oc_egressgateway_peer_health_status_transitions_total`
- `oc_egressgateway_peer_count`
- `oc_egressgateway_peer_available_count`

For information about the metrics, see NSSF Metrics.

Alerts

The following alerts are applicable for this feature:

- OcnssfScpMarkedAsUnavailable
- OcnssfAllScpMarkedAsUnavailable

For more information about the alerts, see NSSF Alerts.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.

2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.16 Support for Kubernetes Resource

4.16.1 Network Policies

Network Policies are an application-centric construct that allows you to specify how a pod is allowed to communicate with various network entities. To control communication between the cluster's pods and services and to determine which pods and services can access one another inside the cluster, it creates pod-level rules.

Previously, NSSF had the privilege to communicate with other namespaces, and pods of one namespace could communicate with others without any restriction. Now, namespace-level isolation is provided for the NSSF pods, and some scope of communications is allowed between the NSSF and pods outside the cluster. The network policies enforces access restrictions for all the applicable data flows except communication from Kubernetes node to pod for invoking container probe.

Managing Support for Network Policies

Enable

To use this feature, network policies need to be applied to the namespace in which NSSF is deployed.

Configure

You can configure this feature using Helm. For information about configuring Network Policy, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

Observe

here is no specific metrics and alerts required for the Network Policy feature.

4.17 Validation of WWW-Authenticate Response Header 4xx with NSSF

When access token validation is enabled, NSSF performs access-token validation of the access token that comes with service requests to it. With this enhancement, NSSF has added supports 3GPP specified 4XX application error codes for these access token checks.

The access token validation include the following checks:

1. **Validating if access token is present in the service request**: If the access token is not present, NSSF returns 401 unauthorized error code together with the "WWW-Authenticate" header as specified in 3GPP 16.5 29.531.
2. **Validating if access token does not have the required scopes to invoke the service operation**: NSSF validates the scope IE in `AccessTokenClaims` (which is the name of the NSSF services for which the access token is authorized) against the NSSF Service that are accessed in this service request. If the validation fails, NSSF returns a 403 Forbidden error code together with the "WWW-Authenticate" header as specified in 3GPP 16.5 29.531.

Managing Validation of WWW-Authenticate Response Header 4xx with NSSF

Enable

This feature does not require any configuration. It is enabled by default when the NSSF is installed.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.18 Deleting Subscription on 404 SUBSCRIPITON_NOT_FOUND Response from AMF

This feature implements a mechanism for the Network Slice Selection Function (NSSF) to manage Access and Mobility Management Function (AMF) subscriptions related to Tracking Area (TA) status changes. When enabled, the NSSF attempts to notify the AMF of any status alterations within a specific TA. If the AMF returns a "404 Subscription Not Found" error, indicating the absence of an active subscription, the NSSF proceeds to delete the associated subscription. This ensures the NSSF maintains an accurate record of active subscriptions.

Advantages:

- **Optimized Notification Flow:** By removing inactive subscriptions, the system prevents the transmission of unnecessary notifications to AMFs that are not subscribed to specific TAs.
- **Resource Efficiency:** Eliminating superfluous notifications reduces network traffic and processing overhead.
- **Reduced Signaling Load:** Prevents unnecessary signaling between the NSSF and AMF.

Use Case Flow:

1. **TA Status Change:** A status change occurs within a designated Tracking Area.
2. **Notification Attempt:** The NSSF attempts to notify the AMF of the status change.
3. **AMF Response:** The AMF responds to the notification attempt.
4. **"404 Subscription Not Found" Response:** If the AMF returns a "404 Subscription Not Found" error, it indicates there is no active subscription for the specific TA.
5. **Subscription Deletion:** Upon receiving the "404" response, the NSSF deletes the subscription associated with the TA.
6. **Notification Cessation:** The AMF no longer receives notifications regarding status changes for the deleted subscription's TA.

> **Note**
>
> This behavior is applicable only when AMF responds with the '404 Subscription Not Found' error. It is not applicable to other failure scenarios.

Managing Deleting Subscription on 404 SUBSCRIPITON_NOT_FOUND Response from AMF

Enable

To enable this feature, set the value of `deleteOnSubscriptionNotFound` parameter to `true` under the `NSSubscription` section in the ocnssf_custom_values_25.2.200.yaml file.

Observe

Metrics

The following metrics are used to provide information about this feature:

- `ocnssf_nssaiavailability_notification_delete_on_subscription_not_found_total`
- `ocnssf_nssaiavailability_notification_db_error`

For information about the Metrics, see NSSF Metrics.

Error Scenarios

The following error logs are generated for this feature:

Table 4-9 Error Scenarios

Scenario	Microservice	Details

Parameter deleteOnSubscriptionNotFound is true but unable to delete NssaiSubscription	NsSubscription	**Request URL**:
		/nnssf-nssubscription/v1/nssai-availability/autoconfignotifications
		/nnssf-nssubscription/v1/nssai-availability/notifications
		Response Code/ Error Title:
		404 SUBSCRIPTION_NOT_FOUND
		Log Snippet:
		<pre>{
		"instant": {
		"epochSecond": 1661327119,
		"nanoOfSecond": 10456886
		},
		"thread": "thread-1",
		"level": "ERROR",
		"loggerName":
		"com.oracle.cgbu.cne.nssf.nsselection.service
		.NsSubscriptionServiceImpl",
		"message": "Failed to delete
		NssaiSubscribtion with ID: 1830762826",
		"endOfBatch": false,
		"loggerFqcn":
		"org.apache.logging.log4j.spi.AbstractLogger"
		,
		"contextMap": {},
		"threadId": 54,
		"threadPriority": 5,
		"ts": "2022-08-24 07:45:19.010+0000",
		"ocLogId": "${ctx:ocLogId}",
		"pod": "ocnssf-nssubscription-5f7bbbffbc-
		hxsfc",
		"processId": "1",
		"vendor": "Oracle",
		"application": "ocnssf",
		"engVersion": "22.3.0",
		"mktgVersion": "22.3.0.0.0",
		"microservice": "nssubscription",
		"namespace": "ocnssf",
		"node_name": "jazz-k8s-node-8"
		}</pre>

Table 4-9 (Cont.) Error Scenarios

Scenario	Microservice	Details

NsSubcription recieves 404 SUBSCRIPTION_NOT_FOUND from client, Param deleteOnSubscriptionNotFound is true hence NssaiSubscription is deleted	NsSubscription	**Request URL**:
		/nnssf-nssubscription/v1/nssai-availability/autoconfignotifications
		/nnssf-nssubscription/v1/nssai-availability/notifications
		Response Code/ Error Title:
		404 SUBSCRIPTION_NOT_FOUND
		Log Snippet:
		
		{
		"instant": {
		"epochSecond": 1679654802,
		"nanoOfSecond": 234222276
		},
		"thread": "task-3",
		"level": "ERROR",
		"loggerName": "com.oracle.cgbu.cne.nssf.nssubscription.service.NsSubscriptionService",
		"message": "Recieved 404 SUBSCRIPTION_NOT_FOUND from http://ocats-amf-stubserver.ocnssf:8080/notification/amf404/, subscriptionId = 402778803, deleteOnSubscriptionNotFound = true, response = NssfRestClientResponse{response=Response{protocol=h2_prior_knowledge, code=404, message=, url=http://ocnssf-egress-gateway:8080/OC_Notify/notification/amf404/}, responseBody='{\"type\": \"NOT_FOUND\", \"title\": \"SUBSCRIPTION_NOT_FOUND\", \"status\": 404, \"detail\": \"subscription not found\", \"cause\": \"SUBSCRIPTION_NOT_FOUND\"}', messageCode=REMOTE_SERVER_EXCEPTION, eventTriggerSubMapId=0, attemptNum=0}",
		"endOfBatch": false,
		"loggerFqcn": "org.apache.logging.log4j.spi.AbstractLogger",
		"contextMap": {},
		"threadId": 760,
		"threadPriority": 5,
		"ts": "2023-03-24 10:46:42.234+0000",
		"ocLogId": "${ctx:ocLogId}",
		"pod": "ocnssf-nssubscription-6d86c4d686-jbxjz",
		"processId": "1",
		"vendor": "Oracle",
		"application": "ocnssf",
		"engVersion": "23.1.1-rc.2",

Table 4-9 (Cont.) Error Scenarios

Scenario	Microservice	Details

		"mktgVersion": "23.1.1-rc.2.0.0",
		"microservice": "nssubscription",
		"namespace": "ocnssf",
		"node_name": "100.77.28.82"
		} {
		"instant": {
		"epochSecond": 1679654802,
		"nanoOfSecond": 408253432
		},
		"thread": "XNIO-1 task-1",
		"level": "INFO",
		"loggerName":
		"com.oracle.cgbu.cne.nssf.nssubscription.helper.HelperFunctions",
		"message": "Successfully deleted Subscription with ID: 402778803",
		"endOfBatch": false,
		"loggerFqcn":
		"org.apache.logging.log4j.spi.AbstractLogger",
		"contextMap": {},
		"threadId": 39,
		"threadPriority": 5,
		"ts": "2023-03-24 10:46:42.408+0000",
		"ocLogId": "${ctx:ocLogId}",
		"pod": "ocnssf-nssubscription-6d86c4d686-jbxjz",
		"processId": "1",
		"vendor": "Oracle",
		"application": "ocnssf",
		"engVersion": "23.1.1-rc.2",
		"mktgVersion": "23.1.1-rc.2.0.0",
		"microservice": "nssubscription",
		"namespace": "ocnssf",
		"node_name": "100.77.28.82"
		}

Table 4-9 (Cont.) Error Scenarios

Scenario	Microservice	Details

NsSubcription recieves 404 SUBSCRIPTION_NOT_FOUND from client, Param deleteOnSubscriptionNotFound is false hence NssaiSubscription is not deleted	NsSubscription	**Request URL**:
		/nnssf-nssubscription/v1/nssai-availability/autoconfignotifications
		/nnssf-nssubscription/v1/nssai-availability/notifications
		Response Code/ Error Title:
		404 SUBSCRIPTION_NOT_FOUND
		Log Snippet:
		
		{
		"instant": {
		"epochSecond": 1679655373,
		"nanoOfSecond": 880206537
		},
		"thread": "task-1",
		"level": "ERROR",
		"loggerName": "com.oracle.cgbu.cne.nssf.nssubscription.service.NsSubscriptionService",
		"message": "Recieved 404 SUBSCRIPTION_NOT_FOUND from http://ocats-amf-stubserver.devnssf-hrithik:8080/notification/amf404/, subscriptionId = 1871925794, deleteOnSubscriptionNotFound = false, response = NssfRestClientResponse{response=Response{protocol=h2_prior_knowledge, code=404, message=, url=http://ocnssf-egress-gateway:8080/OC_Notify/notification/amf404/}, responseBody='{\"type\": \"NOT_FOUND\", \"title\": \"SUBSCRIPTION_NOT_FOUND\", \"status\": 404, \"detail\": \"subscription not found\", \"cause\": \"SUBSCRIPTION_NOT_FOUND\"}', messageCode=REMOTE_SERVER_EXCEPTION, eventTriggerSubMapId=0, attemptNum=0}",
		"endOfBatch": false,
		"loggerFqcn": "org.apache.logging.log4j.spi.AbstractLogger",
		"contextMap": {},
		"threadId": 40,
		"threadPriority": 5,
		"ts": "2023-03-24 10:56:13.880+0000",
		"ocLogId": "${ctx:ocLogId}",
		"pod": "ocnssf-nssubscription-9f68d8bc9-xsm22",
		"processId": "1",
		"vendor": "Oracle",
		"application": "ocnssf",
		"engVersion": "23.1.1-rc.2",

Table 4-9 (Cont.) Error Scenarios

Scenario	Microservice	Details

		"mktgVersion": "23.1.1-rc.2.0.0",
		"microservice": "nssubscription",
		"namespace": "ocnssf",
		"node_name": "100.77.50.225"
		} {
		"instant": {
		"epochSecond": 1679655373,
		"nanoOfSecond": 908894025
		},
		"thread": "XNIO-1 task-1",
		"level": "ERROR",
		"loggerName":
		"com.oracle.cgbu.cne.nssf.nssubscription.service.NsSubscriptionService",
		"message": "Not deleted NssaiSubscribtion with ID: 1871925794",
		"endOfBatch": false,
		"loggerFqcn":
		"org.apache.logging.log4j.spi.AbstractLogger",
		"contextMap": {},
		"threadId": 39,
		"threadPriority": 5,
		"ts": "2023-03-24 10:56:13.908+0000",
		"ocLogId": "${ctx:ocLogId}",
		"pod": "ocnssf-nssubscription-9f68d8bc9-xsm22",
		"processId": "1",
		"vendor": "Oracle",
		"application": "ocnssf",
		"engVersion": "23.1.1-rc.2",
		"mktgVersion": "23.1.1-rc.2.0.0",
		"microservice": "nssubscription",
		"namespace": "ocnssf",
		"node_name": "100.77.50.225"
		}

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.19 DNS SRV Based Selection of SCP in NSSF

NSSF selects Service Communication Proxy (SCP) for indirect communication of notifications using the static configurations done by the operator. This enhancement enables NSSF to learn SCP configuration from DNS SRV based FQDN, in addition to the already existing static manual configuration by the operator.

Egress Gateway (Egress Gateway) supports AlternateRoute Service, which NSSF is using to support DNS SRV based selection of SCP. It enables NSSF to resolve FQDN or Virtual FQDN to alternate FQDNs of SCP. Egress Gateway uses the virtual FQDN of SCP instances to query the AlternateRoute Service and get the list of alternate FQDNs with priorities assigned to each of them. Based on the priorities, Egress Gateway picks up the SCP instances for rerouting attempts.

The AlternateRoute Service allows the configuration of multiple sets of SCP instances in NSSF in contrast to only one static configuration in the previous scenario.

Managing DNS SRV Based Selection of SCP in NSSF

Configure

Configure Using Helm Parameters:

DNS SRV is enabled by default at the time of installation. The `dnsSrvEnabled` parameter is set to `true` by default in the `ocnssf_custom_values_25.2.200.yaml` file:

- `dnsSrvEnabled: true`

> **Note**
>
> The default value of this parameter is set to `true` by default. It is recommended to keep this value as `true` only. Disabling it may cause issues with the functioning of this feature and other features that are depended on it.

- `dnsSrvFqdnSetting.enabled: true`

> **Note**
>
> Flag to enable or disable the usage of custom patterns for the FQDN while triggering DNS-SRV query. It is set to `true` by default.

- `dnsSrvFqdnSetting.pattern: "_{scheme}._tcp.{fqdn}."`

For more information about Helm parameters to configure DNS SRV and Alternate Routing Service, see "Alternate Route Microservice Parameters section" in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

Configure Using REST API:

The feature related REST API configurations are performed at NSSF and Egress Gateway.

1. Perform REST API configurations at Egress Gateway in the following sequence:

a. Configure `peerconfiguration` to define the list of peers to which Egress Gateway can send request.

> **Note**
>
> It is mandatory to configure `peerconfiguration` with `healthApiPath` if you want to enable `peermonitoringconfiguration`.

b. Configure `virtualHost` under `peerconfiguration` where the AMF query is sent.

c. Configure `peersetconfiguration` to logically group the peers into sets.

> **Note**
>
> - `peerIdentifier` must be the value of SCP peer configured in `peerConfiguration`.
> - You cannot configure multiple virtual hosts as peers in the same peer set.
> - Configure the `priority` for each SCP peer in the set. Depending on the priority, it selects the primary, secondary, or tertiary SCP peers to route requests.

d. Configure or update `errorcriteriasets`.

e. Configure or update `erroractionsets`.

f. Configure `routesconfiguration` to define the route and reroute parameters. If SBIRouting functionality is required, then configure `SBIRoutingFilter`. If reroute mechanism is required for that route, then configure `SBIReroute` filter with retries, methods, and statuses.

> **Note**
>
> `peerSetIdentifier` must be the value configured during `peersetconfiguration`.

g. Set the value of `enabled` under `sbiRoutingConfiguration` to `true` to route the AMF queries through SCP configured in the `id` attribute.

> **Note**
>
> `peerconfiguration` and `peersetconfiguration` can be either set to empty list or populated with values. These attributes are used for routing only if `sbiRoutingConfiguration` is enabled for a particular route.

h. <Optional> You can also configure `peermonitoringconfiguration` using REST API or CNC Console. For more information about enabling or configuring `peermonitoringconfiguration`, see Monitoring the Availability of SCPs using SCP Health APIs.

> **Note**
>
> It is mandatory to configure `peerconfiguration` with `healthApiPath` if `peermonitoringconfiguration` is enabled.

2. Perform the following REST API configurations at NSSF:
   a. Configure the `nssaiauth` Managed Object to enable the configuration of network slice authentication rules by configuring Grant status (Allowed_PLMN, Rejected_PLMN, or Rejected_TAC) for S-NSSAI on a per TAI basis.

For more information about REST API parameters and configuration, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

Configure using CNC Console

Taking into account the prerequisites criteria and recommended sequence explained in sections above, you can see Configuring NSSF using CNC Console section for configuring this feature using CNC Console.

Observe

Metrics

No new Metrics or KPIs were added to NSSF. However, the following Egress Gateway metrics for Alternate Route Service are used to provide the information about this feature:

- `oc_fqdn_alternate_route_total`
- `oc_dns_srv_lookup_total`
- `oc_alternate_route_resultset`
- `oc_configclient_request_total`
- `oc_configclient_response_total`

For information about the Metrics, see Egress Gateway Metrics in NSSF Metrics.

Error Scenarios

No new logs are generated for this feature. However, it uses the following Egress Gateway error scenarios:

Table 4-10 Error Scenarios

Scenario	Microservice	Details

Sending Subscription notification failed due to UnknownHost exception	ocnssf-egress-gateway	**Request URL**:
		/nnssf-configuration/v1/nssaiauth
		Response Code/ Error Title:
		503
		Service Unavailable
		Encountered unknown host exception at Egress Gateway
		Log Snippet:
		
		{
		"instant": {
		"epochSecond": 1676964069,
		"nanoOfSecond": 986866249
		},
		"thread": "@6c8fe7a4-217",
		"level": "ERROR",
		"loggerName":
		"ocpm.cne.gateway.jettyclient.DnsResolver",
		"message": "UnExpected error occured ::
		{}",
		"thrown": {
		"commonElementCount": 0,
		"localizedMessage": "ocats-amf-
		stubserver.ocnssf: Name or service not
		known",
		"message": "ocats-amf-
		stubserver.ocnssf:Name or service not known",
		"name": "java.net.UnknownHostException",
		"extendedStackTrace": [
		{
		"class": "java.net.Inet6AddressImpl",
		"method": "lookupAllHostAddr",
		"file": "Inet6AddressImpl.java",
		"line": -2,
		"exact": false,
		"location": "?",
		"version": "?"
		},
		{
		"class":
		"java.net.InetAddress$PlatformNameService",
		"method": "lookupAllHostAddr",
		"file": "InetAddress.java",
		"lin e": 933,
		"exact": false,
		"location": "?",
		"version": "?"
		}
		]
		},

Table 4-10 (Cont.) Error Scenarios

Scenario	Microservice	Details

		<pre>"endOfBatch": false,
		"loggerFqcn":
		"org.apache.logging.log4j.spi.AbstractLogger"
		,
		"contextMap": {},
		"threadId": 217,
		"thread Priority": 5,
		"messageTimestamp":
		"2023-02-21T07:21:09.986+0000",
		"ocLogId": "${ctx:ocLogId}",
		"pod": "${ctx:hostname}",
		"processId": "1",
		"instanceType": "prod",
		"egressTxId": "${ctx:egressTxId}"
		}</pre>

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.20 OAuth Access Token Based Authorization

NSSF supports Oauth 2.0, which is a security feature that NSSF uses to validate and authorize requests from allowed or valid consumers NFs. The consumer NF requests for access token from the issuer NRF, and uses this access token to send the request to NSSF. NSSF validates the requests and approves or discards it based on access token authorization received in the request. The access token is validated with the configured public key certificate in NSSF.

Before this enhancement, NSSF used NRF Instance ID to validate the access token, where Ingress Gateway stored public keys against NRF instance Id. This enhancement allows NSSF to use multiple public certificates for validating access tokens by adding support for Key-ID (K-ID) based access token validation, in addition to the existing NRF Instance ID based access token validation.

This enhancement now allows Ingress Gateway to operate in the following three different modes:

1. K-ID based ONLY
   a. Ingress Gateway validates access token based on public keys indexed with key-id only.
2. Instance ID based ONLY ***(DEFAULT)***

a. Ingress Gateway validates access token based on public keys indexed with NRF Instance ID in the issuer field.

3. K-ID based with Instance ID based as fallback (KID_PREFERRED)
   a. Ingress Gateway validates access token based on public keys indexed with Key-ID. If Key-ID is not FOUND in Access token, Ingress Gateway attempts token validation using public keys indexed with NRF instance ID in the issuer field.
   b. Fallback happens only if the received access token is structured as follows:
      i. Does not contain Key-ID
      ii. Contains Key-ID but does not have public keys configured against the Key-ID

Managing OAuth Access Token Based Authorization Using Key-ID and NRF Instance ID

Prerequisites

This section describes the configurations required to enable access tokens before deploying NSSF.

Generating KeyPairs for NRF Instances

> **Note**
>
> It is at the discretion of the user to create private keys and certificates, and it is not in the scope of NSSF. This section lists only samples to create KeyPairs.

Using the OpenSSL tool, the user can generate private key and public certificates. The commands to generate the KeyPairs are as follows:

Example Command to generate KeyPair for NRF Instance

```
openssl ecparam -genkey -name prime256v1 -noout -out ec_private_key1.pem

openssl pkcs8 -topk8 -in ec_private_key1.pem -inform pem -out
ec_private_key_pkcs8.pem -outform pem -nocrypt

openssl req -new -key ec_private_key_pkcs8.pem -x509 -nodes -days 365 -out
4bc0c762-0212-416a-bd94-b7f1fb348bd4.crt -subj "/C=IN/ST=KA/L=BLR/O=ORACLE/
OU=CGBU/CN=ocnrf-endpoint.ocnrf.svc.cluster.local"
```

> **Note**
>
> For ATS configuration details, see **Configuring Secrets to Enable Access Token** in **Preinstallation Tasks** of *Cloud Native Core Network Slice Selection Function Installation and Upgrade* Guide.

Enabling and Configuring Access Token

To enable access token validation, configure both Helm-based and REST-based configurations on Ingress Gateway.

> **Note**
>
> While Helm based configuration is mandatory, you can also perform CNC Console-based configuration instead of REST-based configurations.

Configuration using Helm:

For Helm-based configuration, perform the following steps:

1. Create a secret that stores NRF public key certificates using the following commands:

   ```
   kubectl create secret generic <secret name> --from-file=<filename.crt> -n
   <Namespace>
   ```

   For Example:

   ```
   kubectl create secret generic oauthsecret --from-file=4bc0c762-0212-416a-
   bd94-b7f1fb348bd4.crt -n ocnssf
   ```

   > **Note**
   >
   > In the above command:
   >
   > - `oauthsecret` is the secret name
   > - `ocnssf` is the namespace
   > - `4bc0c762-0212-416a-bd94-b7f1fb348bd4.crt` is the public key certificate

2. Enable the `oauthValidatorEnabled` parameter on Ingress Gateway by setting its value to `true`. Further, configure the secret and namespace on Ingress Gateway in the OAUTH CONFIGURATION section of the `ocnssf_custom_values_25.2.200.yaml` file using the following fields:
   - `oauthValidatorEnabled`
   - `nfType`
   - `nfInstanceId`
   - `producerScope`
   - `allowedClockSkewSeconds`
   - `enableInstanceIdConfigHook`
   - `nrfPublicKeyKubeSecret`
   - `nrfPublicKeyKubeNamespace`
   - `validationType`
   - `producerPlmnMNC`
   - `producerPlmnMCC`
   - `oauthErrorConfigForValidationFailure`
   - `oauthErrorConfigForValidationFailure.errorCode`

- `oauthErrorConfigForValidationFailure.errorTitle`
- `oauthErrorConfigForValidationFailure.errorDescription`
- `oauthErrorConfigForValidationFailure.errorCause`
- `oauthErrorConfigForValidationFailure.redirectUrl`
- `oauthErrorConfigForValidationFailure.retryAfter`
- `oauthErrorConfigForValidationFailure.errorTrigger`
- `oauthErrorConfigForValidationFailure.errorTrigger.exceptionType`

> **Note**
>
> `4bc0c762-0212-416a-bd94-b7f1fb348bd4.crt` is the public key certificate and we can have any number of certificates in the secret.

The following snippet represents the location of the mentioned parameter in the Helm file:

> **Note**
>
> - The following snippet represents only the sample values.
> - For more information on parameters and their supported values, see **Ingress Gateway Parameters** from **Customizing NSSF** chapter in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*
> - For information about OAuth access token attributes like kid, typ, iss, aud, scope etc., see *https://www.rfc-editor.org/rfc/rfc7515.html* page.

```
#OAUTH CONFIGURATION

  oauthValidatorEnabled: true

  nfType: NSSF

  nfInstanceId: 9faf1bbc-6e4a-4454-a507-aef01a101a01

  producerScope: nnssf-nsselection,nnssf-nssaiavailability

  allowedClockSkewSeconds: 0

  enableInstanceIdConfigHook: true

  nrfPublicKeyKubeSecret: oauthsecret

  nrfPublicKeyKubeNamespace: ocnssf

  validationType: strict

  producerPlmnMNC: 14
```

```
  producerPlmnMCC: 310

  oauthErrorConfigForValidationFailure:

    errorCode: 401

    errorTitle: "Validation failure"

    errorDescription: "UNAUTHORIZED"

    errorCause: "oAuth access Token validation failed"

    redirectUrl:

    retryAfter:

    errorTrigger:

    - exceptionType: OAUTH_CERT_EXPIRED

      errorCode: 408

      errorCause: certificate has expired

      errorTitle:

      errorDescription:

      retryAfter:

      redirectUrl:- exceptionType: OAUTH_MISMATCH_IN_KID

      errorCode: 407

      errorCause: kid configured does not match with the one present in
the token

      errorTitle:

      errorDescription:

      retryAfter:

      redirectUrl:

    - exceptionType: OAUTH_PRODUCER_SCOPE_NOT_PRESENT

      errorCode: 406

      errorCause: producer scope is not present in token

      errorTitle:

      errorDescription:

      retryAfter:
```

```
      redirectUrl:

    - exceptionType: OAUTH_PRODUCER_SCOPE_MISMATCH

      errorCode: 405

      errorCause: produce scope in token does not match with the
configuration

      errorTitle:

      errorDescription:

      retryAfter:

      redirectUrl:

    - exceptionType: OAUTH_MISMATCH_IN_NRF_INSTANCEID

      errorCode: 404

      errorCause: nrf id configured does not match with the one present in
the token

      errorTitle:

      errorDescription:

      retryAfter:

      redirectUrl: - exceptionType: OAUTH_PRODUCER_PLMNID_MISMATCH

      errorCode: 403

      errorCause: producer plmn id in token does not match with the
configuration

      errorTitle:

      errorDescription:

      retryAfter:

      redirectUrl:

    - exceptionType: OAUTH_AUDIENCE_NOT_PRESENT_OR_INVALID

      errorCode: 402

      errorCause: audience in token does not match with the configuration

      errorTitle:

      errorDescription:
```

```
      retryAfter:

      redirectUrl:

    - exceptionType: OAUTH_TOKEN_INVALID

      errorCode: 401

      errorCause: oauth token is corrupted

      errorTitle:

      errorDescription:

      retryAfter:

redirectUrl:oauthErrorConfigOnTokenAbsence:

   errorCode: 400

   errorTitle: "Token not present"

   errorDescription: "UNAUTHORIZED"

   errorCause: "oAuth access Token is not present"

   redirectUrl:

   retryAfter:
```

Configuration using REST API or CNC Console

REST API

After Helm configuration, send the REST requests to use configured public key certificates. Using REST-based configuration, you can distinguish between the certificates configured on different NRFs and can use these certificates to validate the token received from a specific NRF.

For more information about REST API configuration, see **OAuth Validator Configuration** section in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide.*

CNC Console

For more information on CNC Console based configuration, see OAuth Validator Configurations.

Observe

- Added the following success measurements:
  - `oc_oauth_nrf_request_total`
  - `oc_oauth_nrf_response_success_total`
  - `oc_oauth_token_cache_total`
  - `oc_oauth_validation_successful_total`

  - `oc_oauth_cert_expiryStatus`
  - `oc_oauth_cert_loadStatus`
  - `oc.oauth.keyid.count`
- Added the following error measurements:
  - `oc_oauth_nrf_response_failure_total`
  - `oc_oauth_request_failed_internal_total`
  - `oc_oauth_request_invalid_total`
  - `oc_oauth_validation_failure_total`
  - `oc.oauth.request.failed.cert.expiry`

For information on Metrics and KPIs of NSSF, see NSSF Metrics and NSSF KPIs sections respectively.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.21 Overload Control based on Percentage Discards

The Overload Control feature protects the system from overload and maintains the overall health of NSSF. The system needs to not only detect overload conditions but also protect against the same. Further, it needs to mitigate against and avoid the system from entering into an overload condition by taking necessary actions for recovering from overload.

NSSF provides the following means for overload management:

- Predefined threshold load levels
- Tracks number of pending messages
- Tracks CPU and memory usage
- Enforce load shedding during various overload levels

Perf-info performs overload calculations based on the indicators:

- CPU Utilization
- Memory Utilization
- Pending Message Count
- Failure Count

The overload level is configured for the following NSSF microservices:

- NSSelection
- NSAvailability

The Overload Manager module in Perf-info is configured or updated with the threshold value for services. A configurable flag is available for sampling interval as

`ocPolicyMapping.samplingPeriod` based on which Ingress Gateway calculates rate per service in the current sampling period and applies appropriate discard policies and actions in the subsequent sampling period.

Overload Manager triggers Rate Calculator to start calculating the rate of incoming requests per service per sampling period. Ingress Gateway receives a notification event per service with the calculated rates to the Overload Manager filter at the end of every sampling period. It applies an appropriate configured discard policy for a particular service based on the rate of requests.

Ingress Gateway calculates the number of requests to be dropped in the current sampling period based on configured percentage discard.

Overload Thresholds for each service is evaluated based on four metrics namely `cpu`, `svc_failure_count`, `svc_pending_count`, and `memory`. Overload control is triggered if the thresholds for any one metrics are reached.

> **Note**
>
> When the percentage-based overload control discarding policy is enabled, the number of requests to be dropped in the current sampling period is computed based on the configured percentage discard and the "rate of requests outgoing of Ingress Gateway" in the previous sampling period for the service.
>
> Once the number of requests to be dropped in the current sampling period is computed, the gateway does not drop all the new traffic to meet the discard count. Instead, Ingress Gateway executes a random function to decide if a request is to be discarded or not. If the random function returns true, the request is discarded in the current sampling period with the discard action "RejectWithErrorCode". This ensures there is a spread of discard requests in a sampling period.
>
> Since we are calculating the number of requests to be dropped in the current sampling period based on the number of requests sent to the backend service in the previous sampling period and not on the total requests received at Ingress Gateway, the percentage dropped is not exactly the percentage configured.

Managing Overload Control based on Percentage Discards

Enable Overload Control Feature

You can enable Overload Control feature using the following Helm configuration:

1. Open the `ocnssf_custom_values_25.2.200.yaml` file.
2. Set the `global.performanceServiceEnable` parameter to `true` in the `ocnssf_custom_values_25.2.200.yaml` file.
   The following snippet represents the location of the mentioned parameter in the Helm file:

   ```
   #Flag to Enable or Disable Performance Service. The flag is set to true to
   enable the overload control feature by default.
     performanceServiceEnable: true
   ```

3. Set the `perf-info.overloadManager.enabled` parameter to `true` in the `ocnssf_custom_values_25.2.200.yaml` file.

The following snippet represents the location of the mentioned parameter in the Helm file:

```
overloadManager:
   ingressGatewayPort: *httpSignalPort
   #Flag to Enable or Disable overloadManager
   enabled: true
```

4. Configure the Prometheus URI in `perf-info.configmapPerformance.prometheus`
   The following snippet represents the location of the mentioned parameter in the Helm file:

```
perf-info
  configmapPerformance:
    prometheus: http://occne-prometheus-server.occne-infra:80
```

> **Note**
>
> Update the URL as per your setup. It should be a valid Prometheus server URL, which is same as data source URL used on the Grafana dashboard.

5. Save the `ocnssf_custom_values_25.2.200.yaml` file.
6. Run `helm upgrade`, if you are enabling this feature after NSSF deployment. For more information on upgrade procedure, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation and Upgrade Guide*.

Configure

You can configure this feature using Helm, REST API, and CNC Console.

Configure using REST API:

The Overload Control feature related configurations are performed at Ingress Gateway and Perf-info.

The following REST APIs must be configured for this feature in the following order:

1. {apiRoot}/nssf/nf-common-component/v1/igw/errorcodeprofiles

> **Note**
>
> **Dependency**: `errorcodeprofiles` is used in `ocdiscardpolicies` to define how different overload levels trigger rejection with specific errors.

2. {apiRoot}/nssf/nf-common-component/v1/igw/ocdiscardpolicies

> **Note**
>
> **Dependency**:
>
> - `ocdiscardpolicies` use `errorcodeprofiles` to decide the error message when rejecting requests.
> - `ocdiscardpolicies` are used by `ocpolicymapping` to associate services with specific overload handling policies.

3. {apiRoot}/nssf/nf-common-component/v1/igw/ocpolicymapping

> **Note**
>
> **Dependency**:
>
> - Depends on `ocdiscardpolicies` to apply the right rejection policy to each service.

4. {apiRoot}/nssf/nf-common-component/v1/igw/errorcodeserieslist

> **Note**
>
> Not directly linked to other configurations but enhances error handling by classifying errors.

5. {apiRoot}/nssf/nf-common-component/v1/igw/routesconfiguration

> **Note**
>
> `routesconfiguration` defines routing behaviors and associates services with error code series. It references `errorCodeSeriesId` which is defined in `id` attribute in `errorcodeserieslist`.

6. {apiRoot}/nssf/nf-common-component/v1/perfinfo/overloadLevelThreshold

> **Note**
>
> Determines when a service is considered overloaded. It triggers `ocdiscardpolicies` when thresholds are exceeded, causing requests to be rejected with predefined error responses.

For more information about APIs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*

Configure using CNC Console:

The above REST APIs can also be configured using CNC Console UI. For more information, see the follwing sections in the Configuring NSSF using CNC Console chapter:

- Create or update the Error Code Profiles.
- Create or update the Create Overload Control Discard Policies .
- Create or update the Discard Policy Mapping.
- Create or update the Error Code Series.
- Update the Routes Configuration.

Disable Overload Control Feature

You can disable this feature using the following REST and Helm configurations:

1. In the following REST API change the value of `enable` parameter to `false`:
   {apiRoot}/nssf/nf-common-component/v1/igw/ocpolicymapping

For more information about the `ocpolicymapping` REST API, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

2. Open the `ocnssf_custom_values_25.2.200.yaml` file.
3. Set the `perf-info.overloadManager.enabled` parameter to `false` in the `ocnssf_custom_values_25.2.200.yaml` file.
   The following snippet represents the location of the mentioned parameter in the Helm file:

```
overloadManager:
   ingressGatewayPort: *httpSignalPort
   #Flag to Enable or Disable overloadManager
   enabled: false
```

4. Save the `ocnssf_custom_values_25.2.200.yaml` file.
5. Run `helm upgrade`, if you are enabling this feature after NSSF deployment. For more information on upgrade procedure, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation and Upgrade Guide*.

> **Note**
>
> If you want to enable the feature again after disabling it, follow the steps mentioned in the Enable Overload Control Feature and Configure sections.

Observe

Metrics

No new metrics added to NSSF for the Overload Control feature. However, the following Perf-info metrics are used to provide the information about overload control feature:

- `cgroup_cpu_nanoseconds`
- `cgroup_memory_bytes`
- `load_level`

For information about Metrics, see Perf-info metrics for Overload Control in NSSF Metrics.

For information on KPIs of NSSF, see NSSF KPIs section.

Alerts

The following alerts are added for the Overload Control feature:

- OcnssfOverloadThresholdBreachedL1
- OcnssfOverloadThresholdBreachedL2
- OcnssfOverloadThresholdBreachedL3
- OcnssfOverloadThresholdBreachedL4

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.

2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.22 Auto-Population of Configuration Based on NSAvailability Update

This feature simplifies the configuration of authorized S-NSSAIs per TAI in the NSSF by dynamically deriving them from the NssaiAvailabilityData received from the AMF

When the Feature is Enabled

- No need to configure SupportedSNSSAIs per TAI manually.
- Operators must configure:
  - ConfiguredSNSSAIs (allowed S-NSSAIs)
  - BarredSNSSAIs (optional) at the PLMN portfolio level.

NSSF considers an S-NSSAI authorized in a TAI if:

- It is present in the NssaiAvailabilityData from AMF (i.e., AMF supports it in that TAI).
- It is part of the ConfiguredSNSSAIs at the PLMN level.
- It is not barred in that TAI or PLMN.

Additional Rule

- NSSF treats all configured S-NSSAIs as applicable to all TAIs unless explicitly restricted.

Operational Benefit

- No per-TAI support configuration is required, which simplifies setup and reduces operational errors.

When the Feature is Disabled

- Operators must manually configure:
  - ConfiguredSNSSAIs at PLMN level
  - SupportedSNSSAIs per TAI using the Supported in TAI managed object
  - Optional BarredSNSSAIs per TAI

NSSF considers an S-NSSAI authorized in a TAI only if:

- It is present in the NssaiAvailabilityData from AMF for that TAI.
- It is explicitly listed as a SupportedSNSSAI for that TAI in the configuration.
- It is not barred.

Operational Overhead

- Manual per-TAI mapping is required, which increases complexity and maintenance effort.

Walkthrough – Step-by-Step Flow (Feature Enabled)

1. Operator configures ConfiguredSNSSAIs at the PLMN level.
2. AMF sends NssaiAvailabilityData for each TAI.
3. NSSF matches the configured S-NSSAIs with the data received from AMF.

4. Any S-NSSAI found in both the ConfiguredSNSSAIs and AMF data, and not barred, is authorized for the TAI. No TAI-level SupportedSNSSAIs configuration is needed.

Examples

Example 1 – Feature Enabled

- ConfiguredSNSSAIs at PLMN: S1, S2
- NssaiAvailabilityData from AMF:
  - TAI-1: S1
  - TAI-2: S2

Result:

- TAI-1: Authorized S-NSSAIs = S1
- TAI-2: Authorized S-NSSAIs = S2
- No per-TAI support configuration is needed.

Walkthrough – Step-by-Step Flow (Feature Disabled)

1. The operator configures ConfiguredSNSSAIs at the PLMN level.
2. The AMF sends NssaiAvailabilityData for each TAI.
3. The NSSF matches the configured S-NSSAIs with the data received from the AMF.
4. Any S-NSSAI found in both ConfiguredSNSSAIs and AMF data, and explicitly listed as a SupportedSNSSAI for that TAI, is authorized for the TAI. TAI-level SupportedSNSSAIs configuration is required.

Example 2 – Feature Disabled

- ConfiguredSNSSAIs at PLMN: S1, S2
- Operator explicitly configures:
  - Supported in TAI-1: S1
  - Supported in TAI-2: S2
- Even if AMF sends both S1 and S2 for TAI-1 and TAI-2, only what is manually listed for each TAI will be authorized.

Key Rules (Applicable in Both Modes)

- An S-NSSAI is authorized only if it is both configured in NSSF and reported as supported by AMF.
- PLMN-level configuration is mandatory. TAI-level configuration depends on the feature state.
- Any S-NSSAI listed in BarredSNSSAIs (PLMN or TAI) will not be authorized.

How It Works

AMF Sends Availability Update

- AMF sends NssaiAvailabilityData indicating the S-NSSAIs it supports in each TAI.

NSSF Processing Logic

If Feature is Enabled:

- Checks whether the S-NSSAI is configured at the PLMN level.
- Verifies that the S-NSSAI is not barred (PLMN or TAI level).

- Stores availability data in the internal database.
- Creates or updates internal authorization records and notifies subscribed AMFs about new authorizations.

If Feature is Disabled:

- Verifies that the `S-NSSAI` is explicitly allowed in the configured `SupportedSNSSAIs` for the `TAI`.
- Checks barring rules.
- Authorizes only those `S-NSSAIs` that meet all criteria.

Sample Call Flow

Scenario Setup

- Configured at `PLMN`: `S1`, `S2`, `S3`, `S4`
- Barred at `PLMN`: `S4`
- Barred in `TAI-1`: `S3`
- Supported in `TAI-1`: `S1` (only if feature disabled)
- AMF reports `S1`, `S2`, `S3`, `S4` as supported in `TAI-1`

Call Flow (Feature Disabled)

1. AMF sends `NSAvailability Update` to NSSF.
2. NSSF checks `ProvisionDB` and sees that only `S1` is allowed in `TAI-1`.
3. NSSF stores the data in StateDB.
4. NSSF authorizes `S1` and responds to AMF: `"S1 is authorized in TAI-1"`.

Call Flow (Feature Enabled)

1. AMF sends `NSAvailability Update` to NSSF.
2. NSSF:
   - Finds `S1–S4` configured at `PLMN`.
   - Excludes `S4` (barred in `PLMN`) and `S3` (barred in `TAI-1`).
   - NSSF stores the data in StateDB.
   - Sends notification to other AMFs subscribed to `TAI-1`.
3. NSSF responds to AMF: `"S1 and S2 are authorized in TAI-1"`.

Key Differences

Table 4-11 Key Differences

Aspect	Feature Enabled	Feature Disabled
TAI-level support config	Not needed	Required
PLMN-level config required	Yes	Yes
Barring respected	Yes	Yes
Dynamic updates	Yes	No
Operational complexity	Low	High

Benefits

- Reduces manual configuration effort
- Minimizes configuration errors
- Automatically keeps NSSF in sync with AMF capabilities
- Supports efficient, large-scale deployments with minimal overhead

Managing Auto-Population of Configuration Based on NSAvailability Update

Enable

Steps to Enable

You can enable this feature using REST API or CNC Console.

Configure

Helm

There are no Helm configurations required for this feature.

REST API

A boolean parameter, `autoAuthorizeNssaiAvailabilityDataEnable`, is added to enable or disable this feature using the `SystemOptions` API. Update this parameter value to true or false to enable or disable this feature, respectively. By default, it is set to `true`.

> **Note**
>
> - For more information, see **SystemOptions** in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide.*

CNC Console

The value of the `AutoAuthorizeNssaiAvailabilityDataEnable` parameter can also be updated using the CNC Console interface for the NSSF System Option.

Prerequisites

Operator must configure the following for this feature to work:

1. **Configure PLMN Level NSI Profile**: Configure **PLMN Level NSI Profile** for each supported PLMN, as `nssai_auth` autoconfiguration happens only when default profile is configured for the PLMN. For more information on REST based configuration, see **PLMN Level NSI Profile** in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide.* For more information on CNC Console based configuration, see PLMN Level NSI Profile in 'Configuring NSSF using CNC Console'.

> **⚠ Caution**
>
> - **GR Deployment**: In a GR deployment, the operator must configure the same set of AMFs on all sites.
> - **Security:** Trusted AMFs can update NSSF configuration, so operators must carefully manage which AMFs are given this capability.
> - **Site-Level Configuration:** Trusted AMFs and system options are site-level configurations. Operators must ensure consistent configuration across all sites.

Observe

No new metrics or KPIs are generated for this feature. For information on other Metrics and KPIs of OCNSSF, see OCNSSF Metrics and OCNSSF KPIs sections respectively.

Error Scenarios

For Auto-Population of Configuration Based on NSAvailability Update, logs are generated for NSAvailability, when error is due to configuration in the PLMN Level NSI Profile.

Table 4-12 Error Scenarios

Scenario	Microservice	Details
PLMN Level NSI Profile is not configured	Nnssf_NSSAIAvailability	**Response Code/ Error Title**: Configuration issue: PLMN Level Profile is not configured for <MCC> <MNC> Unable to process nsavailability request 500 Response with details missing configuration. Unable to find PLMN level profile for <MCC> <MNC> **Log Snippet:** `{ "instant": { "epochSecond": 1661325081, "nanoOfSecond": 495990110 }, "thread": "XNIO-1 task-1", "level": "ERROR", "loggerName": "com.oracle.cgbu.cne.nssf.nsavailability.dataservicehelper.AmfTaiSnssaiMapDataPopulation", "message": "CONFIGURATION_ERROR: Unable to find Plmn Level Profile", "endOfBatch": false, "loggerFqcn": "org.apache.logging.log4j.spi.AbstractLogger", "contextMap": { "ocLogId": "1661325081404_2998_ocnssf-ingress-gateway-fd65885d6-lppss" }, "threadId": 234, "threadPriority": 5, "ts": "2022-08-24 07:11:21.495+0000", "ocLogId": "1661325081404_2998_ocnssf-ingress-gateway-fd65885d6-lppss", "pod": "ocnssf-nsavailability-65466b5f48-xtvgz", "processId": "1", "vendor": "Oracle", "application": "ocnssf", "engVersion": "22.2.0", "mktgVersion": "22.2.0.0.0", "microservice": "nsavailability", "namespace": "ocnssf", "node_name": "k8s-node-8.bulkhead.lab.us.oracle.com" }`

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.23 Feature Negotiation

This feature negotiates optional features applicable between NSSF and NF Service Consumer (AMF/V-NSSF) for the NSSF supported services. The NF Service Consumer indicates the optional features it supports for the Nnssf_NSSAIAvailability or Nnssf_NSSElection service by including the supported feature attributes.

The following optional supported features are defined for NSSF as per 3GPP:

1. **Nnssf_NSSAIAvailability service supportedFeatures attributes:**
   - **Subscription Modification (SUBMOD)**: This feature allows the operator to modify subscriptions by supporting HTTP Patch on NSAvailability (/nssai-availability/subscriptions/). When Subscription Modification in Subscribe Service Operation (SUMOD) is supported, the operator can modify the subscription by implementing the HTTP Patch method.
   - **Empty Authorized NSSAI Availability Notification (EANAN)**: When this feature is supported, an NF Consumer that supports EANAN accepts an empty array of Authorized NSSAI Availability Data in a notification from NSSF and deletes locally stored Authorized NSSAI Availability Data that was received previously.

There is a negotiation between the NSSF and the requestor NSSF for supported features so that they can be enabled. This is done using the `SupportedFeature` IE.

Supported Feature Information Element (IE): Supported Feature is a hexadecimal string that contains a bitmask indicating supported features. Each character in the string can take a value of "0" to "9", "a" to "f" or "A" to "F". The character representing the highest-numbered features appears first in the string, and the character representing features 1 to 4 appears last in the string. The list of features and their numbering (starting with 1) are defined separately for each API. If the string contains a lower number of characters, then there are defined features for an API.

> **Note**
>
> Features represented by the characters that are not present in the string are not supported.

Behavior Based on `SupportedFeatures` in `NnssaiAvailability` (Update and Subscription)

FeatureNegotiationEnable Flag: true

Validity Rules

- `SupportedFeatures` > 1-byte integer not in (0 to F hexadecimal)
  NSSF responds with 400 Bad Request – Optional parameter invalid.

- `SupportedFeatures` in (0 to F hexadecimal) includes features not enabled (via Helm flags) NSSF responds with 400 Bad Request – Unsupported Feature.

Table 4-13 Accepted `SupportedFeatures` values

SUMOD (Helm)	EANAN (Helm)	Accepted Values for `SupportedFeatures`
true	true	0, 2, 4, 6
true	false	0, 2
false	true	0, 4
false	false	0

FeatureNegotiationEnable Flag: false

- Accepted `SupportedFeatures` value: 0
- `SupportedFeatures` > 0
  NSSF responds with 400 Bad Request – Optional parameter invalid with the message "Feature Negotiation is not enabled from NSSF side".

> **Note**
>
> Features represented by characters that are not present in the string are not supported. ONSSAI and ES3XX are not supported by the NSSF and thus cannot be enabled at the NSSF.

Table 4-14 SupportedFeatures for NSAvailability

Supported Feature based on supported feature set	ES3XX	EANAN	SUMOD	ONSSAI	Applicable to NSSF
"0"	no	no	no	no	Yes
"1"	no	no	no	yes	No
"2"	no	no	yes	no	Yes
"3"	no	no	yes	yes	No

Table 4-14 (Cont.) SupportedFeatures for NSAvailability

Supported Feature based on supported feature set	ES3XX	EANAN	SUMOD	ONSSAI	Applicable to NSSSF
"4"	no	yes	no	no	Yes
"5"	no	yes	no	yes	No
"6"	no	yes	yes	no	Yes
"7"	no	yes	yes	yes	No
"8"	yes	no	no	no	No
"9"	yes	no	no	yes	No
"A"	yes	no	yes	no	No
"B"	yes	no	yes	yes	No
"C"	yes	yes	no	no	No
"D"	yes	yes	no	yes	No
"E"	yes	yes	yes	no	No
"F"	yes	yes	yes	yes	No

Managing Feature Negotiation

Enable

To enable this feature, set the `global.SupportedFeatureNegotiationEnable` parameter to `true` under the `global` section in the `ocnssf_custom_values_25.2.200.yaml` file.

The following snippet represents the location of the mentioned parameter in the Helm file:

```
global:
 SupportedFeatureNegotiationEnable: true
 threegppFeatures:
    NsAvalability:
      SUMOD: true
      EANAN: true
```

Configure

There are no additional configurations required.

Observe

No new metrics or KPIs are generated for this feature. For information on other Metrics and KPIs of OCNSSF, see OCNSSF Metrics and OCNSSF KPIs sections respectively.

Error Scenarios

Table 4-15 Error Scenarios

Scenario	Helm Configuration	Output

NSAvailability request with supported feature. That is,
'2'
	`SupportedFeatureNegotiationEnable: true`
	`threegppFeatures:`
	`NsAvalability:`
	`SUMOD: true`
	`EANAN: true`
		Response with supported feature i.e.
		'2'

NSAvailability request with supported feature. That is,
'3'
	`SupportedFeatureNegotiationEnable: true`
	`threegppFeatures:`
	`NsAvalability:`
	`SUMOD: true`
	`EANAN: true`
		Bad request 400 Error:
		Unsupported Feature requested

Table 4-15 (Cont.) Error Scenarios

Scenario	Helm Configuration	Output

NSAvailability request with supported feature. That is,
'4'
	`SupportedFeatureNegotiationEnable: true`
	`threegppFeatures:`
	`NsAvalability:`
	`SUMOD: true`
	`EANAN: false`
		Bad request 400 Error:
		Unsupported Feature requested

NSAvailability request with supported feature. That is,
'2'
	`SupportedFeatureNegotiationEnable: true`
	`threegppFeatures:`
	`NsAvalability:`
	`SUMOD: false`
	`EANAN: true`
		Bad request 400 Error:
		Unsupported Feature requested

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.24 Subscription Modification Feature

This feature allows operator to modify subscription by supporting HTTP Patch on NsAvailability subscribe service operation (/nssai-availability/subscriptions/). Supported operations on HTTP Patch are ADD, REMOVE, and REPLACE. Whereas, COPY, MOVE, and TEST are not supported.

Managing Subscription Modification

Enable

To enable this feature, set the `global.SupportedFeatureNegotiationEnable` and `global.threegppFeatures.NsAvailability.SUMOD` parameters to `true` under the `global` section in the `ocnssf_custom_values_25.2.200.yaml` file.

The following snippet represents the location of the mentioned parameters in the Helm file:

```
global:
 SupportedFeatureNegotiationEnable: true
```

```
threegppFeatures:
   NsAvalability:
     SUMOD: true
```

Configure

There are no additional configurations required.

Observe

- Added the following error measurements:
  - ocnssf_nssaiavailability_submod_error_response_tx_total
  - ocnssf_nssaiavailability_submod_unimplemented_op_total
  - ocnssf_nssaiavailability_submod_patch_apply_error_total

For information about other Metrics and KPIs of NSSF, see NSSF Metrics and NSSF KPIs sections respectively.

Error Scenarios

Table 4-16 Error Scenarios

Scenario	Microservice	Description
Patch request processing failed due to invalid path	NsSubscription	**Request URL**: nnssf-nssaiavailability/v1/nssai-availability/subscriptions/ **Response Code/ Error Title**: 400 400 Bad Request Error Jason Patch Req processing failed
Subscription with HTTP Patch (Option ADD). Subscription present and TAI addition not supported in PLMN.	NsSubscription	**Request URL**: nnssf-nssaiavailability/v1/nssai-availability/subscriptions/ **Response Code/ Error Title**: HTTP 403 Unsupported PLMN Error Details must specify supported PLMN list
Subscription with HTTP Patch (Option ADD). Duplicate TAI addition.	NsSubscription	**Request URL:** nnssf-nssaiavailability/v1/nssai-availability/subscriptions/ **Response Code/ Error Title:** HTTP 400 Bad Request MANDATORY_IE_INCORRECT
SUBMOD is set to false	NsSubscription	**Request URL**: nnssf-nssaiavailability/v1/nssai-availability/subscriptions/ **Response Code/ Error Title**: HTTP 405 Method not allowed

Table 4-16 (Cont.) Error Scenarios

Scenario	Microservice	Description
Subscription ID is not found	NsSubscription	**Request URL**: nnssf-nssaiavailability/v1/nssai-availability/subscriptions/ **Response Code/ Error Title**: HTTP 404 Not Found

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.25 Empty Authorized NSSAI Availability Notification

Empty Authorized NSSAI Availability Notification (EANAN) feature provides support for sending an empty array of Authorized NSSAI Availability Data when a notification trigger leads to a situation of no Tracking Area (TA) with Authorized NSSAI by the NSSF. An Access and Mobility Management Function (AMF) that supports this feature accepts the empty array of Authorized NSSAI Availability Data in a notification from NSSF and deletes locally stored Authorized NSSAI Availability Data that was received previously.

Managing EANAN

Enable

To enable this feature, set the `global.SupportedFeatureNegotiationEnable` and `global.threegppFeatures.NsAvailability.EANAN` parameters to `true` under the `global` section in the `ocnssf_custom_values_25.2.200.yaml` file.

The following snippet represents the location of the mentioned parameters in the Helm file:

```
global:
 SupportedFeatureNegotiationEnable: true
 threegppFeatures:
  NsAvailability:
    EANAN: true
```

Configure

There are no additional configurations required.

Observe

No new metrics or KPIs are generated for this feature. For information on other Metrics and KPIs of OCNSSF, see OCNSSF Metrics and OCNSSF KPIs sections respectively.

Scenarios

Table 4-17 Scenarios

Scenario	Helm Configuration	Details

Send empty notification when EANAN is supported by both NSSF and Consumer NF for delete as notification trigger	`global.SupportedFeatureNegotiationEnable : true`
	
	`global.threegppFeatures.NsAvailability.EANAN : true`
		Subscription Test CaseSteps
		1. Configure supported slices mapping to allow **S-NSSAI-1** in **TAI-1**.
		2. Send a **subscription request** for **TAI-1** with `supportedFeatures` flag where the **EANAN** bit is set to `true`.
		3. Delete the supported slices mapping.
		Expected Output
		1. Configuration of supported slices mapping is successful.
		2. Subscription response contains **Authorized NSSAI Availability Data** as **S-NSSAI-1** for **TAI-1**, with the **EANAN** bit in `supportedFeatures` set to `true`.
		3. Supported slices mapping is deleted successfully.
		4. Notification from NSSF contains **empty Authorized NSSAI Availability Data**.

Table 4-17 (Cont.) Scenarios

Scenario	Helm Configuration	Details

Do not send empty notification when EANAN is supported by NSSF and not by Consumer NF	`global.SupportedFeatureNegotiationEnable : true`
	
	`global.threegppFeatures.NsAvailability.EANAN : true`
		Subscription Test CaseSteps
		1. Configure supported slices mapping to allow **S-NSSAI-1** in **TAI-1**.
		2. Send a **subscription request** for **TAI-1** with `supportedFeatures` flag where the **EANAN** bit is set to `false`.
		3. Delete the supported slices mapping.
		Expected Output
		1. Configuration of supported slices mapping is successful.
		2. Subscription response contains **Authorized NSSAI Availability Data** as **S-NSSAI-1** for **TAI-1**, with the **EANAN** bit in `supportedFeatures` set to `false`.
		3. Supported slices mapping is deleted successfully.
		4. **No notification** is sent out from NSSF.

Table 4-17 (Cont.) Scenarios

Scenario	Helm Configuration	Details

Do not send empty notification when EANAN is not supported by NSSF and supported by Consumer NF	`global.SupportedFeatureNegotiationEnable : true`
	
	`global.threegppFeatures.NsAvailability.EANAN : false`
		Subscription Test CaseSteps
		1. Configure supported slices mapping to allow **S-NSSAI-1** in **TAI-1**.
		2. Send a **subscription request** for **TAI-1** with `supportedFeatures` flag where the **EANAN** bit is set to `true`.
		3. Delete the supported slices mapping.
		Expected Output
		1. Configuration of supported slices mapping is successful.
		2. Subscription response contains **Authorized NSSAI Availability Data** as **S-NSSAI-1** for **TAI-1**, with the **EANAN** bit in `supportedFeatures` set to `false`.
		3. Supported slices mapping is deleted successfully.
		4. **No notification** is sent out from NSSF.

Table 4-17 (Cont.) Scenarios

Scenario	Helm Configuration	Details

Do not send empty notification when EANAN is not supported by NSSF and supported by Consumer NF	`global.SupportedFeatureNegotiationEnable : true`
	
	`global.threegppFeatures.NsAvailability.EANAN : false`
		Subscription Test CaseSteps
		1. Configure supported slices mapping to allow **S-NSSAI-1** in **TAI-1**.
		2. Send a **subscription request** for **TAI-1** with `supportedFeatures` flag where the **EANAN** bit is set to `true`.
		3. Delete the supported slices mapping.
		Expected Output
		1. Configuration of supported slices mapping is successful.
		2. Subscription response contains **Authorized NSSAI Availability Data** as **S-NSSAI-1** for **TAI-1**, but with the **EANAN** bit in `supportedFeatures` set to `false`.
		3. Supported slices mapping is deleted successfully.
		4. **No notification** is sent out from NSSF.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.26 Georedundancy

NSSF supports up to three-site Georedundancy to ensure service continuity when one of the NSSF sites is down. When NSSF is deployed as georedundant NSSF instances, then:

- All the sites that register with NRF work independently and are in Active state.
- Based on the Rank, each NSSF site subscribes to NRF for any state change of other NSSF sites and gets notified when an NSSF site goes down.
- All NSSF sites retry subscription to the NRF for `nfType` NSSF if the initial attempt fails. The NSSF will continue retrying at fixed intervals until the subscription to the NRF is successful.

- The NFs in a given site need to configure one of the georedundant NSSF as the primary NSSF and others as secondary NSSF and tertiary NSSF, respectively.
- When the primary NSSF is available, the NFs send service requests to the primary NSSF. When the NSSF at the primary site is unavailable, the NFs redirect service requests to the secondary NSSF or tertiary NSSF, until the primary NSSF's Active status is restored.
- Priority based NSSF selection (at NF Consumer or SCP) can be implemented to ensure route traffic based on which NSSF site is up.

The NSSF's data gets replicated between the georedundant sites by using DB tier's replication service.

With NSSF georedundant feature, the NSSF Services (NSSelection and NSAvailability) will continue to work as independent service operations.

Following are the prerequisites for georedundancy:

- Each site must configure remote NSSF sites as georedundant mates.
- The configurations at each site must be same. The NSSF at all sites must handle the NFs in the same manner.
- Once the Georedundancy feature is enabled on a site, it cannot be disabled.
- If the Time Of the Day (TOD) feature is enabled, georedundant sites are time synchronized.
- NFs need to configure georedundant NSSF details as Primary, Secondary, and Tertiary NSSFs.
- Georedundant sites must have REST based configuration as explained in *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.
- At any given time, NFs must communicate with only one NSSF. That is, NFs must register services and maintain heartbeats with only one NSSF. The data must be replicated across the georedundant NSSFs, allowing seamless NF mobility across NSSFs as required.

Managing NSSF Georedundancy Feature

Prerequisites

Following are the prerequisites to enable georedundancy feature in NSSF:

- cnDBTier must be installed and configured for each site. For the installation procedure, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide.*
- The Database Replication Channels between the sites must be up.
- Configure MySQL Database, Users and Secrets. For the configuration procedure, see **Preinstallation Tasks** in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation an Upgrade Guide.*

Enable Georedundancy Feature

Configure the following to enable the georedundancy feature:

> **Note**
>
> Configuring these attributes during deployment is mandatory before enabling the georedundancy feature. Otherwise, georedundancy cannot be enabled, and NSSF at the site will act as a stand-alone NSSF.

Helm Configuration for Database:

Configure the following parameters in the `ocnssf_custom_values_25.2.200.yaml` file for all three sites:

- `global.leaderPodDbName`
- `global.nrfClientDbName`
- `global.stateDbName`
- `global.provisionDbName`
- `global.releaseDbName`
- `global.nameSpace`
- `global.mysql.primary.host`

At Site 1:

```
global:


 # Mysql NSSF Database Names
  stateDbName: 'nssfStateDB'
  provisionDbName: &provDB 'nssfProvSite1DB'



  # Mysql Release Database Name used to maintain release version
  releaseDbName: 'ocnssfReleaseDB'



  # NameSpace where secret is deployed
  nameSpace: &ns ocnssf1


  # Database configuration
  mysql:
    primary:
      host: &dbHost "mysql-connectivity-service.site1"
```

At Site 2:

```
global:

 # Mysql NSSF Database Names
  stateDbName: 'nssfStateDB'
  provisionDbName: &provDB 'nssfProvSite2DB'
```

```
  # Mysql Release Database Name used to maintain release version
  releaseDbName: 'ocnssfRelease2DB'

  # NameSpace where secret is deployed
  nameSpace: &ns ocnssf2


  # Database configuration
  mysql:
    primary:
      host: &dbHost "mysql-connectivity-service.site2"
```

At Site 3:

```
global:

 # Mysql NSSF Database Names
  stateDbName: 'nssfStateDB'
  provisionDbName: &provDB 'nssfProvSite3DB'

  # Mysql Release Database Name used to maintain release version
  releaseDbName: 'ocnssfRelease3DB'

  # NameSpace where secret is deployed
  nameSpace: &ns ocnssf3


  # Database configuration
  mysql:
    primary:
      host: &dbHost "mysql-connectivity-service.site3"
```

Helm Configuration of Parameters:

Configure the following parameters in the `ocnssf_custom_values_25.2.200.yaml` file for all three sites:

- `global.grEnabled`
- `global.nfInstanceId`
- `global.siteId`
- If `global.grEnabled` is set to `true`, Configure the following parameters as well:
  - `global.grEnv.maxSecondsBehindRemote`
  - `global.grEnv.dbMonitorServiceUrl`
  - `global.grEnv.peerGRSitesList.siteId`
  - `global.grEnv.peerGRSitesList.nfInstanceId`

For more information about configuring the parameters, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

Following is the sample configuration at Site named "**site1**" (nfInstanceId: `9faf1bbc-6e4a-4454-a507-aef01a101a01`), which is georedundant with Sites, "**site2**"

(NssfInstanceId: 9faf1bbc-6e4a-4454-a507-aef01a101a02) and "**site3**" (NssfInstanceId: 9faf1bbc-6e4a-4454-a507-aef01a101a03):

```
global:
  #Only applicable for NSSF microservices
  #=====================================================================
  # GR params
  #tag to enable GR
  grEnabled: true
  #InstanceId of NSSF used in case of GR
  nfInstanceId: "9faf1bbc-6e4a-4454-a507-aef01a101a01"
  #SiteID of NSSF used in case of GR
  siteId: "site1"
  #All parameters under this section are valid only if grEnabled is true
  grEnv:
    #Maximum allowed seconds behind remote site for replication
    maxSecondsBehindRemote: 5
    #URL to check db-replication status
    dbMonitorServiceUrl: "http://mysql-cluster-db-monitor-svc.site1:8080/db-
tier/status/replication/realtime"
    #GR sites list
    peerGRSitesList:
      - siteId: "site2"
      - nfInstanceId: "9faf1bbc-6e4a-4454-a507-aef01a101a02"
      - siteId: "site3"
      - nfInstanceId: "9faf1bbc-6e4a-4454-a507-aef01a101a03"
```

Following is the sample configuration at Site named "**site2**" (NssfInstanceId: 9faf1bbc-6e4a-4454-a507-aef01a101a02), which is georedundant with Sites, "**site1**" (NssfInstanceId: 9faf1bbc-6e4a-4454-a507-aef01a101a01) and "**site3**" (NssfInstanceId: 9faf1bbc-6e4a-4454-a507-aef01a101a03):

```
global:
   #Only applicable for NSSF microservices
  #=====================================================================
  # GR params
  #tag to enable GR
  grEnabled: true
  #InstanceId of NSSF used in case of GR
  nfInstanceId: "9faf1bbc-6e4a-4454-a507-aef01a101a02"
  #SiteID of NSSF used in case of GR
  siteId: "site2"
  #All parameters under this section are valid only if grEnabled is true
  grEnv:
    #Maximum allowed seconds behind remote site for replication
    maxSecondsBehindRemote: 5
    #URL to check db-replication status
    dbMonitorServiceUrl: "http://mysql-cluster-db-monitor-svc.site2:8080/db-
tier/status/replication/realtime"
    #GR sites list
    peerGRSitesList:
      - siteId: "site1"
      - nfInstanceId: "9faf1bbc-6e4a-4454-a507-aef01a101a01"
      - siteId: "site3"
      - nfInstanceId: "9faf1bbc-6e4a-4454-a507-aef01a101a03"
```

Following is the sample configuration at Site named "**site3**" (NssfInstanceId: `9faf1bbc-6e4a-4454-a507-aef01a101a03`), which is georedundant with Sites, "**site1**" (NssfInstanceId: `9faf1bbc-6e4a-4454-a507-aef01a101a01`) and "**site2**" (NssfInstanceId: `9faf1bbc-6e4a-4454-a507-aef01a101a02`)

```
global:
  # GR params
  #tag to enable GR
  grEnabled: true
  #InstanceId of NSSF used in case of GR
  nfInstanceId: "9faf1bbc-6e4a-4454-a507-aef01a101a03"
  #SiteID of NSSF used in case of GR
  siteId: "site3"
  grEnv:
    #Maximum allowed seconds behind remote site for replication
    maxSecondsBehindRemote: 5
    #URL to check db-replication status
    dbMonitorServiceUrl: "http://mysql-cluster-db-monitor-svc.site3:8080/db-
tier/status/replication/realtime"
    #GR sites list
    peerGRSitesList:
      - siteId: "site1"
      - nfInstanceId: "9faf1bbc-6e4a-4454-a507-aef01a101a01"
      - siteId: "site2"
      - nfInstanceId: "9faf1bbc-6e4a-4454-a507-aef01a101a02"
```

Observe

No new metrics or KPIs are generated for this feature. For information on other Metrics and KPIs of OCNSSF, see OCNSSF Metrics and OCNSSF KPIs sections respectively.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.27 Multiple PLMN Support

This feature enables single NSSF instance to cater to multiple PLMNs. This enables operator to define slice selection policies for multiple PLMNs, and gives the option for operator to span a network slice across PLMNs.
NSSF allows the user to add the supported PLMN list which must be used for registering with NRF. Any change in supported PLMN list must trigger a Register request towards NRF with updated profile. Requests which have TAI containing other PLMN will be treated as roaming.

Managing Multiple PLMN Support

Enable

This feature is enabled by default. Once the operator configures multiple `PLMNs` in the managed object (`SystemOptions`), NSSF starts supporting multiple `PLMNs`.

To enable support of multiple PLMNs, the operator has to run the SystemOptions PUT REST API.

API Details

- **URL:**/nnssf-configuration/v1/systemoptions
- **Method:**PUT

Request Payload:

```
{
  "autoAuthorizeNssaiAvailabilityDataEnable": true,
  "enhancedPatchBehaviourEnable": true,
  "plmnLevelSystemOptionsList": [
    {
      "plmnId": "311-480",
      "enhancedAllowedNssaiEnable": true
    },
    {
      "plmnId": "100-101",
      "enhancedAllowedNssaiEnable": true
    }
  ]
}
```

Disable

This feature support is inherent in NSSF. In case the operator wants to use a single PLMN, SystemOptions can be configured as shown below:

Sample to Configure a Single PLMN

- **URL:**/nnssf-configuration/v1/systemoptions
- **Method:**PUT

Request Payload:

```
{
  "autoAuthorizeNssaiAvailabilityDataEnable": true,
  "enhancedPatchBehaviourEnable": true,
  "plmnLevelSystemOptionsList": [
    {
      "plmnId": "311-480",
      "enhancedAllowedNssaiEnable": true
    }
  ]
}
```

> **Note**
>
> If there is no PLMN configured in SystemOptions, it indicates that no PLMNs are permitted. This is an undesirable configuration; there must be at least one PLMN supported.

Observe

Following are the metrics related to Multiple PLMN Support:
ocnssf_nsselection_unsupported_plmn_total

ocnssf_nsavailability_unsupported_plmn_total

For further information about the Metrics and KPIs, see NSSF Metrics and NSSF KPIs sections respectively.

Error Scenarios

Table 4-18 Error Scenarios

Scenario	Microservice	Details
Request comes from unknown PLMN	NSSelection	**Request URL:** /nnssf-nsselection/v1/network-slice-information/ **Response Code / Error Title:** 403 - PLMN_NOT_SUPPORTED **Notes:** No query sent to DB. Look into configured PLMNs and respond
Subscription request for unknown PLMN (Subscription request contains one or more TAIs which belong to unsupported PLMN)	NSAvailability	**Request URL:** /nnssf-nssaiavailability/v1/nssai-availability/subscriptions **Response Code / Error Title:** 403 - PLMN_NOT_SUPPORTED **Notes:** No query sent to DB as none of the PLMNs are supported
NssaiAvailability request containing unknown PLMN/s (Availability request contains one or more TAIs which belong to unsupported PLMN)	NSAvailability	**Request URL:** nnssf-nssaiavailability/v1/nssai-availability **Response Code / Error Title:** 403 - PLMN_NOT_SUPPORTED **Notes:** No query sent to DB as none of the PLMNs are supported
Operator tries to configure unsupported PLMN (request contains one or more unsupported PLMNs)	NSConfig	**Request URL:** /nnssf-configuration/v1/plmnconfig **Response Code / Error Title:** 403 - PLMN_NOT_SUPPORTED **Notes:** Currently, as we are not supporting the roaming scenario, the operator must not be allowed to add policy configuration for unknown PLMNs.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.28 Support Indirect Communication

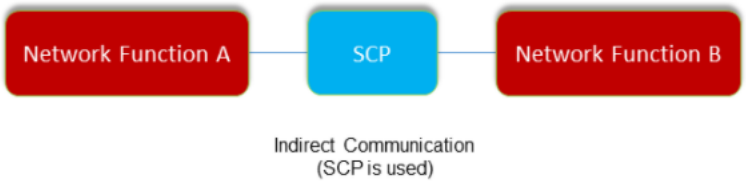
3GPP TS 29.531 Release 16 has introduced a new NF SCP which enables reliability and resiliency within network.

In indirect mode of communication consumers and producers interact through SCP. There are two communication models as described below:

Model C - Indirect communication without delegated discovery: Consumers do discovery by querying the NRF. Based on discovery result, the consumer does the selection of an NF Set or a specific NF instance of NF set. The consumer sends the request to the SCP containing the address of the selected service producer pointing to a NF service instance or a set of NF service instances. In the later case, the SCP selects an NF Service instance. If possible, the SCP interacts with NRF to get selection parameters such as Location, capacity, etc. The SCP routes the request to the selected NF service producer instance.

Model D - Indirect communication with delegated discovery: Consumers do not perform any discovery or selection. The consumer adds any necessary discovery and selection parameters required to find a suitable producer to the service request. The SCP uses the request address and the discovery and selection parameters in the request message to route the request to a suitable producer instance. The SCP can perform discovery with an NRF and obtain a discovery result

Once this feature is enabled on NSSF, it allows consumer NFs (AMF) to perform routing and rerouting to NSSF through SCP leveraging following 3GPP headers "3gpp-Sbi–Binding" and "3gpp-Sbi--Routing-Binding".



Note

- This feature's scope involves the manipulation and updating of headers and values.
- It does not mandate that Notifications must go through SCP. The responsibility for configuring SCP to route the notifications is on the operator. For more information, see DNS SRV Based Selection of SCP in NSSF.
- NSSF only supports the following pattern of 3gpp-Sbi-Binding Header:
  `bl=nf-set;`
  `nfset=set<setId>.region<regionId>.amfset.5gc.mnc<mnc>.mcc<mcc>`
- NSSF only accepts subscription with 3gpp-Sbi-Binding from AMF, provided AMF must be a part of the AMF-Set.

.

Managing Indirect Communication

Enable

To enable this feature, set the value of `indirectCommunicationSupportEnable` to `true` in the `ocnssf_custom_values_25.2.200.yaml` file.

```
# Indirect communication support
indirectCommunicationSupportEnable: true
```

The scope of this feature is Subscription and Notification flow.

When this feature is enabled:

- When AMF sends a NsAvailability Subscribe with 3gpp-Sbi-Binding header, NSSF validates if the header Supported Format is:

  ```
  bl=nf-set;
  nfset=set<setId>.region<regionId>.amfset.5gc.mnc<mnc>.mcc<mcc>
  ```

  - Only the following format is supported:

    ```
    bl=nf-set;
    nfset=set<setId>.region<regionId>.amfset.5gc.mnc<mnc>.mcc<mcc>
    ```

  - In case of a successful validation, the NSSF responds with a 201 status code. The response includes a "3gpp-Sbi-Binding" header containing NSSF's binding information.
  - The NSSF computes its binding information by matching the NSSF set details for the corresponding PLMN.
  - The NSSF stores the Binding header of the AMF set in the database.
  - When sending a notification for the subscription, the same value from the AMF's binding header is included in the notification as the "3gpp-Sbi-Routing-Binding" header.
  - Additionally, the NSSF adds a "3gpp-Sbi-Callback" header with the value `Nnssf_NSSAIAvailability_Notification`.
- If the AMF sends a NsAvailability Subscribe request without the "3gpp-Sbi-Binding" header:
  - The NSSF responds without including the "3gpp-Sbi-Binding" header in the response.
  - The NSSF does not add the "3gpp-Sbi-Routing-Binding" header in the notification.
  - The processing of the request remains unchanged, and there is no impact on the processing and response, except that the mentioned headers are not computed or included as specified above.

Disable

To disable this feature, set the value of `indirectCommunicationSupportEnable` to `false` in the `ocnssf_custom_values_25.2.200.yaml` file.

- When `indirectCommunicationSupportEnable` is set to `false`:
  - When AMF sends a NsAvailability Subscribe with 3gpp-Sbi-Binding header:
    * NSSF ignores the header and process the request.

* NSSF does not add 3gpp-Sbi-Routing-Binding header in the notification.

```
# Indirect communication support
indirectCommunicationSupportEnable: false
```

Observe

- Added the following success measurements:
  - ocnssf_indirect_communication_request_rx_total
  - ocnssf_indirect_communication_response_tx_success_total
  - ocnssf_nssaiavailability_notification_indirect_communication_tx_total
  - ocnssf_nssaiavailability_notification_indirect_communication_rx_total
- Added the following error measurements:
  - ocnssf_nssaiavailability_indirect_communication_subscription_failure_total
  - ocnssf_indirect_communication_response_tx_failure_total
  - ocnssf_nssaiavailability_indirect_communication_notification_failure_total

For more information on above metrics and KPIs, see NSSF Metrics and NSSF KPIs.

Error Scenarios

Table 4-19 Error Scenarios

Scenario	Input Details	Output

Subscription with binding, global.indirectCommunicationSupportEnable is set to true, NSSF is part of nfSet.	**Input message:**
	Subscribe with header
	3gpp-Sbi--Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc012.mcc345
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 201 Created
		Headers
		Location: http://10.178.246.56:30075/nnssf-nssaiavailability/v1/nssai-availability/subscriptions/1
		3gpp-Sbi-Binding: bl=nf-set; nfset=set1.nssfset.5gc.mnc012.mcc345
		Notification with headers
		3gpp-Sbi-Routing-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc012.mcc345
		3gpp-Sbi-Callback: Nnssf_NSSAIAvailability_Notification

Subscription without binding, global.indirectCommunicationSupportEnable is set to true, NSSF is part of nfSet.	**Input message:**
	Subscribe without Header (Direct)
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 201 Created
		Location: http://10.178.246.56:30075/nnssf-nssaiavailability/v1/nssai-availability/subscriptions/1

Table 4-19 (Cont.) Error Scenarios

Scenario	Input Details	Output

Subscription with binding, global.indirectCommunicationSupportEnable is set to false, NSSF is part of nfSet.	**Input message:**
	Subscribe with header
	3gpp-Sbi-Routing-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc012.mcc345
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: false`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 201 Created

Subscription without binding, global.indirectCommunicationSupportEnable is set to false, NSSF is part of nfSet.	**Input message:**
	Subscribe without Header (Direct)
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: false`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 201 Created

Subscription with binding, global.indirectCommunicationSupportEnable is set to true, NSSF is not part of nfSet.	**Input message:**
	Subscribe with Header
	3gpp-Sbi-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc012.mcc345
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet:` Not part of nf-set but part of GR
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 201 Created
		Headers
		Location: http://10.178.246.56:30075/nnssf-nssaiavailability/v1/nssai-availability/subscriptions/1
		3gpp-Sbi-Routing-Binding: bl=nf-instance; nfinst=54804518-4191-46b3-955c-ac631f953ed7; backupnfinst=54804518-4191-46b3-955c-ac631f953ed8
		Notification with headers
		3gpp-Sbi-Routing-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc012.mcc345
		3gpp-Sbi-Callback: Nnssf_NSSAIAvailability_Notification
		ERROR Log

Table 4-19 (Cont.) Error Scenarios

Scenario	Input Details	Output

Subscription with binding, global.indirectCommunicationSupportEnable is set to true, NSSF is not part of nfSet.	**Input message:**
	Subscribe with Header
	3gpp-Sbi-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc012.mcc345
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet:` Not part of nf-set and not part of GR
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status 500 Internal Server Error
		Cause: CONFIGURATION_ERROR
		
		`{ "type": "INTERNAL_SERVER_ERROR", "title": "CONFIGURATION_ERROR", "status": 500, "detail": "Indirect Communication is true but NFset is null and GR is also not enabled.", "instance": "null", "cause": "CONFIGURATION_ERROR" }`

Subscription with binding, global.indirectCommunicationSupportEnable is set to true, NSSF is part of nfSet.	**Input message:**
	Subscribe with Header
	3gpp-Sbi-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc012.mcc345
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nssfApiRoot: Invalid URL(Empty)`
		Subscription Response
		Status: 500 Internal Server Error
		Cause: CONFIGURATION_ERROR
		
		`{ "type": "INTERNAL_SERVER_ERROR", "title": "INVALID_LOCATION_URL", "status": 500, "detail": "Invalid location/ nssfApiRoot url", "instance": "null", "cause": "CONFIGURATION_ERROR" }`

NSSF supports multiple PLMNSubscription with binding, global.indirectCommunicationSupportEnable is set to true, NSSF is part of nfSet. NSSF supports PLMN from which AMF is requesting	**Input message:**
	Subscribe with Header3gpp-Sbi-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc100.mcc101
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nfSet: set1.nssfset.5gc.mnc100.mcc101`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 201 Created
		Headers
		Location: http://10.178.246.56:30075/nnssf-nssaiavailability/v1/nssai-availability/subscriptions/1
		3gpp-Sbi-Binding: bl=nf-set; nfset=set1.nssfset.5gc.mnc100.mcc101
		Notification with headers
		3gpp-Sbi-Routing-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc100.mcc101
		3gpp-Sbi-Callback: Nnssf_NSSAIAvailability_Notification

Table 4-19 (Cont.) Error Scenarios

Scenario	Input Details	Output

NSSF supports multiple PLMNSubscription with binding, global.indirectCommunicationSupportEnable is set to true, NSSF is part of nfSet. NSSF do not support PLMN from which AMF is requesting	**Input message:**
	Subscribe with Header
	3gpp-Sbi-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc200.mcc201
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nfSet: set1.nssfset.5gc.mnc100.mcc101`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 403
		Cause: PLMN_NOT_SUPPORTED
		
		`{ "type": "FORBIDDEN", "title": "PLMN_NOT_SUPPORTED", "status": 403, "detail": "Unsupported PLMN/S received , supported plmn list: [Plmn [mcc=100, mnc=101], Plmn [mcc=100, mnc=02], Plmn [mcc=310, mnc=14], Plmn [mcc=345, mnc=012]]", "instance": "null", "cause": "PLMN_NOT_SUPPORTED" }`

Subscription with invalid binding header, global.indirectCommunicationSupportEnable is set to true, NSSF is part of nfSet.	**Input message:**
	Subscribe with Header
	3gpp-Sbi-Binding: bl=nfserviceset; nfset=set1.region48.amfset.5gc.mnc012.mcc345
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 400 Bad Request
		Cause: INVALID_INPUT_DATA
		
		`{ "type": "BAD_REQUEST", "title": "INVALID_INPUT_DATA", "status": 400, "detail": "Invalid 3gpp-Sbi-Binding Header, Only following pattern is supported bl=nf-set; nfset=set<setId>.region<regionId>.amfset.5gc.mnc<mnc>.mcc<mcc>", "instance": "null", "cause": "INVALID_INPUT_DATA" }`

Table 4-19 (Cont.) Error Scenarios

Scenario	Input Details	Output

Subscription with invalid binding header, global.indirectCommunicationSupportEnable is set to true, NSSF is part of nfSet.	**Input message:**
	Subscribe with Header3gpp-Sbi-Binding: bl=nf-set nfset=set1.amfset.5gc.mnc012.mcc345
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 400 Bad Request
		Cause: INVALID_INPUT_DATA
		
		`{ "type": "BAD_REQUEST", "title": "INVALID_INPUT_DATA", "status": 400, "detail": "Invalid 3gpp-Sbi-Binding Header, Only following pattern is supported bl=nf-set; nfset=set<setId>.region<regionId>.amfset.5gc.mnc<mnc>.mcc<mcc>", "instance": "null", "cause": "INVALID_INPUT_DATA" }`

Subscription with invalid binding header, global.indirectCommunicationSupportEnable is set to true, NSSF is part of nfSet.	**Input message:**
	Subscribe with Header
	3gpp-Sbi-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc8120.mcc345
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 400 Bad Request
		Cause: INVALID_INPUT_DATA
		
		`{ "type": "BAD_REQUEST", "title": "INVALID_INPUT_DATA", "status": 400, "detail": "Invalid mnc 8120 in 3gpp-Sbi-Binding Header", "instance": "null", "cause": "INVALID_INPUT_DATA" }`
		
		Description: Invalid mnc 8120 in 3gpp-Sbi-Binding Header

Table 4-19 (Cont.) Error Scenarios

Scenario	Input Details	Output

Subscription with invalid binding header, global.indirectCommunicationSupportEnable is set to true, NSSF is part of nfSet.	**Input message:**
	Subscribe with Header
	3gpp-Sbi-Binding: bl=nf-set; nfset=set1.region48.amfset.5gc.mnc8120.mcc345;
	scope=callback; scope=other-service
	Helm Parameters and Values:
	`global.indirectCommunicationSupportEnable: true`
	`global.nfSet: set1.nssfset.5gc.mnc012.mcc345`
	`global.nssfApiRoot: http://10.178.246.56:30075/`
		Subscription Response
		Status: 400 Bad Request
		Cause: INVALID_INPUT_DATA
		`{ "type": "BAD_REQUEST", "title": "INVALID_INPUT_DATA", "status": 400, "detail": "Invalid 3gpp-Sbi-Binding Header, Only following pattern is supported bl=nf-set; nfset=set<setId>.region<regionId>.amfset.5gc.mnc<mnc>.mcc<mcc>", "instance": "null", "cause": "INVALID_INPUT_DATA" }`
		Description: After 3 digits of MCC, there are other characters

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.29 Supports Integration with ASM

NSSF leverages the Istio or Envoy service mesh (Aspen Service Mesh) for all internal and external communication. The service mesh integration provides inter-NF communication and allows API gateway co-working with service mesh. The service mesh integration supports the services by deploying a special sidecar proxy in the environment to intercept all network communication between microservices.

See **Configuring NSSF to support ASM** in *Oracle Communication Cloud Native Core, Network Slice Selection Function Installation and Upgrade Guide* for more details on configuring ASM.

4.30 Supports Compression Using Accept-Encoding or Content-Encoding gzip

HTTP data is compressed before it is sent from the server to improve transfer speed and bandwidth utilization.

HTTP headers let the client and the server pass additional information with an HTTP request or response.

The `Content-Encoding` header, when present in a response, indicates which encoding is applied to the entity body. It lets the client know how to decode the content in order to obtain the media type referenced by the `Content-Type` header.

The `Accept-Encoding` header is used to find out the encoding supported by the server. The server responds with the type of encoding used, as indicated by the `Accept-Encoding` response header.

Syntax:

```
Accept-Encoding:gzip
content-encoding:gzip
Content-Type:application/json
```

Managing Supports Compression Using Accept-encoding/Content-encoding gzip

Configure
This sample configuration shows the minimum response size over which compression of the response is triggered.

The Helm parameter `maxPayloadSize` is the acceptable payload size of the request.

```
#Sample configuration gzip compression
  # Minimum response size required for compression to happen (size is in
bytes)
  nsavailability.compressionMinimumResponseSize: 1024
  # Maximum limit for request payload size (size in KB)
  nsavailability.maxPayloadSize: 500
```

Observe
The following measurements are related to *Supports compression using Accept-encoding/ Content-encoding gzip* feature:

ocnssf_nssaiavailability_options_rx_total

ocnssf_nssaiavailability_options_tx_status_ok_total

ocnssf_nssaiavailability_options_tx_status_unsupportedmediatetype_total

For further information about Metrics and KPIs, see NSSF Metrics and NSSF KPIs sections respectively.
Message Scenarios

Table 4-20 Message Scenarios

Scenario	Helm Parameter (nsavailability.contentEncodingEnabled)	Response Details
AMF sends an NSAvailability PUT with Request Message size is more than max acceptable size.	NA	**Response code**: 413 (Request Entity Too Large error) **Response in gzip**: No **Response Header**: NA
Client sends HTTP OPTIONS with "Accept-encoding" of any value (blank or empty included) other than gzip.	Yes	**Response code**: 415 (Unsupported Media Type) **Response in gzip**: NA **Response Header**: Accept-Encoding: gzip Allowed Methods : PUT, PATCH, DELETE Reason: Informs the client to optimize future interactions

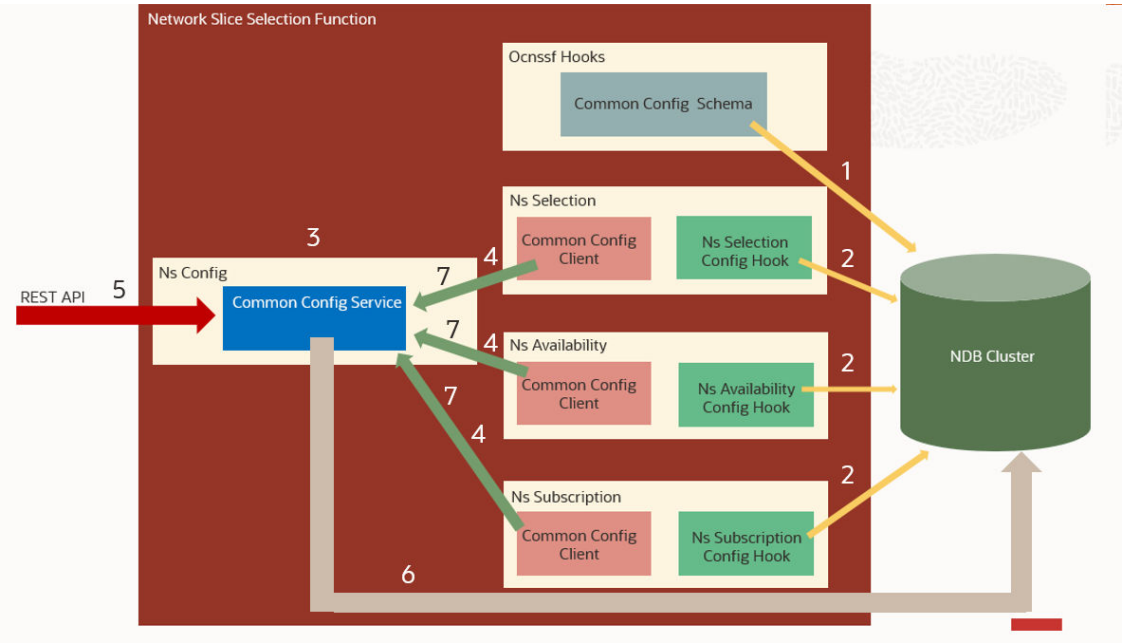
Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.31 Dynamic Log Level Update

Dynamic Log Level Update allows operator to update NSSF log level dynamically without restart.



The log level can be changed by the user at run time. NSSF use common configuration service for dynamically updating Logging Information.

Managing Dynamic Log Level Update

Enable

1. Customize the `ocnssf_custom_values_25.2.200.yaml` helm file.
2. Set `commonCfgClient.enabled` to `true` in the helm file.

Table 4-21 Parameters Configuration

Name	Default	Description
commonCfgClient.enabled	true	Enable/Disable Client.
commonCfgClient.pollingInterval	5000	Set Polling Interval in Milliseconds

Configure

REST API

For more information on REST API configuration, see **Runtime Log Level Update** section in the *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

CNC Console

For more information on CNC Console configuration, see Logging Level Options section.

Observe

No new metrics or KPIs are generated for this feature. For information on other Metrics and KPIs of OCNSSF, see OCNSSF Metrics and OCNSSF KPIs sections respectively.

Maintain

To resolve any alerts at the system or application level, see NSSF Alerts section. If the alerts persist, perform the following:

1. **Collect the logs**: For more information on how to collect logs, see *Oracle Communications Cloud Native Core, Network Slice Selection Function Troubleshooting Guide*.
2. **Raise a service request**: See My Oracle Support for more information on how to raise a service request.

4.32 NF Authentication using TLS Certificate

HTTPS support is a minimum requirement for 5G NFs as defined in 3GPP TS 33.501 Release 15. This feature enables extending identity validation from Transport layer to the Application layer and also provides a mechanism to validate the NF FQDN presence in TLS certificate as added by the Service Mesh against the NF Profile FQDN present in the request.
HTTPS enables end to end encryption of messages to ensure security of data. HTTPS requires creation of TLS (Mutual TLS by 2 way exchange of ciphered keys).

Managing NF Authentication using TLS Certificate

Steps to Enable HTTPS in NSSF

Certificate Creation
To create certificate user must have the following files:

- ECDSA private key and CA signed certificate of NRF (if initial algorithm is ES256)
- RSA private key and CA signed certificate of NRF (if initial algorithm is RSA256)
- TrustStore password file
- KeyStore password file
- CA certificate

Secret Creation

Execute the following command to create secret:

```
$ kubectl create secret generic ocnssfaccesstoken-secret --from-
file=ecdsa_private_key_pkcs8.pem --from-file=rsa_private_key_pkcs1.pem
      --from-file=trustStorePassword.txt --from-file=keyStorePassword.txt --
from-file=ecdsa_ocnssf_certificate.crt--from-file=rsa_ocnssf_certificate.crt -
n
    ocnssf
```

Certificate and Key Exchange

Once the connection is established, both parties can use the agreed algorithm and keys to securely send messages to each other. The handshake has 3 main phases:

- Hello
- Certificate Exchange
- Key Exchange

1. **Hello:** The handshake begins with the client sending a ClientHello message. This contains all the information the server needs in order to connect to the client via SSL, including the various cipher suites and maximum SSL version that it supports. The server responds with a ServerHello, which contains similar information required by the client, including a

decision based on the client's preferences about which cipher suite and version of SSL will be used.

2. **Certificate Exchange:** Now that contact has been established, the server has to prove its identity to the client. This is achieved using its SSL certificate, which is a very tiny bit like its passport. An SSL certificate contains various pieces of data, including the name of the owner, the property (For example: domain) it is attached to, the certificate's public key, the digital signature and information about the certificate's validity dates. The client checks that it either implicitly trusts the certificate, or that it is verified and trusted by one of several Certificate Authorities (CAs) that it also implicitly trusts. The server is also allowed to require a certificate to prove the client's identity, but this only happens in very sensitive applications.
3. **Key Exchange:** The encryption of the actual message data exchanged by the client and server is done using a symmetric algorithm, the exact nature of which was agreed during the Hello phase. A symmetric algorithm uses a single key for both encryption and decryption, in contrast to asymmetric algorithms that require a public or private key pair. Both parties need to agree on this single, symmetric key, a process that is accomplished securely using asymmetric encryption and the server's public or private keys.

The client generates a random key to be used for the main, symmetric algorithm. It encrypts it using an algorithm also agreed upon during the Hello phase, and the server's public key (found on its SSL certificate). It sends this encrypted key to the server, where it is decrypted using the server's private key, and the interesting parts of the handshake are complete. The parties are identified that they are talking to the right person, and have secretly agreed on a key to symmetrically encrypt the data that they are about to send each other. HTTP requests and responses can be sent by forming a plain text message and then encrypting and sending it. The other party is the only one who knows how to decrypt this message, and so Man In The Middle Attackers are unable to read or modify any requests that they may intercept.

NSSF supports following cipher suites

```
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
```

HTTPS Encrypted Communication
Once the HTTPS handshake is complete all communications between the client and the server are encrypted. This includes the full URL, data (plain text or binary), cookies and other headers.

The only part of the communication not encrypted is what domain or host the client requested a connection. This is because when the connection is initiated an HTTP request is made to the target server to create the secure connection. Once HTTPS is established the full URL is used.

This initialization only needs to occur once for each unique connection. This is why HTTP/2 has a distinct advantage over HTTP/1.1 since it multi-plexes connections instead of opening multiple connections.

Helm Configuration to enable HTTPS on NSSF:
Sample values.yaml to enable HTTPS on NSSF:

```
 #Enabling it generates key and trust store for https support
    initssl: true       (Note: secret has to be created if its set to true)
#If true opens https port on egress gateway
   enableincominghttps: false
```

```
#Enabling it egress makes https request outside
   enableoutgoinghttps: true
 (Note: initssl should be set to true if either enableincominghttps or
enableoutgoinghttps is enabled )
#KeyStore and TrustStore related private key and Certificate configuration
(Note: The configuration names specified should be same as the     file names
specified when creating secret)

   privateKey:
   k8SecretName: accesstoken-secret
   k8NameSpace: ocnssf
   rsa:
   fileName: rsa_private_key_pkcs1.pem

   certificate:
   k8SecretName: accesstoken-secret
   k8NameSpace: ocnssf
   rsa:
   fileName: ocnssf.cer

   caBundle:
   k8SecretName: accesstoken-secret
   k8NameSpace: ocnssf
   fileName: caroot.cer

   keyStorePassword:
   k8SecretName: accesstoken-secret
   k8NameSpace: ocnssf
   fileName: key.txt

   trustStorePassword:
   k8SecretName: accesstoken-secret
   k8NameSpace: ocnssf
   fileName: trust.txt

   initialAlgorithm: RSA256
```

4.33 Automated Testing Suite Support

NSSF provides Automated Testing Suite for validating the functionalities. Through Automated Testing Suite (ATS), Oracle Communications aims at providing an end-to-end solution to its customers for deploying and testing its 5G-NFs. See *Oracle Communications Cloud Native Core, Automated Testing Suite Guide* for more information.

5
Configuring NSSF using CNC Console

This chapter describes how to configure different NSSF managed objects using Oracle Communications Cloud Native Configuration Console (CNCC).

5.1 Support for Multicluster Deployment

CNC Console supports both single and multiple cluster deployments by facilitating NSSF deployment in local and remote Kubernetes clusters. For more information about single and multiple cluster deployments, see *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*.

A single instance of CNC Console can configure multiple clusters of NSSF deployments, where each cluster has an agent console installation and a NSSF installation.

5.2 CNC Console Interface

This section provides an overview of the Oracle Communications Cloud Native Configuration Console (CNCC), which includes an interface to configure the NSSF features.

To configure the NSSF services using the CNCC, log in to the CNCC application. To log into CNCC, update the hosts file available at the **C:\Windows\System32\drivers\etc location** when CNCC is hosted on a third party cloud native environment.

1. In the Windows system, open the hosts file in the notepad as an administrator and append the following set of lines at the end of the hosts file:

   ```
   <CNCC Node IP> cncc-iam-ingress-gateway.cncc.svc.cluster.local
   <CNCC Node IP> cncc-core-ingress-gateway.cncc.svc.cluster.local
   ```

   For example:

   ```
   10.75.212.88 cncc-iam-ingress-gateway.cncc.svc.cluster.local
   10.75.212.88 cncc-core-ingress-gateway.cncc.svc.cluster.local
   ```

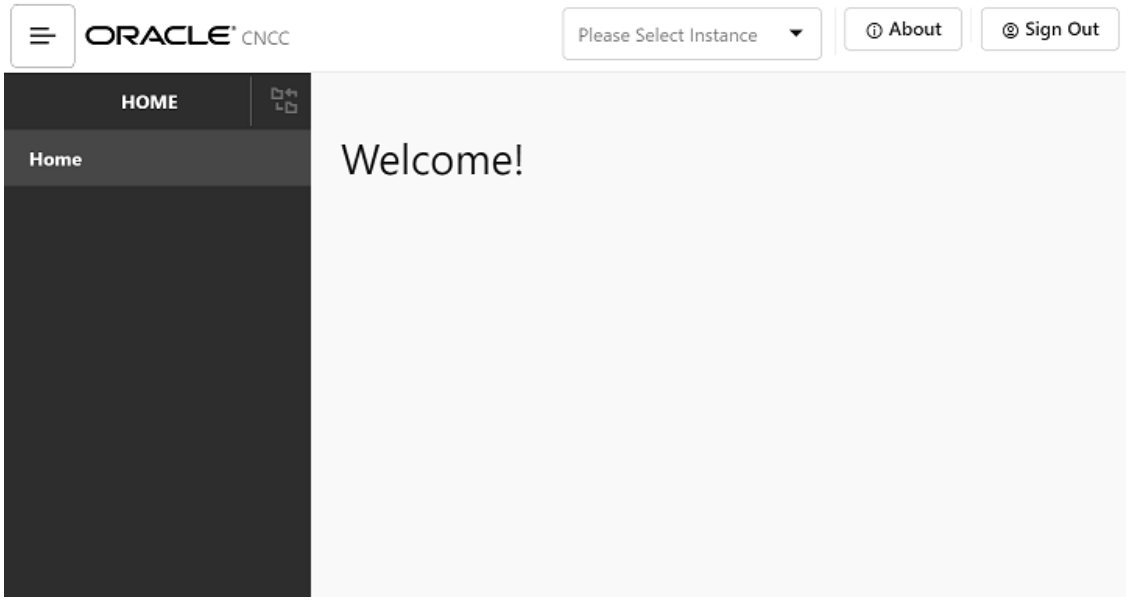
   > **Note**
   >
   > The IP Address mentioned above may change when the deployment cluster changes.

2. Save and close the hosts file.
   Before logging into CNC Console, create a CNCC user name and password. Log in to the CNC Console application using these login credentials. For information on creating a CNC Console user and password, see *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*.

CNC Console Log in

Following is the procedure to log into CNC Console:

1. Open any web browser.
2. Enter the URL: http://<host name>:<port number>.
   where, host name is cncc-iam-ingress-ip and port number is cncc-iam-ingressport.
3. Enter valid login credentials.
4. Click **Log in**. The CNC Console interface is displayed.



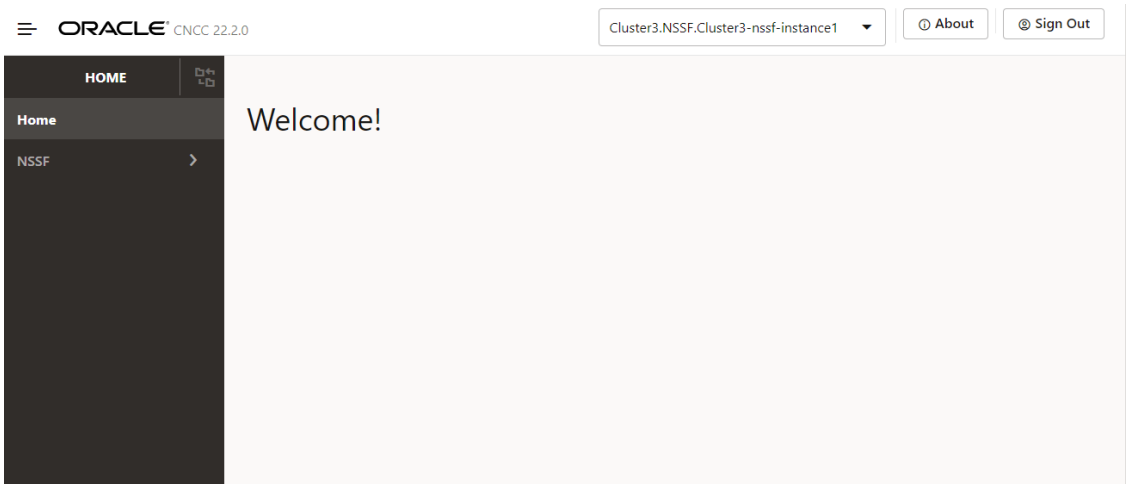
Select the required NF instance from the **Select Instance** drop-down list. The left pane displays the selected network function.

5.3 NSSF Configuration

This section describes how to configure different managed objects of NSSF using CNC Console.

On selecting NSSF instance from the drop-down list, the following screen appears:

Figure 5-1 NSSF Welcome Screen



5.3.1 NSSF System Option

Perform the following procedure to configure NSSF System Option:

1. From the left navigation menu, navigate to **NSSF** and click **NSSF System Option**.
2. The **NSSF System Option** page is displayed. This page shows the existing configurations, if any.
3. To add or update configurations, click **Edit** in the top-right corner. This opens the **Edit NSSF System Option** page.
4. On the **Edit NSSF System Option** page, configure the following fields:
   a. **AutoAuthorizeNssaiAvailabilityDataEnable**: Toggle to enable or disable automatic authorization of NSSAI availability data from the AMF.
   b. **EnhancedPatchBehaviourEnable**: Toggle to enable or disable the `allowZeroSnssaiInTai` setting. When enabled, the NSSF allows removing all supported S-NSSAIs in a TAI via `NssaiAvailabilityPatch` and PUT.
   c. **PlmnLevelSystemOptionsList**: Configure system options at the PLMN level for all supported PLMNs. A three-column table displays existing entries.
5. To add a new item to the **PlmnLevelSystemOptionsList**:
   a. Click **Add** to open the **Add PlmnLevelSystemOptionsList** pop-up.
   b. Fill in the following fields:
      i. **PLMN ID**: Enter the PLMN ID in the format MCC-MNC (e.g., 310-260, where 310 is MCC and 260 is MNC). Ensure the ID is validated against PLMN-level information. All configured PLMNs must be covered; otherwise, the system returns an **HTTP 400 Bad Request**.
      ii. **EnhancedAllowedNssaiEnable**: Toggle to enable or disable the enhancedAllowedNssaiEnable feature for this PLMN.
   c. Click **Save** to add the configuration or **Cancel** to discard changes and return.
6. To edit an existing **PlmnLevelSystemOptionsList** item:
   a. Click **Edit** next to the desired item, make changes, then click **Save**.

7. To delete an item:
    a. Click **Delete** next to the desired item.
    b. **Note**: At least one item must remain in the **PlmnLevelSystemOptionsList**. It cannot be empty.
8. After all updates:
    a. Click **Save** to apply changes to the NSSF System Option configuration.
    b. Click **Cancel** to discard changes and return to the previous page.
9. Click the **Refresh** button in the top-right corner to reload the latest configuration and reset any unsaved changes.

For more information on parameter values, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.3.2 NSI Profile

Perform the following procedure to configure NSI Profile::

1. From the left navigation menu, navigate to **NSSF** and click **NSI Profile**.
2. The **NSI Profile** page is displayed. This page shows the existing configurations in a tabular format, if any.
3. To add configurations, click **Add** in the top-right corner. This opens the **Add NSI Profile** page.
4. On the **Add NSI Profile** page, configure the following fields:
    a. **NsiId**: Enter the unique identifier (NSI ID) of the selected Network Slice instance.
    b. **NRF URI**: Enter the API URI of the NRF NFDiscovery Service. This URI is used to discover and select Network Functions (NFs) and services within the specified Network Slice instance.
    c. **NRF NF Management URI**: If applicable, enter the API URI of the NRF NFManagement Service. This is used for managing NF instances.
    d. **NRF Access Token URI**: If applicable, enter the API URI of the NRF Access Token Service. This is used to retrieve access tokens for authorized communication with the NRF.
5. Click **Save** to add the configuration or **Cancel** to discard changes and return to the NSI Profile page.
6. Click the **Refresh** button in the top-right corner to reload the latest configuration and reset any unsaved changes.

> **Note**
>
> - Use the **Edit**, **Delete**, and **View** icons available in the **Actions** column of the NSI Profile page to update, delete, or view any preconfigured information.
> - For more information on parameter values, refer to the *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.3.3 Supported Slices Mapping Config

> **Note**
>
> **Dependency**: **NsiProfile** and **SystemOptions** must be configured for **PlmnID** and **nsiInformationList** before configuring Supported Slices Mapping.

Perform the following procedure for Supported Slices Mapping Configuration:

1. From the left navigation menu, navigate to **NSSF** and click **Supported Slices Mapping Config**.
2. The **Supported Slices Mapping Config** page is displayed. This page shows the existing configurations in a tabular format, if any.
3. To add configurations, click **Add** in the top-right corner. This opens the **Add Supported Slices Mapping Config** page.
4. On the **Add Supported Slices Mapping Config** page, configure the following fields:
   a. **Name**: Enter a unique name for the mapping. This acts as the identifier for the slice-to-TAI mapping configuration.
   b. **supportedSnssaiList**: Click Add to open the **Add supportedSnssaiList** pop-up. **Note**: Each item must contain valid SNSSAI information, including SST and optionally SD.

> **Note**
>
> Each item must contain valid SNSSAI information, including **SST** and optionally **SD**

> **Note**
>
> **Validation rules:**
>
> - **NSI Profile Existence**: Each NSI profile referenced must exist in the NSI profiles table.
> - **SNSSAI Uniqueness**: Duplicate SNSSAIs are not allowed within the same entry.
> - **SNSSAI Configuration**: Each SNSSAI must exist in the ConfiguredNssai for the PLMN specified in the **plmnTacList**.
> - **Barred SNSSAI Check**: SNSSAIs in the **barredSnssaiList** for the same PLMN are not allowed.
> - **Configuration Conflict**: SNSSAIs must not be listed in **BarredSlicesMappingConfig** for the same TAI.

   i. **SST**: Enter Slice or Service Type.
   ii. **SD**: Slice Differentiator (6-digit hexadecimal).

iii. **AccessType**: Select from the drop-down list:
- 3GPP_ACCESS
- NON_3GPP_ACCESS
- BOTH

iv. **NsiInformationList**: Click **Add** to open the **Add NsiInformationList** pop-up. Configure the following fields:

i. **NsiProfileId**: Select the SliceId associated with the NSI (Network Slice Instance) Profile from the drop-down. This links the configuration to a specific NSI profile.

> **Note**
>
> **Note**: Must correspond to an existing entry in the NSI Profiles table.

ii. **Salience**: Enter a numeric priority value for NSI selection logic. Determines preference when multiple NSIs match a request.

> **Note**
>
> **Note**: Default value is 1 if left empty. Must be greater than 1.

iii. Click **Save** to add the configuration or **Cancel** to discard changes and return to the **Add supportedSnssaiList** pop-up.

iv. Use **Edit** or **Delete** icons to update or remove entries in the **NsiInformationList**.

v. Click **Save** in the **Add supportedSnssaiList** pop-up to add the configuration or **Cancel** to discard changes and return.

vi. Use **Edit** or **Delete** icons to update or remove entries in the **supportedSnssaiList**.

c. **Plmn Tac List**: Specify the list of Tracking Area Codes (TACs) associated with a PLMN. Each entry must include a valid PLMN ID (in MCC-MNC format) and one or more unique TACs. Configure the following fields:

i. **PlmnId**: Select from the drop-down list of predefined PLMN IDs. The PLMN IDs shown in this list are populated here from the **NSSF System Option** configuration.

ii. **Tac List**: Click **Add** to open the **Add Tac List** pop-up. Configure the following fields:

i. **TAC**: Tracking Area Code (4- or 6-digit value).
Notes:
- **TAC Uniqueness**: Must be unique within a single entry and across all entries.
- **Duplicate TACs**: Repeating TACs in the same entry or across multiple entries is invalid.
- **Unsupported PLMN**: Do not include PLMNs not listed in the supported configuration.

ii. Click **Save** to add the TAC or **Cancel** to discard changes and return.

iii. Use **Edit** or **Delete** icons to update or remove TAC entries.

d. Click **Save** on the **Add Supported Slices Mapping Config** page to add the configuration or **Cancel** to discard changes and return to the **Supported Slices Mapping Config** page.

5. Click the **Refresh** button in the top-right corner to reload the latest configuration and reset any unsaved changes.

> **Note**
>
> - Use the **Edit**, **Delete**, and **View** icons in the **Actions** column to update, delete, or view preconfigured information for Supported Slices Mapping.
> - For more information on parameter values, refer to the *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.3.4 PLMN Config

> **Note**
>
> **Dependency**: **NsiProfile** and **SystemOptions** must be configured for **PlmnID** and **nsiInformationList** before configuring PLMN Config.

Perform the following procedure for PLMN Configuration:

1. From the left navigation menu, navigate to **NSSF** and click **Plmn Config**.
2. The **Plmn Config** page is displayed. This page shows the existing configurations in a tabular format, if any.
3. To add configurations, click **Add** in the top-right corner. This opens the **Add Plmn Config** page.
4. On the **Add Plmn Config** page, configure the following fields:
   a. **PlmnId**: Select from the drop-down list of predefined PLMN IDs. The PLMN IDs are populated from the NSSF System Option configuration.
   b. **ConfiguredNssai**: Click **Add** to open the **Add ConfiguredNssai** pop-up.

      > **Note**
      >
      > Each item must contain valid NSSAI information, including **SST** and optionally **SD**

      i. **SST**: Enter Slice or Service Type.
      ii. **SD**: Slice Differentiator (6-digit hexadecimal).
      iii. **AccessType**: Select from the drop-down list:
         - 3GPP_ACCESS
         - NON_3GPP_ACCESS
         - BOTH

iv. **NsiInformationList**: Click **Add** to open the **Add NsiInformationList** pop-up. Configure the following fields:

i. **NsiProfileId**: Select the SliceId associated with the NSI Profile. Must correspond to an existing NSI Profile entry.

ii. **Salience**: Enter a numeric priority value for NSI selection logic. Input value must be greater than 1.

v. Click **Save** to add the configuration or **Cancel** to discard changes and return.

vi. Use **Edit** or **Delete** icons to update or remove entries in the **NsiInformationList**.

i. Click **Save** in the **Add ConfiguredNssai** pop-up to add the configuration or **Cancel** to discard changes.

ii. Use **Edit** or **Delete** icons to update or remove entries in **ConfiguredNssai**.

c. **NsiInformationList**: Click **Add** to open the **Add NsiInformationList** pop-up. Configure the following fields:

i. **NsiProfileId**: Select the SliceId associated with the NSI Profile.

ii. **Salience**: Enter a numeric priority value; must be greater than 1.

iii. Click **Save** to add the configuration or **Cancel** to discard changes.

iv. Use **Edit** or **Delete** icons to update or remove entries in **NsiInformationList**.

d. **BarredSnssaiList**: Click **Add** to open the **Add BarredSnssaiList** pop-up. Configure the following fields:

i. **SST**: Slice or Service Type.

ii. **SD**: Slice Differentiator (6-digit hexadecimal).

iii. Click **Save** to add the configuration or **Cancel** to discard changes.

iv. **Note**: Any S-NSSAI in this list must not appear in the allowed S-NSSAI list of any TAI within the same PLMN.

v. Use **Edit** or **Delete** icons to update or remove entries in **BarredSnssaiList**.

e. Click **Save** on the **Add Plmn Config** page to add the configuration or **Cancel** to discard changes and return to the **Plmn Config** page.

5. Click the **Refresh** button in the top-right corner to reload the latest configuration and reset any unsaved changes.

> **Note**
>
> - Use the **Edit**, **Delete**, and **View** icons in the **Actions** column to update, delete, or view preconfigured PLMN configuration information.
> - For more information on parameter values, refer to the *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.3.5 Barred Slices Mapping Config

> **Note**
>
> **Dependency**: **SystemOptions** and **Plmn Config** must be configured for **PlmnID** before configuring Barred Slices Mapping.

Perform the following procedure for Barred Slices Mapping Configuration:

1. From the left navigation menu, navigate to **NSSF** and click **Barred Slices Mapping Config**.
2. The **Barred Slices Mapping Config** page is displayed. This page shows the existing configurations in a tabular format, if any.
3. To add configurations, click **Add** in the top-right corner. This opens the **Add Barred Slices Mapping Config** page.
4. On the **Add Barred Slices Mapping Config** page, configure the following fields:
    a. **Name**: Enter a unique name for the mapping.
    b. **Barred Snssai List**:
        i. Click **Add** to open the **Add Barred Snssai List** pop-up.
        ii. Each item must contain valid SNSSAI information, including **SST** and optionally **SD**:
            i. **SST**: Enter Slice or Service Type.
            ii. **SD**: Slice Differentiator (6-digit hexadecimal).
        iii. Click **Save** to add the configuration or **Cancel** to discard changes and return to **Add Barred Slices Mapping Config** page.
        iv. Use **Edit** or **Delete** icons to update or remove entries in **Barred Snssai List**.
    c. **Plmn Tac List**:
        i. Specify the list of Tracking Area Codes (TACs) associated with a PLMN. Each entry must include a valid **PLMN ID** (MCC-MNC format) and one or more unique TACs.
        ii. Configure the following fields:
            i. **PlmnId**: Select from the drop-down list of predefined PLMN IDs. These are populated from the NSSF System Option configuration.
            ii. **Tac List**: Click **Add** to open the **Add Tac List** pop-up. Configure the following field:
                i. **TAC**: Tracking Area Code (4- or 6-digit value).

> **Note**
>
> - **TAC Uniqueness**: TACs must be unique within a single entry and across all entries.
> - **Duplicate TACs**: Repeating TACs in the same entry or across multiple entries is invalid.
> - **Unsupported PLMN**: Do not include PLMNs not listed in the supported configuration.

ii. Click **Save** to add the TAC or **Cancel** to discard changes and return.

iii. Use **Edit** or **Delete** icons to update or remove TAC entries.

5. Click **Save** on the **Add Barred Slices Mapping Config** page to add the configuration or **Cancel** to discard changes and return to the **Barred Slices Mapping Config** page.
6. Click the **Refresh** button in the top-right corner to reload the latest configuration and reset any unsaved changes.

> **Note**
>
> - Use the **Edit**, **Delete**, and **View** icons in the **Actions** column to update, delete, or view preconfigured Barred Slices Mapping information.
> - For more information on parameter values, refer to the *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.3.6 Georedundant Sites

Perform the following procedure to configure Georedundant Sites:

1. From the left navigation menu, navigate to **NSSF** and click **Georedundant Sites**.
2. The **Georedundant Sites** page is displayed. This page shows the existing configurations in a tabular format, if any.
3. To add configurations, click **Add** in the top-right corner. This opens the **Add Georedundant Sites** page.
4. On the **Add Georedundant Sites** page, configure the following fields:
   a. **Site ID**: Enter a unique ID for the NSSF site.
   b. **NF Instance ID**: Enter the instance ID of the NSSF site in UUID format.
   c. **Rank**: Specify the priority to be given to this georedundant site.
   d. Click **Save** to add the configuration or **Cancel** to discard changes and return to the **Georedundant Sites** page.
5. Click the **Refresh** button in the top-right corner to reload the latest configuration and reset any unsaved changes.

> **Note**
>
> - Use the **Edit**, **Delete**, and **View** icons in the **Actions** column to update, delete, or view preconfigured Georedundant Sites information.
> - For more information on parameter values, refer to the *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.3.7 Logging Level Options

Perform the following procedure to configure Logging Level Options:

1. From the left navigation menu, navigate to **NSSF**.
2. Select **NSSF** and click **Logging Level Options**.
   The **Logging Level Options** page is displayed.

   This page displays a list of preconfigured log levels with the following details:
   - Service
   - Application Log Level
3. Click **View** on the right most column of a log level from the list to see the preconfigured log level details in a pop-up window named **View Log Level List**.
4. Click **X** icon to close **View Log Level List** pop-up window.
5. Click **Edit** from the top right side to edit **Logging Level Options** parameters.
6. Configure **Logging Level Options** fields as described in the following table:

Table 5-1 Logging Level Options Parameters

Field Name	Description
Service Type	Select the service type you want to configure from the drop-down list with the following options: • App-info • perf-info • alternate route • nsavailability • nsselection • nsaudit • nsconfig • nssubscription • egw • igw • nrf-client-nfmanagement
Application Log Level	Select log level for the application from the drop-down list with the following options: • DEBUG • ERROR • INFO • TRACE • WARN • FATAL

Table 5-1 (Cont.) Logging Level Options Parameters

Field Name	Description
Additional Error Logging	Select Additional Error Logging option from the drop-down list with the following options: • ENABLED • DISABLED **Note** Applicable only when **Service Type** is selected as **nrf-client-nfdiscovery** or **nrf-client-management**.
Log Subscriber Info	Select Log Subscriber Info option from the drop-down list with the following options: • ENABLED • DISABLED **Note** Applicable only when **Service Type** is selected as **nrf-client-nfdiscovery** or **nrf-client-management**.
Package Log Level	This is a list of packages with corresponding log level applicable to the selected Service Type.

7. Click **Edit** under **Package Log Level** to edit the Package Log Level parameters for the selected Service Type. The **Edit Package Log Level** pop-up window appears.
8. Enter the values for **Edit Package Log Level** parameters as described in the following table:

Table 5-2 Package Log Level Parameters

Field Name	Description
Package	This field should not be edited. It is preconfigured based on the selected Service Type. **Caution** Do not edit it.
Log Level	Select log level for the package from the drop-down list with the following options: • DEBUG • ERROR • INFO • TRACE • WARN • FATAL

9. Click **Save** to save or **Cancel** to discard your progress in the **Edit Package Log Level** pop-up window.
10. **Log Discarding**: **<Conditional>** applicable only when **Service Type** is selected as **egw** or **igw**. Configure the following fields in **Log Discarding** form:
    a. **Enabled**: Use the toggle to enable or disable Log Discarding.
    b. **Feature To Threshold Mapping:** Use **Add** icon to add new Feature To Threshold Mapping, or use **Edit** icon to edit the configuration of existing Feature To Threshold Mapping. Configure the following fields in **Add Feature To Threshold Mapping** or **Edit Feature To Threshold Mapping** form:
        i. **Feature Name**: Enter the name of the feature. Available options are:
            - RATE_LIMITING
            - OVERLOAD_CONTROL
            - ROUTE_LEVEL_RATE_LIMITING
            - EGRESS_RATE_LIMITING
        ii. **Threshold Factor**: Enter threshold factor for the feature between 0-100.
        iii. Click **Save** to save or **Cancel** to discard your progress on the **Add Feature To Threshold Mapping** or **Edit Feature To Threshold Mapping** form.
11. Click **Save** to save or **Cancel** to discard your progress on the **Edit Logging Level Options** page.

For more information on parameter values, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide.*

5.3.8 NSSF Restore

This API provides the functionality to restore an existing configuration restored as backup.

1. From the left navigation menu, navigate to **NSSF**.
2. Select **NSSF** and click **NSSF Restore**.
3. The **NSSF Restore** page is displayed.
4. Click **Edit** from the top-right corner.
5. A tabbed interface for **Response** and **Request** body appears.
    a. Click on the **Request** tab to configure request body parameters in JSON format.
    b. Configure the request body using the parameters described in the *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.
6. Click **Submit** to send the request with the configured request body parameters.
7. Click the **Response** tab to view the response body of the request.

> **Note**
>
> 1. Use the **Edit** option to update an existing configuration.
> 2. Use the **Export** option to download the response data as a JSON file.
> 3. Use the **Clear** option to clear the **Request** and **Response** panes.
> 4. For more information on parameter values, refer to the *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.3.9 NSSF Backup

This API provides the functionality to backup an existing configuration.

Configuring NSSF Backup

Perform the following procedure to configure NSSF Backup:

1. From the left navigation menu, navigate to **NSSF**.
2. Select **NSSF** and click **NSSF Backup**.
   The **NSSF Backup** page is displayed.
3. Click **Get** from the top right side.
   A panel interface for **Response** appears, which contains the response body of the sent Get request.

For more information on REST API, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide.*

> **Note**
>
> - Use **Export** option to download the response data as a JSON file.
> - Use **Clear** option to clear the Request and Response panes.

5.4 Common Services Configuration

Use this section to navigate to respective sections of **Egress Gateway** and **Ingress Gateway** configurations.

5.4.1 Egress Gateway

This section contains the Egress Gateway APIs.

5.4.1.1 Peer Configuration

This URI is used to add or update the list of peers wherein each peer consists of ID, host, port or virtualHost, and apiPrefix. The ID of each peer is mapped to Peer Identifier in Peer Set Configuration. The default value is null.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Egress Gateway** option to configure the Egress Gateway APIs.
3. Click the **Peer Configuration** option under **Egress Gateway** to configure peers.
   The **Peer Configuration** page is displayed.
4. Click **Add** to add the peer configuration.
   The **Create Peer Configuration** page is displayed.
5. Configure the following fields in the **Create Peer Configuration** page:
   a. **ID**: Enter a unique Peer identifier.
   b. **Host**: Enter the Host details of a local peer. It can be IPv4, IPv6, and FQDN details.
   c. **Port**: Enter the port details of the local host peer.
   d. **API Prefix**: Enter the API prefix details of a peer.
   Note: It is recommended to set the value as /.
   e. **Virtual Host**: Enter the Host details of a remote peer. This FQDN is sent to an alternate route service.
   f. **healthApiPath**: Parameter to support SCP health check API. It contains path of the health API.

   > **Note**
   >
   > The value of this parameter should be configured to align with the SCP configuration.

6. Click **Save** on the **Create Peer Configuration** page to save the details. Click **Cancel** to discard your progress and go back to **Peer Configuration** page.

> **Note**
>
> - Use **Edit** icon available in the next column of the specific entry to update configured the Peer Configuration information.
> - Use **Refresh** icon to refresh the list of peers configured.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.1.2 Peer Set Configuration

This URI is used to add or update the list of peer sets wherein each peer set consists of id and list of http/https instances. Each instance consists of priority and peer identifier that is mapped to id in peerconfiguration resource. The id of each peer set is mapped to peerSetIdentifier in routesconfiguration resource. The default value is null.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.

2. Click the **Egress Gateway** option to configure the Egress Gateway APIs.
3. Click the **Peer Set Configuration** option under **Egress Gateway** to configure peers. The **Peer Set Configuration** page is displayed.
4. Click **Add** to add the peer set configuration. The **Create Peer Set Configuration** page is displayed.
5. Configure the following fields in the **Create Peer Configuration** page:
   a. **ID**: Enter a unique Peer set identifier.
   b. Click **Add** in **HTTP Configuration** section to add HTTP Configuration. The **Add HTTP Configuration** dialog box is displayed.
      i. Enter the following information on this page:
         i. **Priority**: Enter the Priority of peer to be used in a peer set.
         ii. **Peer Identifier**: Enter the Peer identifier is the value of peer configured during PeerConfiguration.
      ii. Click **Save** to save HTTP Configuration. Click **Cancel** to discard your progress, close the dialog box, and go back to **Create Peer Set Configuration** page.
   c. Click **Add** in **HTTPS Configuration** section to add HTTPs Configuration. The **Add HTTPS Configuration** page is displayed.
      i. Enter the following information on this page:
         i. **Priority**: Enter the Priority of peer to be used in a peer set.
         ii. **Peer Identifier**: Enter the Peer identifier is the value of peer configured during Peer Configuration.
      ii. Click **Save** to save HTTP Configuration. Click **Cancel** to discard your progress, close the dialog box, and go back to **Create Peer Set Configuration** page.
6. Click **Save** on the **Create Peer Set Configuration** page to save the details. Click **Cancel** to discard your progress and go back to **Peer Set Configuration** page.

> **Note**
> - Use **Edit** icon available in the next column of the specific entry to update configured the Peer Set Configuration information.
> - Use **Refresh** icon to refresh the list of peer sets configured.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.1.3 Peer Monitoring Configuration

This URI is used to update the peer configuration with healthApiPath.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Egress Gateway** option to configure the Egress Gateway APIs.

3. Click the **Peer Monitoring Configuration** option under **Egress Gateway** to configure peers.
   The **Peer Monitoring Configuration** page is displayed with default configured values.
4. Click **Edit** to update the peer monitoring configuration.
   The **Edit Peer Monitoring Configuration** page is displayed.
5. Configure the following fields in the **Edit Peer Monitoring Configuration** page:
   a. **Enabled**: Use the switch to enable or disable peer monitoring feature.
   b. **Timeout**: Attribute to configure the duration of time after which calls to the SCP health API is timed out.
   c. **Frequency**: Indicates the frequency or recurring interval at which Egress Gateway initiates health check calls toward SCP.
   d. **FailureThreshold**: Indicates the number of failure responses after which a healthy SCP can be marked as unhealthy.
   e. **SuccessThreshold**: It indicates the number of successful responses after which an unhealthy SCP can be marked as healthy.
6. Click **Save** on the **Edit Peer Monitoring Configuration** page to save the details. Click **Cancel** to discard your progress and go back to **Peer Monitoring Configuration** page.

> **Note**
> - Use **Refresh** icon to refresh the peer monitoring configuration.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.1.4 Routes Configuration

This URI is used to add or update list of routes.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Egress Gateway** option to configure the Egress Gateway APIs.
3. Click the **Routes Configuration** option under **Egress Gateway** to configure peers.
   The **Routes Configuration** page is displayed.
4. Click **Add** to add the peer configuration.
   The **Create Routes Configuration** page is displayed.
5. Configure the following fields in the **Create Routes Configuration** page:
   a. **ID**: Enter a unique route configuration identifier.
   b. **URI**: Provide any dummy URL, or leave the existing URL with existing value.
   c. **Order**: Provide the order of the execution of this route.
   d. Configure the following fields in the **metadata** section:
      i. **httpsTargetOnly**: Enable it to select SBI instances for https list only (if 3gpp sbi target root header is http). Keep it disabled to select as per provided scheme.

> **Note**
>
> SBI Routing feature will not work if this switch is disabled.

ii. **httpRuriOnly**: This switch indicates the scheme of the outgoing request from NSSF. If it is enabled, the scheme of RURI is changed to http. If it is disabled, no change occurs to the scheme.

> **Note**
>
> SBI Routing feature will not work if this switch is disabled.

iii. **sbiRoutingEnabled**: Switch to enable or disable SBI Routing feature.

e. **predicates**: Click **Add** in the right side column to add predicates.
Add predicates window is displayed.

f. Configure the following fields in the **Add predicates** window:

i. **pattern**: Enter pattern details.

ii. **Name**: Enter name of the predicate.

g. Click **Save** to save predicates configuration. Click **Cancel** to discard your progress, close the window, and go back to **Create Routes Configuration** page.

h. **Filters**: Click **Add** in the right side column to add filters.
Add Filters window is displayed.

i. Configure the following fields in the **Add Filters** window:

i. **PeerSetIdentifier**: Enter PeerSetIdentifier for the filter.

ii. **customPeerSelectorEnabled**: Use this switch to enable or disable Custom Peer Selector.

iii. **errorHandling**: Click **Add** in the right side column to add errorHandling scenarios.
Add errorHandling window is displayed.

i. Configure the following fields in the **Add errorHandling** window:

i. **errorCriteriaSet**: Enter errorCriteriaSet.

ii. **actionSet**: Enter actionSet.

iii. **priority**: Enter priority.

ii. Click **Save** to save errorHandling configuration. Click **Cancel** to discard your progress, close the window, and go back to **Add Filters** window.

j. Click **Add** at the bottom of the window to save filters configuration.

k. Click **Remove** to remove filter configuration.
Add Filters window is displayed with reset fields.

l. Click **Save** in **Add Filters** window to save your progress (Add Filter or Remove Filter). Click **Cancel** to discard your progress, close the window, and go back to **Create Routes Configuration** page.

6. Click **Save** on the **Create Routes Configuration** page to save the details. Click **Cancel** to discard your progress and go back to **Routes Configuration** page.

> **Note**
>
> - Use **Edit** icon available in the next column of the specific entry to update configured information.
> - Use **Refresh** icon to refresh the routes configuration.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.1.5 SBI Error Action Sets

This URI is used to list or update SBI error action sets configuration at Egress gateway. By default this configuration is disabled.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Egress Gateway** option to configure the Egress Gateway APIs.
3. Click the **SBI Error Action Sets** option under **Egress Gateway** to configure peers.
   The **SBI Error Action Sets** page is displayed.
4. Click **Add** to add an error action set configuration.
   The **Create SBI Error Action Sets** page is displayed.
5. Configure the following fields in the **Create SBI Error Action Sets** page:
   a. **ID**: Enter an unique ID for SBI routing error action set.
   b. **Action**: Indicate the action that needs to be taken when specific criteria set is matched.
   c. **Attempts**: Enter the maximum number of retries to either same or different peer in case of error or failures from backend.
   d. **Block List**: Perform the following configuration:
      i. **Enabled**: Use the switch to disable or enable the peer bloacking feature using the server headers received in the response.
      ii. **Duration**: Enter the duration for which the peer is blocked and no traffic is routed to that peer for this period.
6. Click **Save** on the **Create Routes Configuration** page to save the details. Click **Cancel** to discard your progress and go back to **Create Routes Configuration** page.

Note:

- Use **Edit** icon available in the next column of the specific entry to update the **Error Action Sets** information.
- Use **Refresh** icon to refresh the configuration.

5.4.1.6 SBI Error Criteria Sets

This URI is used to list or update SBI error criteria sets configuration at Egress Gateway. By default, this configuration is disabled.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Egress Gateway** option to configure the Egress Gateway APIs.
3. Click the **SBI Error Criteria Sets** option under **Egress Gateway** to configure peers.
   The **SBI Error Criteria Sets** page is displayed.
4. Click **Add** to add an error action set configuration.
   The **Create SBI Error Criteria Sets** page is displayed.
5. Configure the following fields in the **Create SBI Error Criteria Sets** page:
   a. **ID**: Enter an unique ID for SBI routing error action set.
   b. **Method**: Indicate the type of methods for which the re-route need to be attempted.
   c. **Exceptions**: Enter the specific exceptions for which reroute or retry will be triggered.
   d. **Response**: Configure the following fields under **Statuses** section. Click **Add** to add HTTP status details:
      i. **Status Series**: Enter the HTTP status series for which reroute or retry is triggered, when the error response is received from downstream.
      ii. **Status**: Specify HTTP statuses that belongs to above mentioned status series for which reroute or retry is triggered. To enable retry or reroute for all the HTTP status belonging to a status series, configure this as -1.
      iii. Click **Save** in **Add Statuses** window to save statuses configuration. Click **Cancel** to discard your progress, close the window, and go back to **Create SBI Error Criteria Sets** page.
6. Click **Save** on the **Create SBI Error Criteria Sets** page to save the details. Click **Cancel** to discard your progress and go back to **Create SBI Error Criteria Sets** page.

Note:

- Use **Edit** icon available in the next column of the specific entry to update the **SBI Error Criteria Sets** information.
- Use **Refresh** icon to refresh the configuration.

For more information about the configuration parameters, see *Oracle Communication Cloud Native Core, Network Repository Function REST Specification Guide.*

5.4.1.7 User Agent Header Generation

This URI is used to Enable or Disable User-Agent Header.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Egress Gateway** option to configure the Egress Gateway APIs.
3. Click the **User Agent Header Generation** option under **Egress Gateway** to configure peers.
   The **User Agent Header Generation** page is displayed with default configured values.
4. Click **Edit** to update the User Agent Header Generation.
   The **Edit User Agent Header Generation** page is displayed.
5. Configure the following fields in the **Edit User Agent Header Generation** page:

a. **Enabled**: Use the switch to enable or disable User Agent Header feature.

b. **NF Type:** : Attribute to configure the nfType that is used to generate the User Agent Header. In this case, it is NSSF.

c. **NF Instance ID:** : Indicates the UUID (Instance ID) of the NSSF deployment used to generate the User Agent Header.

d. **NF FQDN:** : This is an optional parameter, if operators want to include the FQDN string configured under this section then the parameter Add Fqdn To Header needs to be enabled.

e. **Add Fqdn To Header:** : Use the allow or deny User Agent from appending the NSSF FQDN information while generating the User Agent Header.

f. **Overwrite Header**: Use this switch to govern if you want to include the User Agent Header generated at NSSF Egress Gateway or forward the User Agent received from service request.

> **Note**
>
> - When User Agent Header is enabled but the header information is missing, then it is picked from the OAuthClient module.
> - If the User Agent Header is present in the request towards AMF or NRF, then the value present in the header is overwritten or forwarded based on the **Overwrite Header** switch. If this switch is enabled, then the header is overwritten.

6. Click **Save** on the **Edit User Agent Header Generation** page to save the details. Click **Cancel** to discard your progress and go back to **User Agent Header Generation** page.

> **Note**
>
> - Use **Refresh** icon to refresh the User Agent Header Generation.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.2 Ingress Gateway Configuration

This section contains the Ingress Gateway APIs.

5.4.2.1 Error Code Profiles

This URI can be used to update the errorCodeProfiles that is used in Overload Control feature for populating details in error responses when a request is discarded. By default, the errorCodeProfiles remains null.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Ingress Gateway** option to configure the Ingress Gateway APIs.

3. Click the **Error Code Profiles** option under **Ingress Gateway**.
   The **Error Code Profiles** page is displayed.
4. Click **Add** to add the profiles.
   The **Add Error Code Profiles** page is displayed.
5. Configure the following fields in the **Add Error Code Profiles** page:
   a. **Name**: Enter the name for the error profile. This name is also used while create Create Overload Control Discard Policies .
   b. **Error Code**: Enter the HttpStatusCode. This field in the errorScenario determines the HttpStatusCode that needs to be populated in ProblemDetails (HttpStatus field) response from Ingress Gateway when the exception occurred at Ingress Gateway matches the configured errorScenario's exceptionType field.
   c. **Error Cause**: Enter the error cause details. This field in the errorScenario determines the error cause that needs to be populated in ProblemDetails (Cause field) response from Ingress Gateway when the exception occurred at Ingress Gateway matches the configured errorScenario's exceptionType parameter.
   d. **Error Title**: Enter the error title. This field in the errorScenario determines the title that needs to be populated in ProblemDetails (Title field) response from Ingress Gateway when the exception occurred at Ingress Gateway matches the configured errorScenario's exceptionType parameter.
   e. **Redirect URL**: Enter the redirection URL. This value is populated in LOCATION header while sending response from Ingress Gateway. The header is populated only when the exception occurred at Ingress Gateway matches the configured errorScenario's exceptionType parameter, the errorCode configured for the particular errorScenario lies in 3xx error series and the redirectUrl field for the particular errorScenario is configured appropriately.
   f. **Retry After**: Enter the value in seconds or particular date after which the service should be retried, this value is populated in Retry-After header while sending response from Ingress Gateway.
   g. **Error Description**: Enter the description that needs to be populated in ProblemDetails (Detail field) response from Ingress Gateway when the exception occurred at Ingress Gateway matches the configured errorScenario's exceptionType field.
6. Click **Save** on the **Add Error Code Profiles** page to save the details. Click **Cancel** to discard your progress and go back to **Add Error Code Profiles** page.

> **Note**
>
> - Use **Edit** icon available in the next column of the specific entry to update the **Error Code Profiles** information.
> - Use **Refresh** icon to refresh the configuration.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.2.2 Create Overload Control Discard Policies

This URI can be used to update discard policies that will be used in overload control to select the appropriate policy from the configured list based on the load level of a particular service. By default, ocDicardPolicies is null.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Ingress Gateway** option to configure the Ingress Gateway APIs.
3. Click the **Overload Control Discard Policies** option under **Ingress Gateway**. The **Overload Control Discard Policies** page is displayed.
4. Click **Add** to add the configuration.
   The **Create Overload Control Discard Policies** page is displayed.
5. Configure the following fields in the **Create Overload Control Discard Policies** page:
   a. **Name**: Enter the name of the discarded policy.
   b. **Scheme**: Enter the discarded policy scheme based on percentage.
   c. Click **Add** in the **Policies** section.

      The **Add Policies** page is displayed.
   d. Configure the following fields under the **Add Policies** page:
      i. **Value**: Enter the value of priority above which requests are considered as potential candidates for drop. It is the percentage of requests to drop in the current sampling period over the calculated rate in the previous sampling period.
      ii. **Action**: Enter the action to be taken on selected requests rejection based on error code. For example, RejectWithErrorCode.
      iii. **Level**: Enter the overload level.
      iv. **Error Code Profile**: Enter the name of the error code profile created in Error Code Profiles.
      v. Click **Save** on the **Add Policies** page to save the details. Click **Cancel** to discard your progress and go back to **Add Policies** page.
6. Click **Save** on the **Create Overload Control Discard Policies** page to save the details. Click **Cancel** to discard your progress and go back to **Create Overload Control Discard Policies** page.

> **Note**
>
> - Use **Edit** or **Delete** icon available in the next column of the specific entry to update or delete the **Policies** information.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.2.3 Discard Policy Mapping

This URI can be used to update service names and corresponding policy names for the service which is mapped to "ocDiscardPolicies" based on "policyName" and also to enable or disable the Overload Control feature and the sampling period in overload control. By default, the Overload Control feature is disabled and the sampling period is 6000.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Ingress Gateway** option to configure the Ingress Gateway APIs.
3. Click the **Discard Policy Mapping** option under **Ingress Gateway** to configure peers. The **Discard Policy Mapping** page is displayed with default configured values.
4. Click **Edit** to update the Discard Policy Mapping. The **Edit Discard Policy Mapping** page is displayed.
5. Configure the following fields in the **Edit Discard Policy Mapping** page:
   a. **Enabled**: Use the switch to enable or disable discard policy mapping.
   b. **Mappings**: Configure the following fields in **Mappings** section. Click **Add** to open **Add Mappings** window and add mapping details:

      > **Note**
      >
      > A value for **Mappings** is required when **Enabled** is switched on. If **Mappings** is empty, the Overload Control feature will behave as if it is disabled.

      - **Service Name**: Enter the service name. This field is used to determine a mapping between service and discard policy name per service name. It must be added in the following format:
        <deployment-name>-<servicename>

        > **Note**
        >
        > – If a value for **Policy Name** is provided, then **Service Name** is mandatory.
        > – servicename is fixed and cannot be changed.

      - **Policy Name**: Enter the policy name. It determines a mapping between the service and discards policy name per service.

        > **Note**
        >
        > If a value for **Service Name** is provided, then **Policy Name** is mandatory.

   c. Click **Save** on the **Add Mappings** window to save the details. Click **Cancel** to discard your progress and go back to **Edit Discard Policy Mapping** page.

> **Note**
>
> - Use **Edit** icon to edit an existing configuration.
> - Use **Delete** icon to remove an existing configuration.

6. **Sampling Period**: Add sampling period. It is the time frame for each cycle of Overload Control per service. Its value is in milliseconds.
7. Click **Save** on the **Edit Discard Policy Mapping** page to save the details. Click **Cancel** to discard your progress and go back to **Discard Policy Mapping** page.

> **Note**
>
> - Use **Refresh** icon to refresh the peer monitoring configuration.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.2.4 Error Code Series

This URI can be used to update the errorcodeserieslist that are used in Overload Control feature and Server Header feature to list the configurable exception or error for an error scenario in Ingress Gateway.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Ingress Gateway** option to configure the Ingress Gateway APIs.
3. Click the **Error Code Series** option under **Ingress Gateway**.
   The **Error Code Series**page is displayed.
4. Click **Edit** to add the code series.
   The **Edit Error Code Series** page is displayed.
5. Configure the following fields in the **Edit Error Code Series**page:
   a. **ID**: Enter an unique ID for error code.
   b. **Exception List**: Lists the configurable exception or error for an error scenario in Ingress Gateway. The supported values are: ConnectionTimeout, RequestTimeout, UnknownHostException, ConnectException, RejectedExecutionException, InternalError, NotFoundException, ClosedChannelException, and BlackListIpException
6. Click **Add** in the **Error Code Series** section.
   The **Add Error Code Series** page is displayed.
7. Configure the following fields in the **Add Error Code Series** page:
   a. **Error Set**: Enter the possible values include all error codes in the respective HttpSeries value assigned for "errorSet".
   Note: Use single value of "-1" if all error codes in that HttpSeries are to be considered.
   b. **Error Codes**: Enter the possible values for "errorSet" attribute: 5xx, 4xx, 3xx, 2xx, 1xx.

c. Click **Save** on the **Add Error Code Series** page to save the details. Click **Cancel** to discard your progress and go back to **Add Error Code Series** page.

8. Click **Save** on the **Edit Error Code Series** page to save the details. Click **Cancel** to discard your progress and go back to **Edit Error Code Series** page.

> **Note**
>
> - Use **Edit** icon available in the next column of the specific entry to update the **Error Code Series** information.
> - Use **Refresh** icon to refresh the configuration.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.2.5 Routes Configuration

The configuration of "routesconfiguration" is required for Server Header and Overload control feature to map route ID and its corresponding route-level configuration. By default, this configuration is null.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Ingress Gateway** option to configure the Ingress Gateway APIs.
3. Click the **Routes Configuration** option under **Ingress Gateway**.
   The **Routes Configuration** page is displayed.
4. Click **Add** to add the configuration.
   The **Create Routes Configuration** page is displayed.
5. Configure the following fields in the **Create Routes Configuration** page.
   a. **ID**: Value of "id" attribute defines a specific service for route configuration. It specifies the route IDs for which you need to define server header.

      > **Note**
      >
      > Ensure that route ID used in REST configuration is same as the route ID used in values.yaml file

   b. **serverHeaderDetails**: Configure the following fields in **serverHeaderDetails** section:
      i. **Enabled**: Use the switch to enable or disable server header at route level.
      ii. **Error Code Series Id:** : Specify the error list ID.

      > **Note**
      >
      > Ensure that an errorCodeSeries exists corresponding to the errorCodeSeriesId.

6. Click **Save** to save Routes configuration. Click **Cancel** to discard your progress, close the window, and go back to **Create Routes Configuration** page.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.2.6 OAuth Validator Configurations

This REST API configuration is required for enabling access token validation using NRF Instance ID and key-ID (K-ID).

Before this configuration, perform the prerequisite steps and helm configuration explained in OAuth Access Token Based Authorization.

After Helm configuration, send the REST requests to use configured public key certificates. Using REST-based configuration, you can distinguish between the certificates configured on different NRFs and can use these certificates to validate the token received from a specific NRF.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Ingress Gateway** option to configure the Ingress Gateway APIs.
3. Click the **OAuth Validator Configurations** option under **Ingress Gateway**.
   The **OAuth Validator Configurations** page is displayed.
4. Click **Edit** to add the OAuth Validator Configurations.
   The **Edit OAuth Validator Configurations** page is displayed.
5. Configure the following fields in the **Edit OAuth Validator Configurations** page:
   a. **Key ID List**: Click **Add** in the **Key ID List** section and configure the following fields in **Add Key ID List** window:
      - **Key ID**: Enter the Key-ID.
      - **Kubernetes Secret Name**: Enter Kubernetes Secret Name.
      - **Kubernetes Secret Key For Certificate**: Enter Kubernetes Secret Key for the certificate.
      - **Access Token Generation Algorithm**: Enter the Access Token Generation Algorithm.
   b. Click **Save** on the **Add Key ID List** page to save the details. Click **Cancel** to discard your progress, close the window, and go back to **Edit OAuth Validator Configurations** page.
   c. **Instance ID List**: Click **Add** in the **Instance ID List** section and configure the following fields in **Add Instance ID List** window:
      - **Instance ID**: Enter the NRF Instance ID.
      - **Kubernetes Secret Name**: Enter the Kubernetes Secret Name.
      - **Kubernetes Secret Key For Certificate**: Enter Kubernetes Secret Key for the certificate.
      - **Access Token Generation Algorithm**: Enter the Access Token Generation Algorithm.

6. Click **Save** on the **Add Instance ID List** page to save the details. Click **Cancel** to discard your progress, close the window, and go back to **Edit OAuth Validator Configurations** page.
7. **Access Token Validation Mode**: Enter the mode of validation, which are INSTANCEID_ONLY, KID_ONLY, or KID_PREFERRED. It will check for keyIdList or instanceIdList for validation of token received based on mode selected.
8. Click **Save** on the **Edit OAuth Validator Configurations** page to save the details. Click **Cancel** to discard your progress and go back to **Edit OAuth Validator Configurations** page.

> **Note**
>
> - Use **Edit** icon available in the next column of the specific entry to update the **Key ID List** or **Instance ID List**.
> - Use **Refresh** icon to refresh the configuration.

For more information on recommended parameter values, range, default values, and whether they are mandatory, optional, or conditional, see *Oracle Communications Cloud Native Core, Network Slice Selection Function REST Specification Guide*.

5.4.2.7 Server Header Details

This API can be used for adding Server Header in the error responses sent from Ingress Gateway. By default, this feature is disabled. To enable the feature, invoke the following REST API and update the enable switch.

Perform the following configurations:

1. From the left navigation menu, navigate to **NSSF** and click the **Common Configuration** option.
2. Click the **Ingress Gateway** option to configure the Ingress Gateway APIs.
3. Click the **Server Header Details** option under **Ingress Gateway** to configure peers. The **Server Header Details** page is displayed with default configured values.
4. Click **Edit** to update the Server Header Details. The **Edit Server Header Details** page is displayed.
5. Configure the following fields in the **Edit Server Header Details** page:
   a. **Enabled**: Use the switch to enable or disable Server Header.
   b. **Error Code Series Id**: Specify the error list ID.

      > **Note**
      >
      > Ensure that an errorCodeSeries exists corresponding to the errorCodeSeriesId.

   c. **Configuration**: Configure the following fields in **Configuration** section:
      - **NF Type**: Specify the type of network function. In this case, it is NSSF.
      - **NF Instance Id:**: Enter the NSSF instance ID. It represents the UUID of the NSSF deployment that is used to generate the Server Header.

6. Click **Save** on the **Edit Server Header Details** page to save the details. Click **Cancel** to discard your progress and go back to **Server Header Details** page.

5.5 cnDBTier APIs

1. From the left navigation menu, navigate to **NSSF** and then click **cnDBTier** tab.
   The **cnDBTier** page is displayed.
2. Click the **Backup List** to view the list of completed backups along with backup ID, backup size, and backup creation timestamp.
   The **Backup List** page is displayed.

   > **Note**
   >
   > The following APIs are read-only.

   Table 5-3 Backup List

Fields	Description
Site Name	This attribute displays the name of the current site to which NSSF is connected.
Backup Details	This attribute displays the following information like backup id, backup size, and backup creation timestamp.
Backup Id	This attribute displays the ID of the stored backup.
Backup Size (bytes)	This attribute displays the size of the stored backup.
Creation TimeStamp	This attribute displays the time recorded when the backup was stored.

3. Click **cnDBTier Backup Status** to view the backup status details.
   The **cnDBTier Backup Status** page is displayed.
   - **Last Backup Run Schedule**: Shows the time stamp of last backup attempt.
   - **Last Backup Run Status**: Shows status if the last backup attempt was completed or failed.
   - **Next Backup Run Schedule**: Shows the time stamp of next scheduled backup attempt.
4. Click **cnDBTier Health** to view the health status of the microservices like replication, backup manager, monitor services, and NDB services.
   The **cnDBTier Health** page is displayed.
   - Click the **Backup Manager Health Status** to view the health status of the backup manager.
     The **Backup Manager Health Status** page is displayed.

     > **Note**
     >
     > The following APIs are read-only.

Table 5-4 Backup Manager Health Status

Fields	Description
Service Name	This attribute displays the service name of the backup manager microservice.
Service Status	This attribute displays the service status of the backup manager microservice. Possible values are UP and DOWN.
DB Connection Status	This attribute displays the database connection status of the backup manager microservice. Possible values are UP and DOWN.
Overall Backup Manager Service Health	This attribute displays the overall health status of the backup manager microservice. Possible values are UP and DOWN.
Backup Executor Health Status	This attribute displays the following information like node id and DB connection status of the backup executor.
Node Id	This attribute displays the id of the node.
DB Connection Status	This attribute displays the backup executor database connection status with the nodes. Possible values are UP and DOWN.

- Click the **Monitor Health Status** to view the health status of the services.
  The **Monitor Health Status** page is displayed.

> **Note**
>
> The following APIs are read-only.

Table 5-5 Monitor Health Status details

Attribute	Description
Service Name	This attribute displays the service name of the monitor microservice.
DB Connection Status	This attribute displays the database connection status of the monitor microservice. Possible values are UP and DOWN.
Metric Scrape Status	This attribute displays the status of the metric scrape, that is if the metrics are fetched or not. If the metrics are fetched then the service is up and vice versa. Possible values are UP and DOWN.
Overall Monitor Service Health	This attribute displays the overall health status of the monitor microservice. Possible values are UP and DOWN.

- Click the **NDB Health Status** to view the health status of the network database.
  The **NDB Health Status** page is displayed.

> **Note**
>
> The following APIs are read-only.

Table 5-6 NDB Health Status details

Attribute	Description
Local Site Name	This attribute displays the name of the current site. For example, site 1, site 2.
NDB Health Status Details	This attribute displays the health status of the network database like name of the NDB service, status of the service, health status of PVC.
Service Name	This attribute displays the service name. For example, ndbmgmd-0, ndbmtd-0, ndbmyappsqld-1, ndbmysqld-2.
Service Status	This attribute displays the status of the service. Possible values are UP and DOWN.

- Click the **Replication Health Status** to view the health status of the replication sites. The **Replication Health Status** page is displayed.

> **Note**
>
> The following APIs are read-only.

Table 5-7 Replication Health Status details

Attribute	Description
Local Site Name	This attribute displays the name of the current site (site 1, site 2).
Health Status Details	This attribute displays the health status details of the local site like replication service name, replication service status, database connection status of the replication service, and the overall health status of the replication micorservices. The number of rows in this table varies depending on the type of deployment (for example, two-site, three-site deployments).
Service Name	This attribute displays the name of the available replication service.
Service Status	This attribute displays the status of the available replication service. Possible values are UP and DOWN.
DB Connection Status	This attribute displays the database connection status of the replication microservice. Possible values are UP and DOWN.

Table 5-7 (Cont.) Replication Health Status details

Attribute	Description
Overall Replication Service Health	This attribute displays the overall health status of the replication microservice. Possible values are UP and DOWN.

5. Click **cnDBTier Version** to view the version.
The **cnDBTier Version** page is displayed.

> **Note**
>
> The following APIs are read-only.

Table 5-8 cnDBTier Version Attributes

Attribute	Description
cnDBTier Version	This attribute displays the cnDBTier version.
NDB Version	This attribute displays the network database (NDB) version.

6. Click the **Database Statistics Report** to view the available databases.
The **Database Statistics Report** page is displayed.

> **Note**
>
> The following APIs are read-only.

Table 5-9 Database Statistics Report

Fields	Description
Database Count	This attribute displays the number of available database.
Database Tables Count	This attribute displays the available database names and their table count.
Database Name	This attribute displays the database name.
Table Count	This attribute displays the table count for each database.
Database Table Rows Count	This attribute displays the table rows present in each table.
Database Name	This attribute displays the database name.

a. Click on the **View** icon available next to the database name to view the **View Database Table Rows Count** screen.
The **View Database Table Rows Count** page is displayed.

Table 5-10 View Database Table Rows Count

Fields	Description
Database Name	This attribute displays the database name.
Tables	This attribute displays the table names and the corresponding rows in each table.
Table Name	This attribute displays the table name.
Row Count	This attribute displays the table rows present in each table.

7. Click **Georeplication Recovery** to access the Georeplication Recovery Status of the cnDBTier cluster. This includes options such as Update Cluster As Failed, Start Georeplication Recovery, and Georeplication Recovery Status.
Georeplication Recovery page is displayed.
   a. Click **Update Cluster As Failed** to mark the cnDBTier cluster as FAILED.
   The **Update Cluster As Failed** page is displayed.

Table 5-11 Update Cluster As Failed

Attribute	Description
Cluster Name	This field displays a list of cnDBTier clusters that can be marked as failed.
Failed Cluster Name	This field displays a cnDBTier cluster that is marked as failed.

      i. **Update Cluster**: Click to update the cluster details.
      ii. **Cancel**: Click to discard progress and go back to **Georeplication Recovery** page.

   b. Click **Start Georeplication Recovery** to start the georeplication recovery process for a failed site.
   The **Start Georeplication Recovery** page is displayed.

Table 5-12 Start Georeplication Recovery

Attribute	Description
Failed Cluster Name	This field displays a list of all the cnDBTier clusters that have been marked as failed.
Backup Cluster Name (Optional)	This field displays a list of all the healthy cnDBTier clusters. If no cnDBTier cluster is selected, the system uses the first available healthy cnDBTier cluster for the backup.

      i. **Start Georeplication Recovery**: Click to start georeplication recovery.
      ii. **Cancel**: Click to discard progress and go back to **Georeplication Recovery** page.

   c. Click **Georeplication Recovery Status** to view the status of georeplication recovery for cnDBTier clusters.
   The **Georeplication Recovery Status** page is displayed.

Table 5-13 Georeplication Recovery Status

Attribute	Description
Local Cluster Name	This field displays the name of the local cnDBTier cluster.
Georeplication Recover Status Details	This field displays the details of the georeplication recovery status of cnDBTier clusters.
Cluster Name	This field displays the cnDBTier clusters by name.
Georeplication Recovery Status	This field displays the current georeplication recovery status of the corresponding cnDBTier cluster. For more information about the georeplication recovery status, see *Oracle Communications Cloud Native Core, cnDBTier User Guide*.

i. **Refresh**: Click to refresh georeplication recovery status.

8. Click **Georeplication Status** to view the local site and remote site name to which NSSF is connected.
The **Georeplication Status** page is displayed.

> **Note**
>
> The following APIs are read-only.

Table 5-14 GeoReplication Status

Attribute	Description
Local Site Name	This attribute displays the local site name to which NSSF is connected. **Note:** The number of local site names may vary depending on the type of georeplication used in NSSF.
Remote Site Name	This attribute displays the remote site name. **Note:** The number of remote site names may vary depending on the type of georeplication used in NSSF.
Replication Status	This attribute displays the replication status with corresponding sites. **Note:** The number of replication statuses may vary depending on the type of georeplication used in NSSF.
Seconds Behind Remote Site	This attribute displays the number of seconds that the last record read by the local site is behind the latest record written by the remote site for all the replication groups. **Note:** The number of replication statuses may vary depending on the type of georeplication used in NSSF.

a. Click on the **View** icon in the **Actions** menu, to view the **View Georeplication Status** screen.
The **Georeplication Status** page is displayed.

Table 5-15 Georeplication Status

Attribute	Description
Replication Group Delay	This attribute displays the seconds behind the remote site for individual replication groups.
Replication Channel Group Id	This attribute displays the ID of the replication channel group.

b. Click on the **View** icon to view the **Replication Group Delay** attributes.
The **Replication Group Delay** page is displayed.

Table 5-16 View Replication Group Delay

Attribute	Description
Channel Details	This attribute displays the channel details such as Remote Replication IP and Role.
Remote Replication IP	This attribute displays the IP of the remote replication channel.
Role	This attribute displays the role of the replication channel IP.

9. Click **Georeplication Status Across All Sites** to access the georeplication status across all configured sites in the cnDBTier cluster.
The **Georeplication Status Across All Sites** page is displayed.

Table 5-17 Georeplication Status Across All Sites

Fields	Description
Site Name	This field displays the primary site for which the georeplication status is being reported.
Remote Site Name	This field displays the remote site involved in replication with the local site.
Status	This field displays the georeplication status.
Actions	Click the icon in actions column to view the View Georeplication Status Across All Sites attributes.

a. **Actions**: Click the icon in actions column to view the View Georeplication Status Across All Sites attributes.
The **View Georeplication Status Across All Sites** page is displayed.

Table 5-18 View Replication Details Per Group ID

Attribute	Description
Replication Group ID	This field displays the unique identifier assigned to the replication group.
Replication Channel Group Status	This field displays the replication status for the specific replication group.
All SQL Status Details	This field displays the detailed SQL and replication state information for all SQL nodes for the specified replication group.
Action	Click on the icon in action column to view the View All SQL Status Details attributes.

i. **Action**: Click on the icon in action column to view the View All SQL Status Details attributes.
The **View All SQL Status Details** page is displayed.

Attribute	Description
SQL Node	This field displays the SQL node whose replication metrics are being reported.
Source Host	This field displays the remote host from which the SQL node pulls replication data.
External IP	This field displays the external endpoint through which the SQL node is accessible within the site.
Replica I/O Running Status	This field displays whether the replication I/O thread is currently active or not. Possible values: Yes or No.
Replica SQL Running Status	This field displays whether the SQL thread responsible for applying replicated changes is running or not. Possible values: Yes or No.
Server ID	This field displays the unique server identifier used in the replication topology.
Role	This field displays the role of the SQL node within the replication setup (for example, ACTIVE or STANDBY).
Last Error Code	This field displays the most recent SQL thread error code encountered during replication.
Last Error Message	This field displays the most recent error message reported by the SQL thread.
Last I/O Error Code	This field displays the most recent I/O thread error code encountered during replication.
Last I/O Error Message	This field displays the most recent error message reported by the I/O thread.
Last SQL Error Code	This field displays the most recent SQL execution error code encountered while applying replicated transactions.
Last SQL Error Message	This field displays the last SQL execution error message reported.
SQL Remaining Delay	This field displays the replication lag or delay (if any) between the source site and this SQL node. Unit: seconds.

10. Click **Local Cluster Status** to view the local cluster status for the current site.
The **Local Cluster Status** page is displayed.

> **Note**
>
> The following APIs are read-only.

Table 5-19 Local Cluster Status

Attribute	Description
Site Name	This attribute displays the name of the current site to which NSSF is connected.
Cluster Status	This attribute displays the local cluster status for the current site.

11. Click the **On Demand Backup** to create a new backup and view the status of initiated on-demand backups.
The **On Demand Backup** page is displayed.

> **Note**
>
> The following APIs are read-only.

Table 5-20 On Demand Backup Details

Fields	Description
Site Name	This attribute displays the name of the current site to which NSSF is connected.
DR Status	This attribute displays the disaster recovery status .
Backup Id	This attribute displays the ID of the stored backup.
Backup Status	This attribute displays the status of backup.
Remote Transfer Status	The attribute displays the status of remote transfer.
Initiate Backup	The attribute displays whether the backup is initiated or not. **Note:**You can read and write this API.

- Click **Edit**. The **Edit** On Demand Backup page appears.

> **Note**
>
> The **Edit** mode is available only for Initiate Backup.

- Use the Toggle option to Initiate the backup and click **Save**. A confirmation message "Save successfully" appears.
- Click **Cancel** to navigate back to the On Demand Backup page.
- Click **Refresh** to reload the On Demand Backup page.

12. Click the **Replication HeartBeat Status** to view the connectivity between local site and remote site name to which NSSF is connected.
The **Replication HeartBeat Status** page is displayed.

> **Note**
>
> The following APIs are read-only.

Table 5-21 Replication HeartBeat Status

Fields	Description
Site Name	This attribute displays the name of the current site to which NSSF is connected.

Table 5-21 (Cont.) Replication HeartBeat Status

Fields	Description
HeartBeat Details	This attribute displays the following information like remote site name, heart beat status, heart beat lag, and replication channel group id.
Remote Site Name	This attribute displays the remote site name.
Heartbeat Status	This attribute displays the connectivity status with corresponding sites.
Heartbeat Lag	This attribute displays the lag or latency in seconds it took to synchronize between sites.
Replication channel Group Id	This attribute displays the ID of the replication channel group.

6
NSSF Metrics, KPIs, and Alerts

This chapter includes information about Metrics, KPIs, and Alerts for Oracle Communications Cloud Native Core, Network Slice Selection Function.

> **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.

6.1 NSSF Metrics

This section includes information about dimensions, common attributes, and metrics for NSSF.

Metric Types

The following table describes the NSSF metric types used to measure the health and performance of NSSF and its core functionalities:

Table 6-1 Metric Type

Metric Type	Description
Counter	Represents the total number of occurrences of an event or traffic, such as measuring the total amount of traffic received and transmitted by NSSF, and so on.
Gauge	Represents a single numerical value that changes randomly. This metric type is used to measure various parameters, such as SCP load values, memory usage, and so on.

Table 6-1 (Cont.) Metric Type

Metric Type	Description Suffix
Histogram	Represents samples of observations (such as request durations or response sizes) and counts them in configurable buckets. It also provides a sum of all observed values. _max _bucket _count _sum

Latency Metrics Format Change for NSSF Microservices

With the migration of Spring boot to Micronaut, the support for latency metric `*_latency_seconds` has been deprecated. The below metrics `*_latency_seconds_[suffix]` continue to be supported and can be used in lieu of `*_latency_seconds`

- `*_latency_seconds_max`
- `*_latency_seconds_bucket`
- `*_latency_seconds_count`
- `*_latency_seconds_sum`

This update applies to the metrics of all NSSF associated microservices, as well as Ingress and Egress Gateway microservices.

> **Note**
>
> Support for the metric `*_latency_seconds` continues to be present only in Ingress and Egress Gateway.

Dimensions

The following table describes different types of metric dimensions:

Table 6-2 Dimensions

Dimension	Description	Values
AMF Instance Id	NF-Id of AMF	NA
authority	Used in Gateway metrics. Indicates the destination address.	NA
BackendSvc	Used in Gateway metrics. Indicates the address of destination.	NA
BackendSvcAddressType	Used in Gateway metrics. Indicates the IP type (IPv4/IPv6) of the destination from the Egress Gateway.	IPv4, IPv6
CauseCode	It specifies the cause code of an error response.	Cause Code of the error response. For example, "SUBSCRIPTION_NOT_FOUND"
certificateName	Determines the certificate name inside a specific secret that is configured via persistent configuration when oauth is enabled.	NA
ClientCertIdentity	Cerificate Identity of the client.	SAN=127.0.0.1,localhost CN=localhost, N/A if data is not available
ConfigurationType	Determines the type of configuration in place for OAuth Client in Egress Gateway. If nrfClientQueryEnabled Helm parameter in oauthClient Helm configurations at Egress Gateway is false then the ConfigurationType is STATIC, else DYNAMIC.	STATIC, DYNAMIC
configVersion	Indicates the configuration version that Ingress or gateway is currently maintaining.	Value received from config server (1, 2...)
ConsumerNFInstanceId	NF instance id of the NF service consumer.	NA
ConsumerNFType	The NF type of the NF service consumer.	NRF, UDM, AMF, SMF, AUSF, NEF, PCF, SMSF, NSSF, UDR, LMF, GMLC,5G_EIR, SEPP, UPF, N3IWF, AF, UDSF, BSF, CHF, NWDAF
DestinationHost	Used in Gateway metrics. Indicates the destination IP address or FQDN of the host.	NA
destinationHostAddressType	Used in Gateway metrics. Indicates the destination IP type (IPv4 or IPv6) from Egress Gateway.	IPv4, IPv6
Direction	Indicates the direction of connection established, that is, whether it is incoming or outgoing.	ingress, egressOut
dnsResolvedType	Used in Gateway metrics. Indicates the actual DNS resolved IP type (IPv4 or IPv6) of the destination.	IPv4, IPv6

Table 6-2 (Cont.) Dimensions

Dimension	Description	Values
duration_type	Used in NSSF cache metrics to denote the type of duration measured by the timer metric.	total_cache_processing_duration, cache_fetch_all_records_db_query_duration, cache_stale_fetch_db_query_duration, populate_tainssaimap_cache_duration
egressRoutingMode	Used in Gateway metrics. Indicates the value of the egressRoutingMode configured in Egress Gateway.	IPv4, IPv6, IPv4_IPv6, IPv6_IPv4, None
error_reason	Indicates the reason for failure response received. If message is sent in the response, then it is filled with the message otherwise exception class is filled. In case of successful response it is filled with "no-error".	• "no_error" (In case successful response is received) • "java.nio.channels.ClosedChannelException" • "unable to find valid certification path to requested target" • "SSL handshake failed due to invalid SNI"
ErrorOriginator	Captures the ErrorOriginator.	ServiceProducer, Nrf, IngresGW, None
ERRORTYPE	Determines the type of error.	DB_ERROR/MISSING_CONFIGURATION/ UNKNOWN
Host	Specifies IP or FQDN port of ingress gateway.	NA
HttpVersion	Specifies Http protocol version.	HTTP/1.1, HTTP/2.0
id	Determines the keyid or instance id that is configured via persistent configuration when oauth is enabled.	NA
InstanceIdentifier	Prefix of the pod configured in helm when there are multiple instances in same deployment.	Prefix configured in helm, UNKNOWN
issuer	NF instance ID of NRF	NA
managed_object	The managed object of NSSF	plmnconfig, supportedslicesmapping, barredslicesmapping, nsiprofiles, systemoptions, grsites, deleteConfiguration, allconfig, unsupported_managed_object, configuredNssai, supportedplmnlist
Message Type	This specifies the type of NS-Selection query message.	INITIAL_REGISTRATION/PDU_SESSION/ UE_CONFIG_
Message Type	This specifies the type of NS-Selection query message.	INITIAL_REGISTRATION/PDU_SESSION/ UE_CONFIG_UPDATE
Method	HTTP method	POST/PUT/PATCH/DELETE/GET/OPTIONS
NegotiatedTLSVersion	This denotes the TLS version used for communication between the server and the client.	TLSv1.2, TLSv1.3.
NFServiceType	Name of the Service within the NF.	For Eg: Path is /nxxx-yyy/vz/..... Where nxxx-yyy is NFServiceType UNKNOWN if unable to extract NFServiceType from the path.
NFType	It specifies the name of the NF Type.	For example: Path is /nxxx-yyy/vz/..... Where XXX(Upper Case) is NFType UNKNOWN if unable to extract NFType from the path.
NrfUri	URI of the Network Repository Function Instance.	For example: nrf-stubserver.ocnssf-site:8080

Table 6-2 (Cont.) Dimensions

Dimension	Description	Values
number_of_records	The number of records fetched or updated.	Integer values
Operation	NSAvailability Operation	UPDATE/DELETE/SUBSCRIBE/UNSUBSCRIBE
Port	Port number	Integer values
quantile	Captures the latency values with ranges as 10ms, 20ms, 40ms, 80ms, 100ms, 200ms, 500ms, 1000ms and 5000ms.	Integer values
query_type	Type of DB read query	applypolicy_reg/applypolicy_pdu/evaluate_amfset/ evaluate_resolution
reason	The reason contains the human readable message for oauth validation failure.	NA
receivedAddressType	Used in Gateway metrics. Indicates the IP type (IPv4/IPv6) of the remote client connected to the Ingress Gateway.	IPv4, IPv6
releaseVersion	Indicates the current release version of Ingress or Egress gateway.	Picked from helm chart {{ .Chart.Version }}
ResponseCode	HTTP response code.	Bad Request, Internal Server Error etc. (HttpStatus.*)
retryCount	The attempt number to send a notification.	Depends on the helm parameter httpMaxRetries (1, 2...)
Route_Path	Path predicate or Header predicate that matches the current request.	NA
Scheme	Specifies the Http protocol scheme.	HTTP, HTTPS, UNKNOWN
scope	NF service name(s) of the NF service producer(s), separated by whitespaces.	NA
secretName	Determines the secret name that is configured via persistent configuration when oauth is enabled	NA
serialNumber	Indicates the type of the certificate.	serialNumber=4661 is used for RSA and serialNumber =4662 is used for ECDSA
Source	Determines if the configuration is done by the operator or fetched from AMF.	OperatorConfig/LearnedConfigAMF
Status	HTTP response code	NA
StatusCode	Status code of NRF access token request.	Bad Request, Internal Server Error etc. (HttpStatus.*)
status_code	Status Code of any response	All supported HTTP status code like 2xx, 4xx, 5xx, etc.
subject	NF instance ID of service consumer	NA

Table 6-2 (Cont.) Dimensions

Dimension	Description	Values
Subscription_removed	The dimension indicates the status of a subscription upon receiving a 404 response from the AMF after a notification is sent.	"**false**": The subscription was not deleted. This value applies if the feature is disabled, indicating no deletion attempt was made. "**true**": The subscription was successfully deleted. This value applies if the feature is enabled and the deletion process completed successfully. "**error**": The subscription was not deleted due to internal issues, such as a database error, despite the feature being enabled and a deletion attempt being made.
Subscription- Id	Subscription -ID	NA
TargetNFInstanceId	NF instance ID of the NF service producer	NA
TargetNFType	The NF type of the NF service producer.	NRF, UDM, AMF, SMF, AUSF, NEF, PCF, SMSF, NSSF, UDR, LMF, GMLC,5G_EIR, SEPP, UPF, N3IWF, AF, UDSF, BSF, CHF, NWDAF
type	For NSSF Cache Metrics, the type denotes the type of cache operation done	cache_create, cache_update, cache_refresh
updated	Indicates whether the configuration is updated or not.	True, False
VirtualFqdn	FQDN that shall be used by the alternate service for the DNS lookup	Valid FQDN

Common Attributes

The following table includes information about common attributes for NSSF.

Table 6-3 Common Attributes

Attribute	Description
application	The name of the application that the microservice is a part of.
eng_version	The engineering version of the application.
microservice	The name of the microservice.
namespace	The namespace in which microservice is running.
node	The name of the worker node that the microservice is running on.

6.1.1 NSSF Success Metrics

This section provides details about the NSSF success metrics.

Table 6-4 nssf_subscription_to_nrf_successful

Field	Details
Description	Indicates if subscription to NRF for nfType NSSF is successful in case of GR enabled NSSF setup.

Table 6-4 (Cont.) nssf_subscription_to_nrf_successful

Field	Details
Type	Gauge **Note** value is 1 when subscription is successful and 0 if it fails and retry for subscription.
Service Operation	NSConfig
Dimension	retryCount

Table 6-5 ocnssf_nsselection_rx_total

Field	Details
Description	Count of request messages received by NSSF for the Nnssf_NSSelection service.
Type	Counter
Service Operation	NSSelection
Dimension	• AMF Instance Id • Message Type • plmn

Table 6-6 ocnssf_nsselection_success_tx_total

Field	Details
Description	Count of success response messages sent by NSSF for requests for the Nnssf_NSSelection service.
Type	Counter
Service Operation	NSSelection
Dimension	• AMF Instance Id • Message Type • plmn

Table 6-7 ocnssf_nssaiavailability_rx_total

Field	Details
Description	Count of request messages received by NSSF for the Nnssf_NSSAIAvailability service.
Type	Counter
Service Operation	NSAvailability
Dimension	• NfId • Method • Message Type

Table 6-8 ocnssf_nssaiavailability_success_tx_total

Field	Details
Description	Count of success response messages sent by NSSF for requests for the Nnssf_NSSAIAvailability service.
Type	Counter
Service Operation	NSAvailability
Dimension	• NfId • Method • Message Type

Table 6-9 ocnssf_nssaiavailability_options_rx_total

Field	Details
Description	Count of HTTP options received at NSAvailability service.
Type	Counter
Service Operation	NSAvailability
Dimension	• Method • Message Type

Table 6-10 ocnssf_nssaiavailability_options_tx_status_ok_total

Field	Details
Description	Count of HTTP options response with status 200 OK.
Type	Counter
Service Operation	NSAvailability
Dimension	• Method • Message Type

Table 6-11 ocnssf_nssaiavailability_notification_indirect_communication_rx_total

Field	Details
Description	Count of request notification messages sent by NSSF using indirect communication.
Type	Counter
Service Operation	NSAvailability
Dimension	• Method • Message Type

Table 6-12 ocnssf_nssaiavailability_notification_indirect_communication_tx_total

Field	Details
Description	Count of notification response messages received by NSSF using indirect communication.
Type	Counter
Service Operation	NSAvailability
Dimension	• Method • Message Type

Table 6-13 ocnssf_nssaiavailability_notification_success_response_rx_total

Field	Details
Description	Count of success notification response messages received by NSSF for requests for the Nnssf_NSSAIAvailability service.
Type	Counter
Service Operation	NSSubscription
Dimension	• Method • Message Type

Table 6-14 ocnssf_nssaiavailability_notification_tx_total

Field	Details
Description	Count of notification messages sent by NSSF as part of Nnssf_NSSAIAvailability service.
Type	Counter
Service Operation	NSSubscription
Dimension	• Method • Message Type

Table 6-15 ocnssf_notification_trigger_rx_total

Field	Details
Description	Count of notification triggers received by NSSF.
Type	Counter
Service Operation	NSSubscription
Dimension	• Trigger type

Table 6-16 ocnssf_notification_trigger_tx_success_total

Field	Details
Description	Count of success notification trigger responses sent by NsSubscription.
Type	Counter
Service Operation	NSSubscription
Dimension	• Trigger type • returnCode

Table 6-17 ocnssf_indirect_communication_request_rx_total

Field	Details
Description	Count of subscription creation requests received from AMF when indirect communication was enabled.
Type	Counter
Service Operation	NSSubscription
Dimension	• method • bindingHeaderPresent

Table 6-18 ocnssf_indirect_communication_response_tx_success_total

Field	Details
Description	Count of subscription creation responses sent by NSSF when indirect communication was enabled.
Type	Counter
Service Operation	NSSubscription
Dimension	• method • returnCode • bindingHeaderPresent

Table 6-19 ocnssf_subscription_request_rx_total

Field	Details
Description	Count of subscription creation requests received from AMF
Type	Counter
Service Operation	NSSubscription
Dimension	• method

Table 6-20 ocnssf_subscription_response_tx_success_total

Field	Details
Description	Count of subscription creation responses sent by NSSF.
Type	Counter
Service Operation	NSSubscription
Dimension	• method • returnCode

Table 6-21 stale_records_computation_DQD_amf_tai_slice_availability_data_seconds_count

Field	Details
Description	This metric captures the amount of time taken by availability fetch query for computing stale records. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-22 stale_records_computation_DQD_amf_tai_slice_availability_data_seconds_max

Field	Details
Description	This metric shows the longest (maximum) observed duration in seconds. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-23 stale_records_computation_DQD_amf_tai_slice_availability_data_seconds_sum

Field	Details
Description	This metric is the sum of all observed durations. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-24 stale_records_computation_JL_amf_tai_slice_availability_data_seconds_count

Field	Details
Description	This metric captures the amount of time taken by code execution to compute stale records to be deleted. It starts after fetch availability query.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-25 stale_records_computation_JL_amf_tai_slice_availability_data_seconds_max

Field	Details
Description	This metric shows the longest (maximum) observed duration in seconds. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-26 stale_records_computation_JL_amf_tai_slice_availability_data_seconds_sum

Field	Details
Description	This metric is the sum of all observed durations. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-27 total_stale_records_APD_amf_tai_slice_availability_data_seconds_count

Field	Details
Description	This metric captures the total number of observations or events recorded for the timer. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-28 total_stale_records_APD_amf_tai_slice_availability_data_seconds_max

Field	Details
Description	This metric shows the longest (maximum) observed duration in seconds. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-29 total_stale_records_APD_amf_tai_slice_availability_data_seconds_sum

Field	Details
Description	This metric is the sum of all observed durations. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-30 count_amf_tai_slice_availability_data_stale_records_deleted_total

Field	Details
Description	This is the counter metrics for the deleted records. It is being pegged when application delete the stale records.
Type	Counter
Service Operation	NsAuditor
Dimension	NA

Table 6-31 stale_records_computation_DQD_nssai_subscriptions_seconds_count

Field	Details
Description	This metric captures the total number of observations or events recorded for the timer for subscription fetch query for computing stale records. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-32 stale_records_computation_DQD_nssai_subscriptions_seconds_max

Field	Details
Description	This metric shows the longest (maximum) observed duration in seconds for subscription fetch query for computing stale records. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-33 stale_records_computation_DQD_nssai_subscriptions_seconds_sum

Field	Details
Description	This metric is the sum of all observed durations. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-34 stale_records_computation_JL_nssai_subscriptions_seconds_count

Field	Details
Description	This metric captures the total number of observations or events recorded for the timer for code execution to compute subscription stale records to be deleted. It starts after fetch subscription query.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-35 stale_records_computation_JL_nssai_subscriptions_seconds_max

Field	Details
Description	This metric shows the longest (maximum) observed duration in seconds for code execution to compute subscription stale records to be deleted. It starts after fetch subscription query.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-36 stale_records_computation_JL_nssai_subscriptions_seconds_sum

Field	Details
Description	This metric is the sum of all observed durations for code execution to compute subscription stale records to be deleted. It starts after fetch subscription query.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-37 total_stale_records_APD_nssai_subscriptions_seconds_count

Field	Details
Description	This metric captures the total number of observations or events recorded for the timer for whole process including fetch subscription query, code execution and delete stale records query execution time. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-38 total_stale_records_APD_nssai_subscriptions_seconds_max

Field	Details
Description	This metric shows the longest (maximum) observed duration in seconds for whole process including fetch subscription query, code execution and delete stale records query execution time. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-39 total_stale_records_APD_nssai_subscriptions_seconds_sum

Field	Details
Description	This metric is the sum of all observed durations for whole process including fetch subscription query, code execution and delete stale records query execution time. It starts as soon as the scheduler starts the task.
Type	Timer
Service Operation	NsAuditor
Dimension	NA

Table 6-40 count_nssai_subscriptions_stale_records_deleted_total

Field	Details
Description	This is the counter metrics for the subscription deleted records. It is being pegged when application delete the subscription stale records.
Type	Counter
Service Operation	NsAuditor
Dimension	NA

Table 6-41 ocnssf_nsselection_latency_seconds_count

Field	Details
Description	This metric captures the total number of observations or events recorded for the timer.
Type	Histogram
Service Operation	NsSelection
Dimension	NA

Table 6-42 ocnssf_nsselection_latency_seconds_max

Field	Details
Description	This metric shows the longest (maximum) observed duration in seconds.
Type	Histogram
Service Operation	NsSelection
Dimension	NA

Table 6-43 ocnssf_nsselection_latency_seconds_sum

Field	Details
Description	This metric is the sum of all observed durations.
Type	Histogram
Service Operation	NsSelection
Dimension	NA

Table 6-44 ocnssf_nsselection_latency_seconds_bucket

Field	Details
Description	This metric records the count of events (or observations) whose value falls below or equal to a specific duration threshold (bucket).
Type	Histogram
Service Operation	NsSelection
Dimension	NA

Table 6-45 ocnssf_nsavailability_latency_seconds_count

Field	Details
Description	This metric captures the total number of observations or events recorded for the timer.
Type	Histogram
Service Operation	NsAvailability
Dimension	NA

Table 6-46 ocnssf_nsavailability_latency_seconds_max

Field	Details
Description	This metric shows the longest (maximum) observed duration in seconds.
Type	Histogram
Service Operation	NsAvailability
Dimension	NA

Table 6-47 ocnssf_nsavailability_latency_seconds_sum

Field	Details
Description	This metric is the sum of all observed durations.
Type	Histogram
Service Operation	NsAvailability
Dimension	NA

Table 6-48 ocnssf_nsavailability_latency_seconds_bucket

Field	Details
Description	This metric records the count of events (or observations) whose value falls below or equal to a specific duration threshold (bucket).

Table 6-48 (Cont.) ocnssf_nsavailability_latency_seconds_bucket

Field	Details
Type	Histogram
Service Operation	NsAvailability
Dimension	NA

6.1.2 NSSF Error Metrics

This section provides details about the NSSF error metrics.

Table 6-49 ocnssf_configuration_database_read_error_total

Field	Details
Description	Count of errors encountered when trying to read the configuration database.
Type	Counter
Service Operation	NSSelection
Dimension	None

Table 6-50 ocnssf_state_data_write_error_total

Field	Details
Description	Count of errors encountered when trying to write to the state database.
Type	Counter
Service Operation	NSAvailability
Dimension	None

Table 6-51 ocnssf_nsselection_unsupported_plmn_total

Field	Details
Description	Count of request messages that did not find mcc and mnc in the PLMN list.
Type	Counter
Service Operation	NSSelection
Dimension	None

Table 6-52 ocnssf_nssaiavailability_notification_error_response_rx_total

Field	Details
Description	Count of failure notification response messages received by NSSF for requests by the Nnssf_NSSAIAvailability service.
Type	Counter
Service Operation	NSSubscription

Table 6-52 (Cont.) ocnssf_nssaiavailability_notification_error_response_rx_total

Field	Details
Dimension	• MessageType • Method • ResponseCode • CauseCode • retryCount

Table 6-53 ocnssf_nsavailability_unsupported_plmn_total

Field	Details
Description	Count of request messages with unsupported PLMN received by NSSF for the ocnssf_NSAvailability service.
Type	Counter
Service Operation	NSAvailability
Dimension	• Message Type • Method

Table 6-54 ocnssf_nsavailability_invalid_location_url_total

Field	Details
Description	Count of invalid location header.
Type	Counter
Service Operation	NSAvailability
Dimension	• Message Type • Method

Table 6-55 ocnssf_nssaiavailability_submod_error_response_tx_total

Field	Details
Description	Count of error response messages sent by NSSF for HTTP patch for subscription (SUBMOD) requests for ocnssf_NSSAIAvailability service.
Type	Counter
Service Operation	NSAvailability
Dimension	• ReturnCode • SubscriptionId • Method

Table 6-56 ocnssf_nssaiavailability_submod_unimplemented_op_total

Field	Details
Description	Count of HTTP patch request messages received by NSSF for ocnssf_NSSAIAvailability service for which PATCH operation (op) is not implemented.
Type	Counter
Service Operation	NSAvailability

Table 6-56 (Cont.) ocnssf_nssaiavailability_submod_unimplemented_op_total

Field	Details
Dimension	• ReturnCode • SubscriptionId • Method

Table 6-57 ocnssf_nssaiavailability_submod_patch_apply_error_total

Field	Details
Description	Count of HTTP patch request messages received by OCNSSFfor ocnssf_NSSAIAvailability service for which PATCH application returned error.
Type	Counter
Service Operation	NSAvailability
Dimension	• ReturnCode • SubscriptionId • Method

Table 6-58 ocnssf_nssaiavailability_notification_delete_on_subscription_not_found_total

Field	Details
Description	Triggered when 404 Subscription with SUBSCRIPTION_NOT_FOUND is received by AMF.
Type	Counter
Service Operation	NsSubscription
Dimension	Subscription_Removed

Table 6-59 ocnssf_nssaiavailability_notification_db_error_total

Field	Details
Description	Triggered when DB error or exception occurs when trying to delete NssaiSubscription.
Type	Counter
Service Operation	NsSubscription
Dimension	None

Table 6-60 ocnssf_nssaiavailability_indirect_communication_subscription_failure_total

Field	Details
Description	Count of failure when subscription messages sent by NSSF using indirect communication.
Type	Counter
Service Operation	NSAvailability
Dimension	• Message Type • Method

Table 6-61 ocnssf_nssaiavailability_indirect_communication_notification_failure_total

Field	Details
Description	Count of failure when notification messages sent by NSSF using indirect communication.
Type	Counter
Service Operation	NSSubscription
Dimension	• ReturnCode • Message Type • Method

Table 6-62 ocnssf_notification_trigger_tx_failure_total

Field	Details
Description	Count of failing notification triggers responses sent by NsSubscription.
Type	Counter
Service Operation	NSSubscription
Dimension	• Trigger type • returnCode

Table 6-63 ocnssf_indirect_communication_response_tx_failure_total

Field	Details
Description	Count of subscription creation responses sent by NSSF when indirect communication was enabled.
Type	Counter
Service Operation	NSSubscription
Dimension	• method • returnCode • bindingHeaderPresent • exceptionType

Table 6-64 ocnssf_subscription_response_tx_failure_total

Field	Details
Description	Count of subscription creation responses sent by NSSF
Type	Counter
Service Operation	NSSubscription
Dimension	• method • returnCode • exceptionType

Table 6-65 ocnssf_nssaiavailability_error_tx_total

Field	Details
Description	Count of error response messages sent by NSSF for requests for the Nnssf_NSSAIAvailability service.
Type	Counter

Table 6-65 (Cont.) ocnssf_nssaiavailability_error_tx_total

Field	Details
Service Operation	NSAvailability
Dimension	• AMF Instance Id • Method • Message Type • ReturnCode

Table 6-66 ocnssf_nssaiavailability_options_tx_status_unsupportedmediatype_total

Field	Details
Description	Count of HTTP OPTIONS response with status 415 Unsupported Media type.
Type	Counter
Service Operation	NSAvailability
Dimension	• Message Type • Method

Table 6-67 ocnssf_nsavailability_unsupported_plmn_total

Field	Details
Description	Count of request messages with unsupported PLMN received by NSSF for the ocnssf_NSAvailability service.
Type	Counter
Service Operation	NSAvailability
Dimension	• AMF Instance Id • Message Type • Method

6.1.3 NSSF OAuth Metrics

This section provides details about the NSSF OAuth metrics.

Table 6-68 oc_oauth_nrf_request_total

Field	Details
Description	This is pegged in the OAuth client implementation if the request is sent to NRF for requesting the OAuth token. OAuth client implementation is used in Egress gateway.
Type	Counter
Dimension	• ConsumerNFInstanceId • ConsumerNFType • TargetNFType TargetNFInstanceId • scope NrfFqdn

Table 6-69 oc_oauth_nrf_response_success_total

Field	Details
Description	This is pegged in the OAuth client implementation if an OAuth token is successfully received from the NRF. OAuth client implementation is used in Egress gateway.
Type	Counter
Dimension	• ConsumerNFInstanceId • ConsumerNFType • TargetNFType • TargetNFInstanceId • scope StatusCode NrfFqdn

Table 6-70 oc_oauth_nrf_response_failure_total

Field	Details
Description	This is pegged in the OAuthClientFilter in Egress gateway whenever GetAccessTokenFailedException is captured.
Type	Counter
Dimension	• ConsumerNFInstanceId • ConsumerNFType • TargetNFType • TargetNFInstanceId • scope StatusCode NrfFqdn

Table 6-71 oc_oauth_nrf_response_failure_total

Field	Details
Description	This is pegged in the OAuthClientFilter in Egress gateway whenever GetAccessTokenFailedException is captured.
Type	Counter
Dimension	• ConsumerNFInstanceId • ConsumerNFType • TargetNFType • TargetNFInstanceId • Scope • StatusCode • ErrorOriginator • NrfFqdn

Table 6-72 oc_oauth_request_failed_internal_total

Field	Details
Description	This is pegged in the OAuthClientFilter in Egress gateway whenever InternalServerErrorException is captured.
Type	Counter

Table 6-72 (Cont.) oc_oauth_request_failed_internal_total

Field	Details
Dimension	• ConsumerNFInstanceId • ConsumerNFType • TargetNFType • TargetNFInstanceId • scope • StatusCode • ErrorOriginator • NrfFqdn

Table 6-73 oc_oauth_token_cache_total

Field	Details
Description	This is pegged in the OAuth Client Implementation if the OAuth token is found in the cache.
Type	Counter
Dimension	• ConsumerNFInstanceId ConsumerNFType • TargetNFType TargetNFInstanceId scope

Table 6-74 oc_oauth_request_invalid_total

Field	Details
Description	This is pegged in the OAuthClientFilter in Egress gateway whenever a BadAccessTokenRequestException/JsonProcessingException is captured.
Type	Counter
Dimension	• ConsumerNFInstanceId • ConsumerNFType • TargetNFType • TargetNFInstanceId • scope • StatusCode • ErrorOriginator

Table 6-75 oc_oauth_validation_successful_total

Field	Details
Description	This is pegged in OAuth validator implementation if the received OAuth token is validated successfully. OAuth validator implementation is used in Ingress gateway.
Type	Counter
Dimension	• issuer • subject • scope

Table 6-76 oc_oauth_validation_failure_total

Field	Details
Description	This is pegged in OAuth validator implementation if the validation of the received OAuth token is failed. OAuth validator implementation is used in Ingress gateway.
Type	Counter
Dimension	• issuer • subject • scope • reason

Table 6-77 oc_oauth_cert_expiryStatus

Field	Details
Description	Metric used to peg expiry date of the certificate. This metric is further used for raising alarms if certificate expires within 30 days or 7 days.
Type	Gauge
Dimension	• id • certificateName • secretName

Table 6-78 oc_oauth_cert_loadStatus

Field	Details
Description	Metric used to peg whether given certificate can be loaded from secret or not. If it is loadable then "0" is pegged otherwise "1" is pegged. This metric is further used for raising alarms when certificate is not loadable.
Type	Gauge
Dimension	• id • certificateName • secretName

Table 6-79 oc_oauth_request_failed_cert_expiry

Field	Details
Description	Metric used to keep track of number of requests with keyId in token that failed due to certificate expiry. Pegged whenever oAuth Validator module throws oauth custom exception due to certificate expiry for an incoming request.
Type	Metric
Dimension	• target nf type • target nf instance id • consumer nf instance id • nrf instance id • service name of nf • producer service key id

Table 6-80 oc_oauth_keyid_count

Field	Details
Description	Metric used to keep track of number of requests received with keyId in token. Pegged whenever a request with an access token containing kid in header comes to oAuth Validator. This is independent of whether the validation failed or was successful.
Type	Metric
Dimension	• target nf type • target nf instance id • consumer nf instance id • nrf instance id • service name of nf • producer service key id

6.1.4 NsConfig Metrics

This section provides details about the NSSF Managed Object (MO) metrics.

Table 6-81 ocnssf_nsconfig_config_added_total

Field	Details
Description	Count of number of managed object which were added by NSConfig. Trigger Condition: Operator configuration of the Managed Object. Operator configuration of the Managed Object. This is pegged when HTTP POST request is received by NSSF's NsConfig.
Type	Counter
Service Operation	POST operation on plmnConfig, supportedSlicesMapping, barredSlicesMapping, nsiProfileConfig, grSites
Dimension	managed_object

Table 6-82 ocnssf_nsconfig_config_updated_total

Field	Details
Description	Count of number of managed object which were updated by NSConfig.Trigger Condition: Operator configuration of the Managed Object. Operator configuration of the Managed Object. This is pegged when HTTP PUT request is received by NSSF's NsConfig.
Type	Counter
Service Operation	PUT operation on plmnConfig, supportedSlicesMapping, barredSlicesMapping, nsiProfileConfig, grSites, systemOptions, allConfig
Dimension	managed_object

Table 6-83 ocnssf_nsconfig_config_deleted_total

Field	Details
Description	Count of number of managed object which were deleted by NSConfig.Trigger Condition: Operator configuration of the Managed Object. Operator configuration of the Managed Object. This is pegged when HTTP DELETE, or PUT request is received by NSSF's NsConfig.
Type	Counter
Service Operation	DELETE operation on plmnConfig, supportedSlicesMapping, barredSlicesMapping, nsiProfileConfig, grSites, deleteConfig and PUT operation on allConfig
Dimension	managed_object

Table 6-84 ocnssf_nsconfig_requests_rx_total

Field	Details
Description	Count of number of requests which were received by NSConfig.Trigger Condition: Operator configuration of the Managed Object. Operator configuration of the Managed Object. This is pegged when HTTP DELETE, or PUT request is received by NSSF's NsConfig.
Type	Counter
Service Operation	This is pegged when any HTTP request is received by NSSF's NsConfig.
Dimension	• managed_object • method

Table 6-85 ocnssf_nsconfig_response_tx_total

Field	Details
Description	Count of number of requests which were returned successful by NSConfig.Trigger Condition: Operator configuration of the Managed Object. Operator configuration of the Managed Object.
Type	Counter
Service Operation	This is pegged when any HTTP request is received by NSSF's NsConfig.
Dimension	• managed_object • method • status_code

Table 6-86 ocnssf_nsconfig_error_response_tx_total

Field	Details
Description	Count of number of requests which returned Error by NSConfig.Trigger Condition: Operator configuration of the Managed Object. Operator configuration of the Managed Object.
Type	Counter
Service Operation	This is pegged when any HTTP request is received by NSSF's NsConfig.
Dimension	• managed_object • method • status_code

6.1.5 Perf-info metrics for Overload Control

This section provides details about Perf-info metrics for overload control.

Table 6-87 cgroup_cpu_nanoseconds

Field	Details
Description	Reports the total CPU time (in nanoseconds) on each CPU core for all the tasks in the cgroup.
Type	Gauge
Dimension	NA

Table 6-88 cgroup_memory_bytes

Field	Details
Description	Reports the memory usage.
Type	Gauge
Dimension	NA

Table 6-89 load_level

Field	Details
Description	Provides information about the overload manager load level.
Type	Gauge
Dimension	• service • namespace

6.1.6 Egress Gateway Metrics

This section provides details about Egress Gateway metrics.

Table 6-90 oc_egressgateway_incoming_ip_type

Field	Details
Description	This is incremented when the IP type of the active incoming connections from the NSSF microservices to the Egress Gateway.
Type	Gauge
Dimension	• host • receivedAddressType

Table 6-91 oc_egressgateway_outgoing_ip_type

Field	Details
Description	This is incremented when the IP type of the active outgoing connections from Egress Gateway to the destination.
Type	Gauge

Table 6-91 (Cont.) oc_egressgateway_outgoing_ip_type

Field	Details
Dimension	• destinationHost • destinationHostAddressType

Table 6-92 oc_egressgateway_dual stack_ip_rejected_total

Field	Details
Description	This is incremented by counting the total IP rejections which are caused by a IP mismatch between the IP type configured in the `egressRoutingMode` and the IP type returned by DNS resolution.
Type	Gauge
Dimension	• authority • egressRoutingMode • dnsResolvedType

Table 6-93 oc_egressgateway_connection_failure_total

Field	Details
Description	Metric to capture failure when the destination is not reachable by Egress Gateway. Here, the destination is producer NF.
Type	Counter
Service Operation	Egress Gateway
Dimensions	• Host • Port • InstanceIdentifier • Direction • error_reason

Table 6-94 oc_egressgateway_outgoing_tls_connections

Field	Details
Description	Number of TLS connections received on the Egress Gateway and their negotiated TLS versions. The versions can be TLSv1.3 or TLSv1.2
Type	Gauge
Service Operation	Egress Gateway
Dimension	• NegotiatedTLSVersion • Host • Direction • InstanceIdentifier

Table 6-95 oc_fqdn_alternate_route_total

Field	Details
Description	Tracks number of registration, deregistration and GET calls received for a given scheme and FQDN. **Note**: Registration does not reflect active registration numbers. It captured number of registration requests received.
Type	Counter
Service Operation	Egress Gateway
Dimension	• type: Register/Deregister/GET • binding_value: <scheme>+<FQDN>

Table 6-96 oc_dns_srv_lookup_total

Field	Details
Description	Track number of time DNS SRV lookup was done for a given scheme and FQDN.
Type	Counter
Service Operation	Egress Gateway
Dimension	binding_value: <scheme>+<FQDN>

Table 6-97 oc_alternate_route_resultset

Field	Details
Description	Value provides number of alternate routes known for a given scheme and FQDN. Whenever DNS SRV lookup or static configuration is done, this metric provide number of known alternate route for a given pair. For example, <"http", "abc.oracle.com">: 2.
Type	Gauge
Service Operation	Egress Gateway
Dimension	binding_value: <scheme>+<FQDN>

Table 6-98 oc_configclient_request_total

Field	Details
Description	This metric is pegged whenever a polling request is made from config client to the server for configuration updates.
Type	Counter
Service Operation	Egress Gateway
Dimension	Tags: releaseVersion, configVersion. • releaseVersion tag indicates the current chart version of alternate route service deployed. • configVersion tag indicates the current configuration version of alternate route service.

Table 6-99 oc_configclient_response_total

Field	Details
Description	This metric is pegged whenever a response is received from the server to client.
Type	Counter
Service Operation	Egress Gateway

Table 6-99 (Cont.) oc_configclient_response_total

Field	Details
Dimension	Tags: releaseVersion, configVersion, updated. • releaseVersion tag indicates the current chart version of alternate route service deployed. • configVersion tag indicates the current configuration version of alternate route service. • updated tag indicates whether there is a configuration update or not.

Table 6-100 oc_egressgateway_peer_health_status

Field	Details
Description	It defines Egress Gateway peer health status. This metric is set to 1, if a peer is unhealthy. This metric is reset to 0, when it becomes healthy again.
Type	Gauge
Service Operation	Egress Gateway
Dimension	• peer • vfqdn

Table 6-101 oc_egressgateway_peer_health_ping_request_total

Field	Details
Description	It defines Egress Gateway peer health ping request. This metric is incremented every time Egress Gateway send a health ping towards a peer.
Type	Counter
Service Operation	Egress Gateway
Dimension	• peer • vfqdn • statusCode • cause

Table 6-102 oc_egressgateway_peer_health_ping_response_total

Field	Details
Description	Egress Gateway Peer health ping response. This metric is incremented every time a Egress Gateway receives a health ping response (irrespective of success or failure) from a peer.
Type	Counter
Service Operation	Egress Gateway
Dimension	• peer • vfqdn • statusCode • cause

Table 6-103 oc_egressgateway_peer_health_status_transitions_total

Field	Details
Description	It defines Egress Gateway peer health status transitions. Egress Gateway increments this metric every time a peer transitions from available to unavailable or unavailable to available.
Type	Counter
Service Operation	Egress Gateway
Dimension	• peer • vfqdn • from • to

Table 6-104 oc_egressgateway_peer_count

Field	Details
Description	It defines Egress Gateway peer count. This metric is incremented every time for the peer count.
Type	Gauge
Service Operation	Egress Gateway
Dimension	peerset

Table 6-105 oc_egressgateway_peer_available_count

Field	Details
Description	It defines Egress Gateway available peer count. This metric is incremented every time for the available peer count.
Type	Gauge
Service Operation	Egress Gateway
Dimension	peerset

Table 6-106 oc_egressgateway_user_agent_consumer

Field	Details
Description	Whenever the feature is enabled and User-Agent Header is getting generated.
Type	Counter
Service Operation	Egress Gateway
Dimension	ConsumerNfInstanceId: ID of consumer NF (NSSF) as configured in Egress Gateway.

Table 6-107 oc_egressgateway_ip_addresses_fetch_failure

Field	Details
Description	This metric is pegged when an exception occurs while fetching IP addresses of the services from kube-api-server.
Type	Counter
Service Operation	Egress Gateway
Dimension	NA

6.1.7 Ingress Gateway Metrics

This section provides details about Ingress Gateway metrics.

Table 6-108 oc_ingressgateway_incoming_ip_type

Field	Details
Description	This is incremented when the IP type of the active incoming connections from the client to Ingress Gateway.
Type	Gauge
Dimension	• host • receivedAddressType

Table 6-109 oc_ingressgateway_outgoing_ip_type

Field	Details
Description	This is incremented when the IP type of the active outgoing connections from Ingress Gateway to the backend services.
Type	Gauge
Dimension	• BackendSvc • BackendSvcAddressType

Table 6-110 oc_ingressgateway_connection_failure_total

Field	Details
Description	Metric to capture the connection failures when connected to the destination service fails. Here in case of Ingress Gateway, the destination service is a backend microservice of the NF.
Type	Counter
Service Operation	Ingress Gateway
Dimensions	• Host • Port • InstanceIdentifier • Direction • error_reason • ErrorOriginator

Table 6-111 oc_ingressgateway_incoming_tls_connections

Field	Details
Description	Number of TLS connections received on the Ingress Gateway and their negotiated TLS versions. The versions can be TLSv1.3 or TLSv1.2.
Type	Gauge
Service Operation	Ingress Gateway

Table 6-111 (Cont.) oc_ingressgateway_incoming_tls_connections

Field	Details
Dimension	• NegotiatedTLSVersion • Host • Direction • InstanceIdentifier

Table 6-112 oc_ingressgateway_ip_addresses_fetch_failure

Field	Details
Description	This metric is pegged when an exception occurs while fetching IP addresses of the services from kube-api-server
Type	Counter
Service Operation	Ingress Gateway
Dimension	NA

6.1.8 NSSF Common metrics

This section provides details about the NSSF common metrics.

Table 6-113 security_cert_x509_expiration_seconds

Field	Details
Description	Indicates the time to certificate expiry in epoch seconds.
Type	Histogram
Dimension	serialNumber

Table 6-114 http_requests_total

Field	Details
Description	This is pegged as soon as the request reaches the Ingress or Egress gateway in the first custom filter of the application.
Type	Counter
Dimension	• direction: ingress or egress • method: the method from the request line • uri: the URI from the request line • http_version: the HTTP version from the request line • host: the value of the Host header field • NFType • NFServiceType • HttpVersion • Scheme • Route_path • InstanceIdentifier • ClientCertIdentity

Table 6-115 http_responses_total

Field	Details
Description	Responses received or sent from the microservice .
Type	Counter
Dimension	• Status • Method • Route_path • NFType • NFServiceType • Host • HttpVersion • Scheme • InstanceIdentifier • ClientCertIdentity

Table 6-116 http_request_bytes

Field	Details
Description	Size of requests, including header and body. Grouped in 100 byte buckets.
Type	Histogram
Dimension	• direction • method • uri • http_version

Table 6-117 http_response_bytes

Field	Details
Description	Size of responses, including header and body. Grouped in 100 byte buckets.
Type	Histogram
Dimension	• direction • http_version

Table 6-118 bandwidth_bytes

Field	Details
Description	Amount of ingress and egress traffic sent and received by the microservice.
Type	Counter
Dimension	direction

Table 6-119 request_latency_seconds

Field	Details
Description	This metric is pegged in the last custom filter of the Ingress or Egress gateway while the response is being sent back to the consumer NF. It tracks the amount of time taken for processing the request. It starts as soon the request reaches the first custom filter of the application and lasts till the response is sent back to the consumer NF from the last custom filter of the application.

Table 6-119 (Cont.) request_latency_seconds

Field	Details
Type	Histogram
Dimension	• quantile • InstanceIdentifier • Route_path • Method

Table 6-120 connection_failure_total

Field	Details
Description	This metric is pegged by jetty client when the destination is not reachable by Ingress or Egress gateway. In case of Ingress gateway, the destination service will be a back-end microservice of the NF, and TLS connection failure metrics when connecting to ingress with direction as ingress. For Egress gateway, the destination is producer NF.
Type	Counter
Dimension	• Host • Port • InstanceIdentifier • Direction • error_reason

Table 6-121 request_processing_latency_seconds

Field	Details
Description	This metric is pegged in the last custom filter of the Ingress or Egress gateway while the response is being sent back to the consumer NF. This metric captures the amount of time taken for processing of the request only within Ingress or Egress gateway. It starts as soon the request reaches the first custom filter of the application and lasts till the request is forwarded to the destination.
Type	Timer
Dimension	• quantile • InstanceIdentifier • Route_path • Method

Table 6-122 jetty_request_stat_metrics_total

Field	Details
Description	This metric is pegged for every event occurred when a request is sent to Ingress or Egress gateway.
Type	Counter
Dimension	• event • client_type • InstanceIdentifier

Table 6-123 jetty_response_stat_metrics_total

Field	Details
Description	This metric is pegged for every event occurred when a response is received by Ingress or Egress gateway.
Type	Counter
Dimension	• event • client_type • InstanceIdentifier

Table 6-124 server_latency_seconds

Field	Details
Description	This metric is pegged in Jetty response listener that captures the amount of time taken for processing of the request by jetty client
Type	Timer
Dimension	• quantile • InstanceIdentifier • Method

Table 6-125 roundtrip_latency_seconds

Field	Details
Description	This metric is pegged in Netty outbound handler that captures the amount of time taken for processing of the request by netty server.
Type	Timer
Dimension	• quantile • InstanceIdentifier • Method

Table 6-126 oc_configclient_request_total

Field	Details
Description	This metric is pegged whenever config client is polling for configuration update from common configuration server.
Type	Counter
Dimension	• Release version • Config version

Table 6-127 oc_configclient_response_total

Field	Details
Description	This metrics is pegged whenever config client receives response from common configuration server.
Type	Counter
Dimension	• Release version • Config version • Updated

Table 6-128 incoming_connections

Field	Details
Description	This metric pegs active incoming connections from client to Ingress or Egress gateway.
Type	Gauge
Dimension	• Direction • Host • InstanceIdentifier

Table 6-129 outgoing_connections

Field	Details
Description	This metric pegs active outgoing connections from Ingress gateway or Egress gateway to destination
Type	Gauge
Dimension	• Direction • Host • InstanceIdentifier

Table 6-130 sbitimer_timezone_mismatch

Field	Details
Description	This metric pegs when sbiTimerTimezone is set to ANY and time zone is not specified in the header then above metric is pegged in ingress and egress gateways.
Type	Gauge
Dimension	• Route_path • Method

Table 6-131 nrfclient_nrf_operative_status

Field	Details
Description	The current operative status of the NRF Instance. **Note**: The **HealthCheck** mechanism is an important component that allows monitoring and managing the health of NRF services. When enabled, it makes periodic HTTP requests to NRF services to check their availability and updates their status accordingly so that the metric `nrfclient_nrf_operative_status` updates properly. When disabled, for each NRF route, it is checked whether the retry time has expired. If so, the health state is reset to `"HEALTHY"`, and the retry time is cleared.
Type	Gauge
Dimension	NrfUri - URI of the NRF Instance

Table 6-132 nrfclient_dns_lookup_request_total

Field	Details
Description	Total number of times a DNS lookup request is sent to the alternate route service.
Type	Counter

Table 6-132 (Cont.) nrfclient_dns_lookup_request_total

Field	Details
Dimension	• Scheme • VirtualFqdn

Table 6-133 oc_certificatemanagement_tls_certificate_info

Field	Details
Description	This metric pegs the status as 1 if any file in the configured secret fails to load its name, and Reason is pegged in dimension. When the secret is loaded again, the status of the current entry is changed to 0. If the certificate is loaded without any issue, there will be no entry in Prometheus and the previous entry value will be changed to 0.
Type	Gauge
Dimension	• Name • SecretName • Reason (CERT_CORRUPT,CERT_MISSING,CERT_EXPIRED) • Service (IngressGateway) **Note:** From this release, the Status dimension is renamed as Reason and the CertificateName dimension is renamed as Name.

Table 6-134 oc_certificatemanagement_tls_secret_status

Field	Details
Description	This metric is used to determine whether the configured secret is available in the namespace. If the secret is available in the namespace, the metric is pegged with value 0. If the secret is unavailable or if it is removed, then the metric is pegged with status 1.
Type	Gauge
Dimension	• SecretName • Namespace • Service

Table 6-135 oc_certificatemanagement_tls_certs_reload_failure

Field	Details
Description	This metric is pegged when any exception occurs while reloading certificates. For example, failure in the creation of a keystore. If there is an error, the metric is pegged with value 1. If the reload is successful, the metric is pegged with value 0.
Type	Gauge
Dimension	NA

Table 6-136 cgiu_jetty_ip_address_fetch_failure

Field	Details
Description	This metric would be pegged when exception occurs during fetching of IP addresses of services from kube-api-server.
Type	Counter
Dimension	NA

6.1.9 NSSF Cache Metrics

Table 6-137 ocnssf_cache_latency_seconds_count

Field	Details
Description	This metric is used to show the time taken in seconds to create/update/rebuild the cache. NOTE: Right now only rebuild cache's duration will be shown via this metric
Type	Histogram
Dimension	• type • duration_type • number_of_records

Table 6-138 ocnssf_cache_latency_seconds_max

Field	Details
Description	This metric is used to show the time taken in seconds to create/update/rebuild the cache. NOTE: Right now only rebuild cache's duration will be shown via this metric
Type	Histogram
Dimension	• type • duration_type • number_of_records

Table 6-139 ocnssf_cache_latency_seconds_sum

Field	Details
Description	This metric is used to show the time taken in seconds to create/update/rebuild the cache. NOTE: Right now only rebuild cache's duration will be shown via this metric
Type	Histogram
Dimension	• type • duration_type • number_of_records

Table 6-140 ocnssf_cache_latency_seconds_bucket

Field	Details
Description	This metric is used to show the time taken in seconds to create/update/rebuild the cache. NOTE: Right now only rebuild cache's duration will be shown via this metric

Table 6-140 (Cont.) ocnssf_cache_latency_seconds_bucket

Field	Details
Type	Histogram
Dimension	• type • duration_type • number_of_records

Table 6-141 ocnssf_cache_rebuild_success_total

Field	Details
Description	This metric is used to show the time taken in seconds to create/update/rebuild the cache for NsSelection and NsSubscription. NOTE: Right now only rebuild cache's success will be shown via this metric
Type	Counter
Dimension	NA

Table 6-142 ocnssf_cache_rebuild_error_total

Field	Details
Description	This metric is used to show the time taken in seconds to create/update/rebuild the cache for NsSelection and NsSubscription. NOTE: Right now only rebuild cache's failure will be shown via this metric
Type	Counter
Dimension	NA

Table 6-143 cache_create_success_total

Description	Count of fresh cache created successfully in selection service for required tables
Type	Counter
Service Operation	NSSF Service Name
Dimensions	• table_name • no_of_records • query_duration • cache_update_duration

Table 6-144 cache_update_success_total

Description	Count of cache updated successfully in selection service for required tables
Type	Counter
Service Operation	NSSF Service Name
Dimensions	• table_name • no_of_records • query_duration • cache_update_duration

Table 6-145 cache_create_error_total

Description	Count of fresh cache creation error in selection service for required tables
Type	Counter
Service Operation	NSSF Service Name
Dimensions	• table_name • error_cause • query_duration

Table 6-146 cache_update_error_total

Description	Count of cache updation error in selection service for required tables
Type	Counter
Service Operation	NSSF Service Name
Dimensions	• table_name • error_cause • query_duration

6.2 NSSF KPIs

This section includes information about KPIs for Oracle Communications Cloud Native Core, Network Slice Selection Function.

The following are the NSSF KPIs:

6.2.1 NSSelection KPIs

Table 6-147 NSSF NSSelection Initial Registration Success Rate

Field	Details
Description	Percentage of NSSelection Initial registration messages with success response
Expression	sum(ocnssf_nsselection_success_tx_total{message_type=\"registration\"})/ sum(ocnssf_nsselection_rx_total{message_type=\"registration\"}))*100"

Table 6-148 NSSF NSSelection PDU establishment success rate

Field	Details
Description	Percentage of NSSelection PDU establishment messages with success response
Expression	sum(ocnssf_nsselection_success_tx_total{message_type=\"pdu_session\"})/ sum(ocnssf_nsselection_rx_total{message_type=\"pdu_session\"}))*100"

Table 6-149 NSSF NSSelection UE-Config Update success rate

Field	Details
Description	Percentage of NSSelection UE-Config Update messages with success response
Expression	sum(ocnssf_nsselection_success_tx_total{message_type=\"ue_config_update\"})/ sum(ocnssf_nsselection_rx_total{message_type=\"ue_config_update\"}))*100",

Table 6-150 4xx Responses (NSSelection)

Field	Details
Description	Rate of 4xx response for NSSelection
Expression	sum(increase(oc_ingressgateway_http_responses_total{Status=~"4.* ",Uri=~".*nnssf-nsselection.*",Method="GET"}[5m]))

Table 6-151 5xx Responses (NSSelection)

Field	Details
Description	Rate of 5xx response for NSSelection
Expression	sum(increase(oc_ingressgateway_http_responses_total{Status=~"5.* ",Uri=~".*nnssf-nsselection.*",Method="GET"}[5m])

6.2.2 NSAvailability KPIs

Table 6-152 NSSF NSAvailability PUT success rate

Field	Details
Description	Percentage of NSAvailability UPDATE PUT messages with success response
Expression	sum(ocnssf_nssaiavailability_success_tx_total{message_type=\"availability_update\"}{method=\"PUT"})/sum(ocnssf_nssaiavailability_rx_total{message_type=\"availability_update\"}{method=\"PUT"}))*100"

Table 6-153 NSSF NSAvailability PATCH success rate

Field	Details
Description	Percentage of NSAvailability UPDATE PATCH messages with success response
Expression	sum(ocnssf_nssaiavailability_success_tx_total{message_type=\"availability_update\"}{method=\"PATCH"})/sum(ocnssf_nssaiavailability_rx_total{message_type=\"availability_update\"}{method=\"PATCH"}))*100"

Table 6-154 NSSF NSAvailability Delete success rate

Field	Details
Description	Percentage of NSAvailability Delete messages with success response
Expression	sum(ocnssf_nssaiavailability_success_tx_total{message_type=\"availability_update\"}{method=\"DELETE"})/ sum(ocnssf_nssaiavailability_rx_total{message_type=\"availability_update\"}{method=\"DELETE"}))*100""

Table 6-155 NSSF NSAvailability Subscribe success rate

Field	Details
Description	Percentage of NSAvailability Subscribe messages with success response

Table 6-155 (Cont.) NSSF NSAvailability Subscribe success rate

Field	Details
Expression	sum(ocnssf_nssaiavailability_success_tx_total{message_type=\"availability_subscribe\"}{method=\"POST\"})/ sum(ocnssf_nssaiavailability_rx_total{message_type=\"availability_subscribe\"}{method=\"POST\"}))*100"

Table 6-156 NSSF NSAvailability Unsubscribe success rate

Field	Details
Description	Percentage of NSAvailability Unsubscribe messages with success response
Expression	sum(ocnssf_nssaiavailability_success_tx_total{message_type=\"availability_subscribe\"}{method=\"DELETE\"})/ sum(ocnssf_nssaiavailability_rx_total{message_type=\"availability_subscribe\"}{method=\"DELETE\"}))*100"

Table 6-157 4xx Responses (NSAvailability)

Field	Details
Description	Rate of 4xx response for NSAvailability
Expression	sum(increase(oc_ingressgateway_http_responses_total{Status=~"4.* ",Uri=~".*nnssf-nsavailability.*",Method="GET"}[5m]))

Table 6-158 5xx Responses (NSAvailability)

Field	Details
Description	Rate of 5xx response for NSAvailability
Expression	sum(increase(oc_ingressgateway_http_responses_total{Status=~"4.* ",Uri=~".*nnssf-nsavailability.*",Method="GET"}[5m]))

6.2.3 Ingress Gateway KPIs

Table 6-159 NSSF Ingress Request

Field	Details
Description	Rate of HTTP requests received at NSSF Ingress Gateway
Expression	oc_ingressgateway_http_requests

6.3 NSSF Alerts

This section includes information about alerts for Oracle Communications Network Slice Selection Function.

> **Note**
>
> The performance and capacity of the NSSF system may vary based on the call model, feature or interface configuration, network conditions, and underlying CNE and hardware environment.

You can configure alerts in Prometheus and `ocnssf_alert_rules_25.2.200.yaml` file.

The following table describes the various severity types of alerts generated by NSSF:

Table 6-160 Alerts Levels or Severity Types

Alerts Levels / Severity Types	Definition
Critical	Indicates a severe issue that poses a significant risk to safety, security, or operational integrity. It requires immediate response to address the situation and prevent serious consequences. Raised for conditions may affect the service of NSSF.
Major	Indicates a more significant issue that has an impact on operations or poses a moderate risk. It requires prompt attention and action to mitigate potential escalation. Raised for conditions may affect the service of NSSF.
Minor	Indicates a situation that is low in severity and does not pose an immediate risk to safety, security, or operations. It requires attention but does not demand urgent action. Raised for conditions may affect the service of NSSF.
Info or Warn (Informational)	Provides general information or updates that are not related to immediate risks or actions. These alerts are for awareness and do not typically require any specific response. WARN and INFO alerts may not impact the service of NSSF.

> **Caution**
>
> User, computer and applications, and character encoding settings may cause an issue when copy-pasting commands or any content from PDF. The PDF reader version also affects the copy-pasting functionality. It is recommended to verify the pasted content when the hyphens or any special characters are part of the copied content.

> **Note**
>
> - `kubectl` commands might vary based on the platform deployment. Replace `kubectl` with Kubernetes environment-specific command line tool to configure Kubernetes resources through kube-api server. The instructions provided in this document are as per the Oracle Communications Cloud Native Environment (OCCNE) version of kube-api server.
> - The alert file can be customized as required by the deployment environment. For example, namespace can be added as a filtered criteria to the alert expression to filter alerts only for a specific namespace.

6.3.1 Alert Configuration

This section describes how to configure alert rules for the NSSF in Prometheus. It provides guidance on setting up measurement-based alert rules, where the alerting system evaluates

metrics reported by NSSF microservices against specified rule conditions to generate alerts as needed.

Prometheus Alert Configuration

In a Prometheus environment, NSSF alert rules are configured based on metrics reported by NSSF components. The alerting workflow monitors these metrics and issues notifications when the defined conditions are met.

For more information about configuring NSSF alerts in Prometheus, see the “Alert Configuration” section in *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*.

6.3.2 System Level Alerts

This section lists the system level alerts.

6.3.2.1 OcnssfNfStatusUnavailable

Table 6-161 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 6-161 (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*.

6.3.2.2 OcnssfPodsRestart

Table 6-162 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 6-162 (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*.

6.3.2.3 OcnssfSubscriptionServiceDown

Table 6-163 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 6-163 (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*.

6.3.2.4 OcnssfSelectionServiceDown

Table 6-164 OcnssfSelectionServiceDown

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

Table 6-164 (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*.

6.3.2.5 OcnssfAvailabilityServiceDown

Table 6-165 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 6-165 (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*.

6.3.2.6 OcnssfConfigurationServiceDown

Table 6-166 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 6-166 (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*.

6.3.2.7 OcnssfAppInfoServiceDown

Table 6-167 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 6-167 (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*.

6.3.2.8 OcnssfIngressGatewayServiceDown

Table 6-168 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 6-168 (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*.

6.3.2.9 OcnssfEgressGatewayServiceDown

Table 6-169 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 6-169 (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*.

6.3.2.10 OcnssfOcpmConfigServiceDown

Table 6-170 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 6-170 (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*.

6.3.2.11 OcnssfPerfInfoServiceDown

Table 6-171 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 6-171 (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*.

6.3.2.12 OcnssfNrfClientManagementServiceDown

Table 6-172 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 6-172 (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*.

6.3.2.13 OcnssfAlternateRouteServiceDown

Table 6-173 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 6-173 (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*.

6.3.2.14 OcnssfAuditorServiceDown

Table 6-174 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 6-174 (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*.

6.3.2.15 OcnssfTotalIngressTrafficRateAboveMinorThreshold

Table 6-175 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 6-175 (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.

6.3.2.16 OcnssfTotalIngressTrafficRateAboveMajorThreshold

Table 6-176 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.

6.3.2.17 OcnssfTotalIngressTrafficRateAboveCriticalThreshold

Table 6-177 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.

6.3.2.18 OcnssfTransactionErrorRateAbove1Percent

Table 6-178 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 6-178 (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.

6.3.2.19 OcnssfTransactionErrorRateAbove10Percent

Table 6-179 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 6-179 (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.

6.3.2.20 OcnssfTransactionErrorRateAbove25Percent

Table 6-180 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 6-180 (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.

6.3.2.21 OcnssfTransactionErrorRateAbove50Percent

Table 6-181 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 6-181 (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.

6.3.3 Application Level Alerts

This section lists the application level alerts.

6.3.3.1 OcnssfOverloadThresholdBreachedL1

Table 6-182 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 6-182 (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*.

6.3.3.2 OcnssfOverloadThresholdBreachedL2

Table 6-183 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*.

6.3.3.3 OcnssfOverloadThresholdBreachedL3

Table 6-184 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*.

6.3.3.4 OcnssfOverloadThresholdBreachedL4

Table 6-185 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 6-185 (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*.

6.3.3.5 OcnssfScpMarkedAsUnavailable

Table 6-186 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.

6.3.3.6 OcnssfAllScpMarkedAsUnavailable

Table 6-187 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 6-187 (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.

6.3.3.7 OcnssfTLSCertificateExpireMinor

Table 6-188 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*.

6.3.3.8 OcnssfTLSCertificateExpireMajor

Table 6-189 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*.

6.3.3.9 OcnssfTLSCertificateExpireCritical

Table 6-190 OcnssfTLSCertificateExpireCritical

Field	Details
Description	'TLS certificate to expire in one month.'

Table 6-190 (Cont.) OcnssfTLSCertificateExpireCritical

Field	Details
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*.

6.3.3.10 OcnssfNrfInstancesInDownStateMajor

Table 6-191 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*.

6.3.3.11 OcnssfAllNrfInstancesInDownStateCritical

Table 6-192 OcnssfAllNrfInstancesInDownStateCritical

Field	Details
Description	'When current operative status of all the NRF Instances is unavailable/unhealthy'

Table 6-192 (Cont.) OcnssfAllNrfInstancesInDownStateCritical

Field	Details
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*.

6.3.4 Configuring SNMP Notifier

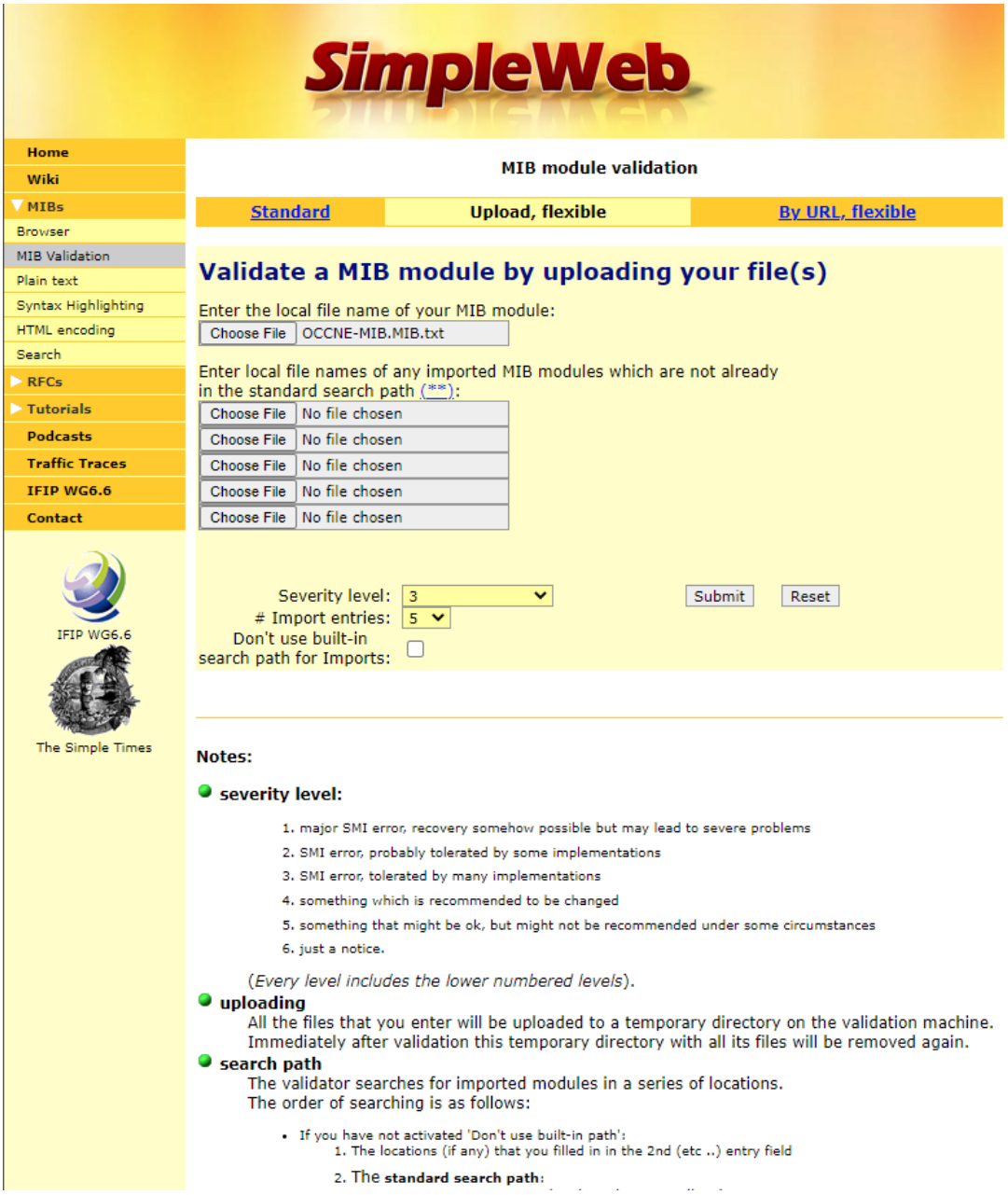
This section describes the procedure to configure SNMP Notifier.

The SNMP MIB files are used to define the MIB objects. When uploaded to Wireshark and MIB tools, such as MIB Browser and Trap Receiver, users can see the detailed MIB definition instead of just the OID. All tools require the valid syntax of MIB files, and even minor errors can cause the upload to fail.

Procedure to Validate MIB Files

This procedure explains how to validate the MIB files and how to fix some common errors.

1. **Download MIB Files:** Download the MIB files onto your PC. In the NSSF environment, the files are named as follows:
   - NSSF-MIB.mib
   - NSSF-TC.mib
   - TEKELEC-TOPLEVEL-REG.mib
     These files are located in the path /ocnssf/observability/mib.
2. **Open Simpleweb MIB Validator:** Open the Simpleweb MIB validator page.
3. **Upload MIB Files:**



a. Under "**Enter the local file name of your MIB module**," click the "**Choose File**" button.

b. Select the MIB file you want to validate, click "**Open**," and the file will be added to the web page.

4. **Inspect MIB Definitions:**

   a. Open the MIB file on your PC using any suitable application.

   b. In the "**IMPORTS**" section, identify the MIB definitions listed after the word "**FROM**."

   c. Skip standard MIBs such as "**SNMPv2-SMI**," "**SNMPv2-TC**," etc., as they are already included in the Simpleweb MIB validator by default.

5. **Handling Private MIBs:**

a. For other MIBs, especially private MIB files, locate the corresponding MIB file for each definition.

b. If the MIB file names differ from the MIB DEFINITIONS, rename the file as <MIB definition>.mib. For example:

- Original MIB file name: Private-MIB-File.mib
- MIB definition in "IMPORTS" section: FROM PRIVATE-MIB Rename the file to: PRIVATE-MIB.mib

6. **Upload Corrected File Names:**

a. Upload the corrected file names into the Simpleweb MIB validator. In NSSF, the corrected file names will be NSSF_MIB.mib, NSSF-TC.mib, TEKELEC-TOPLEVEL-REG.mib.

b. Click "**Choose Files**" and then click "**Submit**" to complete the process.

By following these steps, you can ensure the proper validation of MIB files, including the handling of standard and private MIBs.

Fixes for Common MIB Compliance Issues

Import SNMPv2-SMI:

Message: "Invalid status 'current' in SMIv1 MIB"

Fix Method: Add "MODULE-IDENTITY FROM SNMPv2-SMI"

Example format:

```
IMPORTS
     TEXTUAL-CONVENTION FROM SNMPv2-TC
     MODULE-IDENTITY FROM SNMPv2-SMI
     oracleCNE FROM TEKELEC-TOPLEVEL-REG;
```

Last Update and Revision:

Messages:

- "Revision date after last update"
- "Revision not in reverse chronological order"
- "Revision for the last update is missing"

Fix Method:

- Ensure LAST-UPDATED is exactly the same as the most recent REVISION.
- Create a separate "REVISION HISTORY" section.
- Put all REVISIONs in this section in reverse chronological order.

Example format:

```
oracleNssfMIB MODULE-IDENTITY
     LAST-UPDATED "202302091734Z"
     ...
     REVISION    "202302091734Z"
     DESCRIPTION "Updated."
     ::= { oracleNSSF 1 }
```

Case-Sensitive Names:

Message: "<name> should start with a lowercase letter"

Fix Method: Change the first letter to lowercase.

Example format:

```
OCNSSFConfigurationServiceDown NOTIFICATION-TYPE
     ...
```

Fix:

```
ocnssfConfigurationServiceDown NOTIFICATION-TYPE
     ...
```

Duplicate OID:

Message: "`Identifier ocnssfIngressGatewayServiceDown` registers object identifier already registered by `ocnssfConfigurationServiceDown'"`

Fix Method: Change the OID to an unused number.

Example format:

```
ocnssfIngressGatewayServiceDown NOTIFICATION-TYPE
     ...
     ::= { oracleNssfMIBNotifications  9006 }
```

Fix:

```
ocnssfIngressGatewayServiceDown NOTIFICATION-TYPE
     ...
     ::= { oracleNssfMIBNotifications  9007 }
```

MIB File Missing "END":

Message: "

```
Syntax error, unexpected $end'"
```

Fix Method: Add the word "END" at the end of the MIB file.

Example format:

```
ocnssfPerfInfoServiceDown NOTIFICATION-TYPE
     ...
     ::= { oracleNssfMIBNotifications  9036 }
END
```

7

Appendix A- HTTP Response Codes

NsSelection Scenarios & Error Codes

Table 7-1 General Scenarios

Scenario	Error Code	Error String
Mandatory query params missing	400	Missing required parameters: <param>
Duplicates present in query params	400	Duplicate query param: <param>
Unknown Query params	400	Unknown Parameter/Parameters found: <param>
Incorrect Json Format	400	Invalid JSON format: <reason>
Unrecognized property in Json	400	Unrecognized Property detected: <param>
Unsupported PLMN received	403	Unsupported PLMN received
Both content-type & accept headers invalid	415	Content Type [application/invalid] not allowed. Allowed types: [application/json]
Invalid Accept header	406	Specified Accept Types [application/invalid] not supported. Supported types: [application/json]

Table 7-2 Initial Registration and UE Configuration Update

Scenarios	Error Code	Updated Error Response
Constraint violations for request parameters (multiple validation issues)	400	- `Mandatory parameter missing`: `<param>`. - `Optional parameter incorrect`: Size must be between `1–8` (for requested NSSAI list). - `Optional parameter incorrect`: `sd` must be a 6-digit hexadecimal. - `Mandatory parameter incorrect`: `sst` must be between `0–255`. - `Mandatory parameter incorrect`: `tac` is incorrect — must match regex `(^([0-9a-fA-F]){4}$)
Duplicates present in lists (`subscribed` / `requested`)	400	Duplicate S-NSSAI present in `subscribedNssai` list: `{snssai}`. Duplicate S-NSSAI present in `requestedNssai` list: `{snssai}`.
When AutoConfig is off and no AMF supports any allowed S-NSSAI	403	Unable to find an AMF that supports the allowed S-NSSAIs.
No allowed S-NSSAI found	403	Unable to compute `AllowedNssai`.
No subscribed S-NSSAI is part of configured S-NSSAI	403	Unable to compute `AllowedNssai`.

Table 7-2 (Cont.) Initial Registration and UE Configuration Update

Scenarios	Error Code	Updated Error Response
No S-NSSAI configured for PLMN	403	Unable to compute `AllowedNssai`. (With the re-architectured NsSelection code, `AllowedNssai` is computed first; if no S-NSSAI is configured for the PLMN in `plmnInfo.configuredNssai` we respond with `Unable to compute AllowedNssai`.)

Table 7-3 PDU Session Establishment

Scenarios	Error Code	Updated Error Response
No data configured for requested PLMN (`plmnInfo.configuredNssai` is empty)	403	S-NSSAI is not authorized — PLMN data is not configured for the requested PLMN. (Detailed error will be returned.)
S-NSSAI not part of `plmnInfo.configuredNssai` for requested PLMN	403	S-NSSAI is not authorized. (Detailed error will be returned.)
S-NSSAI part of `plmnInfo.configuredNssai` but restricted at PLMN	403	S-NSSAI is not authorized. (Detailed error will be returned.)
S-NSSAI part of `plmnInfo.configuredNssai` but restricted at TAI	403	S-NSSAI is not authorized. (Detailed error will be returned.)
AutoConfig is off and `SupportedSlicesMapping` not configured for requested TAI	403	S-NSSAI is not authorized — `SupportedSlicesMapping` data is not configured for the requested TAI. (Detailed error will be returned.)
AutoConfig is off and requested S-NSSAI not supported in the given TAI (operator config)	403	S-NSSAI is not authorized. (Detailed error will be returned.)
S-NSSAI not supported by any AMF in the TAI (based on NsAvailability data)	403	S-NSSAI is not authorized — unable to find an AMF that supports the requested S-NSSAI. (Detailed error will be returned.)

NsAvailability Scenarios & Error Codes

Table 7-4 NsAvailability PUT

Scenarios	Error Code	Updated Error Response
Mandatory params missing	400	- Case-1: • Cause: MANDATORY_IE_MISSING • Detail: Mandatory parameter missing • Reason: Mandatory parameter missing: TAI (parameter name fully capitalized for TAI, PLMN, SNSSAI, SST, SD, TAC, MCC, MNC) - Case-2: • Cause: MANDATORY_IE_MISSING • Detail: Mandatory parameter missing • Reason: Mandatory parameter missing: supportedSnssaiList (param name in normal text format)
Invalid mandatory params value	400	Cause: MANDATORY_IE_INCORRECT • Detail: Mandatory parameter incorrect • Reason: Mandatory parameter incorrect: <param name> is incorrect
Invalid optional params value	400	Cause: OPTIONAL_IE_INCORRECT • Detail: Optional IE incorrect • Reason: Optional parameter incorrect: supportedFeatures (param name with message if required)
Incorrect JSON format	400	Cause: INVALID_INPUT_DATA • Detail: Invalid JSON format
Unrecognized property in JSON	400	Cause: INVALID_INPUT_DATA • Detail: Unrecognized property detected: <param name>
Unsupported PLMN received	403	Cause: PLMN_NOT_SUPPORTED • Detail: Unsupported PLMN(s) received: <unsupported PLMN details>
Unconfigured S-NSSAI received	403	Cause: SNSSAI_NOT_SUPPORTED • Detail: Unconfigured S-NSSAI(s) found: SNSSAI <details> for TAI: <details>
Payload too large	413	Cause: PAYLOAD_TOO_LARGE • Detail: Request payload size is too large, configured size: <configured size>, requested payload size: <requested size>
Duplicate S-NSSAI in request	400	Cause: INVALID_INPUT_DATA • Detail: Duplicate entries found in supportedSnssaiList for SNSSAI <details> for TAI: <details>

Table 7-4 (Cont.) NsAvailability PUT

Scenarios	Error Code	Updated Error Response
Duplicate TAI in request	400	Cause: INVALID_INPUT_DATA • Detail: Duplicate TAI found in supportedNssaiAvailabilityData for TAI: <details>
Request with ONSSAI data (TaiList/ TaiRangeList)	400	Cause: INVALID_INPUT_DATA • Detail: TaiList must not be present since ONSSAI is not supportedCause: UNSUPPORTED_FEATURE • Detail: TaiList must not be present since ONSSAI is not supported
Database error	500	Cause: DB_ERROR • Detail: Failed to store data due to database error: <db exception message>Cause: DB_QUERY_ERROR • Detail: Failed to store data, failed to execute DB queryCause: DB_CONNECTIVITY_ERROR • Detail: Failed to store data due to DB connection error: <db exception message>
Invalid supported features value	400	Cause: OPTIONAL_IE_INCORRECT • Detail: Optional parameter incorrect • Reason: Optional parameter incorrect: supportedFeatures
Unsupported features value	400	Cause: UNSUPPORTED_FEATURE • Detail: Unsupported feature requested: [unsupported feature list]
Negotiation feature not enabled	400	Cause: OPTIONAL_IE_INCORRECT • Detail: Feature negotiation is not enabled on NSSF

Table 7-5 NsAvailability PATCH

Scenarios	Error Code	Updated Error Response
Mandatory params missing	400	Cause: MANDATORY_IE_MISSING • Detail: Mandatory parameter missing • Reason: Mandatory parameter missing: path
Invalid mandatory params value	400	Cause: MANDATORY_IE_INCORRECT • Detail: Mandatory parameter incorrect • Reason: Mandatory parameter incorrect: TAC is incorrect, the value is not in correct format (param name with valid message)
Data not found from DB	404	Cause: RESOURCE_NOT_FOUND • Detail: AMF data not found for amfId: <id>

Table 7-5 (Cont.) NsAvailability PATCH

Scenarios	Error Code	Updated Error Response
Patch document missing in request	400	Cause: MANDATORY_IE_MISSING • Detail: Patch document/List of patch items must not be empty
Incorrect JSON format	400	Cause: INVALID_INPUT_DATA • Detail: Invalid JSON format
Incorrect patch requested	400	Cause: INVALID_INPUT_DATA • Detail: Could not apply patch: Invalid patch requested
JSON parsing exception while applying patch	400	Cause: INVALID_INPUT_DATA • Detail: Could not apply patch: Invalid patch requested
Incorrect path	400	Cause: MANDATORY_IE_INCORRECT • Detail: Could not apply patch: Invalid path
Unsupported PLMN received	403	Cause: PLMN_NOT_SUPPORTED • Detail: Unsupported PLMN(s) received: <unsupported PLMN details>
Unconfigured S-NSSAI received	403	Cause: SNSSAI_NOT_SUPPORTED • Detail: Unconfigured S-NSSAI(s) found: SNSSAI <details> for TAI: <details>
Unrecognized property in JSON	400	Cause: INVALID_INPUT_DATA • Detail: Unrecognized property detected: <param name>
Payload too large	413	Cause: PAYLOAD_TOO_LARGE • Detail: Request payload size is too large, configured size: <configured size>, requested payload size: <requested size>
Duplicate S-NSSAI in request	400	Cause: INVALID_INPUT_DATA • Detail: Duplicate entries found in supportedSnssaiList for SNSSAI <details> for TAI: <details>
Duplicate TAI in request	400	Cause: INVALID_INPUT_DATA • Detail: Duplicate TAI found in supportedNssaiAvailabilityData for TAI: <details>
Database error	500	Cause: DB_ERROR • Detail: Failed to store data due to database error: <db exception message>Cause: DB_QUERY_ERROR • Detail: Unable to store data, failed to execute DB queryCause: DB_CONNECTIVITY_ERROR • Detail: Unable to store data due to DB connection error: <db exception message>

Table 7-5 (Cont.) NsAvailability PATCH

Scenarios	Error Code	Updated Error Response
Patch operation add/replace without value (missing/null)	400	Cause: MANDATORY_IE_INCORRECT • Detail: Invalid patch request: value must not be null for <operation name> operation

Table 7-6 NsAvailability DELETE

Scenarios	Error Code	Updated Error Response
Data not found from DB	404	Cause: RESOURCE_NOT_FOUND • Detail: AMF data not found for amfId: <id>
Delete with request payload	400	Cause: INVALID_INPUT_DATA • Detail: DELETE request must not contain a body

NsSubscription Error Scenarios

Table 7-7 NsSubscription POST

Scenarios	Error Code	Updated Error Response
Feature negotiation is disabled	400	Cause: OPTIONAL_IE_INCORRECT Detail: Feature negotiation is not enabled from NSSF side
Invalid supported features value	400	Cause: OPTIONAL_IE_INCORRECT Detail: Optional IE incorrectinvalidParams: [{"param":"supportedFeatures","reason":"Optional parameter incorrect: supportedFeatures"}]
Unsupported features value	400	Cause: UNSUPPORTED_FEATURE Detail: Unsupported feature requested
Invalid subscription event	400	Cause: MANDATORY_IE_INCORRECT Detail: Unsupported event received
Duplicate TAI in TaiList	400	Cause: MANDATORY_IE_INCORRECT Detail: Duplicate elements detected in TAI list. Duplicate TAI: {TAI}
Unsupported PLMN received	403	Cause: PLMN_NOT_SUPPORTED Detail: Unsupported PLMN(s) received. PLMN: <unsupported plmn>
Received expiry less than min allowed duration	400	Cause: OPTIONAL_IE_INCORRECT Detail: Requested expiry duration is less than minimum allowed expiry duration
Received expiry less than current time	400	Cause: OPTIONAL_IE_INCORRECT Detail: Requested expiry time is less than current time

Table 7-7 (Cont.) NsSubscription POST

Scenarios	Error Code	Updated Error Response
Error converting subscription data to JSON string	400	Cause: INVALID_INPUT_DATA Detail: Failed to convert event subscription data in JSON object: <exception message>
Empty taiRangeList received	400	Cause: OPTIONAL_IE_INCORRECT Detail: taiRangeList cannot be present when ONSSAI is not supported
Valid taiRangeList received	400	Cause: OPTIONAL_IE_INCORRECT Detail: taiRangeList cannot be present when ONSSAI is not supported
Location URL missing/invalid	500	Cause: CONFIGURATION_ERROR Detail: Invalid location/nssfApiRoot URL
Invalid 3gpp-Sbi-Binding header	400	Cause: INVALID_INPUT_DATA Detail: Invalid 3gpp-Sbi-Binding header received. Supported pattern: bl=nf-set; nfset=set<setId>.region<regionId>.amfset.5gc.mnc<mnc>.mcc<mcc> Detail: Invalid setId <setId> received in header Detail: Invalid regionId <regionId> received in header Detail: Invalid mcc <mcc> received in header Detail: Invalid mnc <mnc> received in header
NSSF not part of NF Set, GR disabled	500	Cause: CONFIGURATION_ERROR Detail: Indirect communication is true but NFs and GR are not enabled
Invalid NSSF API Root	500	Cause: CONFIGURATION_ERROR title: INVALID_LOCATION_URL Detail: Invalid location/nssfApiRoot URL
Unsupported PLMN part of binding header	500	Cause: PLMN_NOT_SUPPORTED Detail: Unsupported PLMN(s) received. PLMN: <unsupported plmn>
Failed to compute routing binding header	500	Cause: UNSPECIFIED_NF_FAILURE Detail: Unable to compute 3gpp-Sbi-Binding header for NSSF

Table 7-8 NsSubscription PATCH

Scenarios	Error Code	Updated Error Response
Subscription not found or deleted	404	Cause: RESOURCE_NOT_FOUND Detail: Data not found for subscriptionId: <subId>
SUMOD is disabled	405	Cause: METHOD_NOT_ALLOWED Detail: Request method PATCH is not supported when SUMOD is false
JSON parsing failed	400	Cause: INVALID_INPUT_DATA Detail: Could not apply patch. Invalid patch requested
DB exception	500	Cause: DB_ERROR Detail: Failed to store data due to database error. <db exception message> Cause: DB_QUERY_ERROR Detail: Failed to store data, failed to execute DB query Cause: DB_CONNECTIVITY_ERROR Detail: Failed to store data due to DB connection error. <db exception message>
Replication down and Patch/Delete received for another site	500	Cause: REPLICATION_CHANNEL_BROKEN Detail: Replication channel is down. Cannot apply patch
Empty patch document received	400	Cause: MANDATORY_IE_MISSING Detail: Patch document/List of patch items must not be empty
Unsupported patch operation received	400	Cause: MANDATORY_IE_INCORRECT Detail: Mandatory parameter incorrectinvalidParams: [{"param":"op","reason":"Mandatory parameter incorrect: must be a valid HTTP PATCH operation"}]

Table 7-9 Generic Exceptions List

Scenarios	Error Code	Updated Error Response
DB exceptions	500	Cause: DB_ERROR Detail: Failed to store data due to database error. <db exception message> Cause: DB_QUERY_ERROR Detail: Failed to store data, failed to execute DB query Cause: DB_CONNECTIVITY_ERROR Detail: Failed to store data due to DB connection error. <db exception message>
Missing param	400	Cause: MANDATORY_IE_MISSING Cause: OPTIONAL_IE_MISSING
Incorrect param	400	Cause: MANDATORY_IE_INCORRECT Cause: OPTIONAL_IE_INCORRECT
Unexpected exceptions	500	Cause: INTERNAL_SERVER_EXCEPTION Detail: <localized error message>