Oracle® Communications
Network Slice Selection Function (NSSF) Cloud Native User's Guide

Release 1.3
F31726-01
May 2020

Oracle Communications Network Slice Selection Function (NSSF) Cloud Native User's Guide, Release 1.3

F31726-01

Contents

1 My Oracle Support

Locate Product Documentation on the Oracle Help Center Site 1-1

2 What's New in This Guide

3 Overview

NSSF Architecture 3-2

NSSF Supported Features 3-3

References 3-10

Acronyms and Terminology 3-10

4 NSSF Supported Services

Network Slice Selection Service 4-1

NSSAI Availability Service 4-4

5 Configuring NSSF Using REST APIs

Managed Objects 5-1

Configure NSSF using REST APIs 5-8

6 NSSF Metrics

7 NSSF KPIs

8 NSSF Alerts

NSSF Alert Configuration 8-1

9 Troubleshooting Information

A HTTP Response Codes

B Open API Specification

List of Figures

3-1	Network Slice Selection Function Architecture Diagram	3-2
4-1	Initial Register	4-2
4-2	PDU Session Establishment	4-3
4-3	UE-Config-Update	4-4
4-4	Update the S-NSSAIs the AMF supports per TA	4-5
4-5	Create a Subscription	4-5
4-6	Unsubscribe a Subscription	4-6
4-7	Update the AMF with any S-NSSAI restricted per TA	4-6
4-8	Delete the NSSAI Availability Information at NSSF	4-7

List of Tables

3-1	Acronyms	3-10
5-1	NSI Profile - Parameters	5-1
5-2	Supported REST APIs - NSI Profile	5-1
5-3	NSS Rule Parameters	5-2
5-4	Supported REST APIs - NSS Rule	5-2
5-5	AMF Resolution - Parameters	5-3
5-6	Supported REST APIS - AMF Resolution	5-3
5-7	Configured NSSAI - Parameters	5-3
5-8	Supported REST APIs - Configured NSSAI	5-4
5-9	TimeProfile - Parameters	5-4
5-10	Supported REST APIs - Time Profile	5-5
5-11	Auth NSSAI - Parameters	5-5
5-12	Supported REST APIs - Auth NSSAI	5-5
5-13	Behavior	5-6
5-14	TargetAmfSet	5-6
5-15	CandidateAmf	5-6
5-16	Timespan	5-6
5-17	Plmnid	5-7
5-18	Snssai	5-7
5-19	Grant	5-7
5-20	Access Type	5-7
5-21	DayofWeek	5-7
7-1	NSSF KPIs	7-1
8-1	NSSF Alert Details	8-1
A-1	HTTP Response Codes	A-1

1 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.

Locate Product Documentation on the Oracle Help Center Site

Oracle Communications customer documentation is available on the web at the Oracle Help Center site, http://docs.oracle.com. You do not have to register to access these documents. Viewing these files requires Adobe Acrobat Reader, which can be downloaded at http://www.adobe.com.

1. Access the Oracle Help Center site at http://docs.oracle.com.
2. Click the **Industries** link to display the **Industries Documentation** page.
3. Click **Oracle Communications**, select your product and respective release.
   A list of the documentation set for the selected product and release displays.
4. To download a file to your location, right-click the **PDF** link, select **Save target as** (or similar command based on your browser), and save to a local folder.

2 What's New in This Guide

New and Updated Features in Release 1.3:

- **Open Authorization (OAuth)** authorization.
- **Rate Limiting** to control the incoming and outgoing traffic to or from a network.
- **Hypertext Transfer Protocol Secure (https)** is used for secure communication over the network.
- **Ingress Gateway** and **Egress Gateway** to define an entry points and exit points for all incoming and exit traffic to flow through.

3 Overview

This document provides information on how to use the Oracle Communications Network Slice Selection Function (OCNSSF) in the cloud native 5G core network.

Network slices enables the users to provide customized networks with different functionality (Example: mobility), performance requirements (Example: latency, availability, reliability etc). Network slices may differ for supported features and network function optimisations. 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 exactly the same features but for different groups of User Equipments (UE). As these instances deliver a different committed service because they are dedicated to a customer, in which case such network slices may have different S-NSSAIs with the same slice or service type but different slice differentiators. OCNSSF fulfills the requirement for determining the individual network function pertaining to a slice. This section includes information about the role of OCNSSF in the 5G Service based architecture.

Network Slice Selection Function is a functional element that supports the following functionalities:

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

NSSF is responsible for providing the following information as and when queried by the AMF:

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

OCNSSF supports the above functions through the following NSSF services:

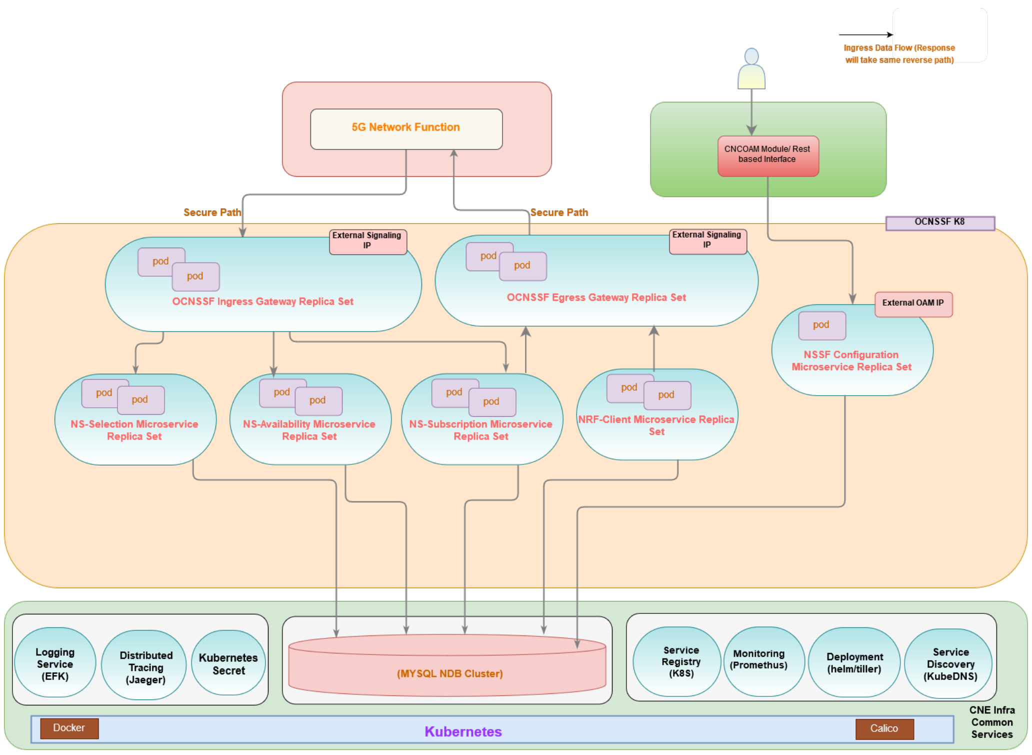
- **Network Slice Selection service** (*Nnssf_NSSelection*): This service is used by an NF Service Consumer (AMF) to retrieve the information related to network slice. Network Slice Selection Service enables Network Slice selection in the serving Home Public Land Mobile Network (HPLMN).

- **NS-Availability Service** (*Nnssf_NSAvailability*): This service is used by an NF Service Consumer (AMF) to update the S-NSSAI(s) the AMF supports on a per TA basis on the NSSF. Also to notify any change in status, on a per TA basis, of the SNSSAIs available per TA (unrestricted) and the restricted SNSSAI(s) per PLMN in that TA in the serving PLMN of the UE.

NSSF Architecture

NSSF comprises of various microservices deployed in Kubernetes based Cloud Native Environment (CNE, example: OCCNE). Some common services like logs or metrics data collection, analysis and graphs or charts visualization, etc. is provided by the environment. The microservices integrates with them and provide them necessary data. The following diagram describes the overall architecture of the NSSF:

Figure 3-1 Network Slice Selection Function Architecture Diagram


Ingress Data Flow (Response will take same reverse path)
5G Network Function
CNCOAM Module/ Rest based Interface
Secure Path
Secure Path
OCNSSF K8
External Signaling IP
pod
pod
OCNSSF Ingress Gateway Replica Set
External Signaling IP
pod
pod
OCNSSF Egress Gateway Replica Set
External OAM IP
pod
NSSF Configuration Microservice Replica Set
pod
pod
NS-Selection Microservice Replica Set
pod
pod
NS-Availability Microservice Replica Set
pod
pod
NS-Subscription Microservice Replica Set
pod
pod
NRF-Client Microservice Replica Set
Logging Service (EFK)
Distributed Tracing (Jaeger)
Kubernetes Secret
(MYSQL NDB Cluster)
Service Registry (K8S)
Monitoring (Promethus)
Deployment (helm/tiller)
Service Discovery (KubeDNS)
Docker
Kubernetes
Calico
CNE Infra Common Services

The architecture has the following components:

NS Selection Microservice

This microservice receives all NS-Selection requests and provides network slice information.

NS Availability Microservice

This service supports the NS-Availability service of NSSF and stores subscriptions and AMF data.

NS Subscription Microservice

This microservice sends notifications based on subscribed events through NS Availability. Notifications are sent to subscribed AMFs to signify changes in authorization state with respect to S-NSSAIs on TAI as per 3GPP TS 29.531.

NS Configuration Microservice

This microservice is responsible for configuring policy rules. This microservice implements a REST messaging server that receives configuration HTTP messages, validates and stores the configuration in the database.

NRF Client Microservice

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

- NRF Registration and Heartbeat: First the Registration profile is configured using helm. Then the Performance service calculates load and capacity of NF. NS Registration requests load and capacity from performance service and send it to NRF with heartbeat.
- NRF Subscription: NSSF subscribes to NRF for AMF based on Target AMF Set and Region ID for Registration and Deregistration and load update.

OCNSSF Ingress Gateway Microservice

This microservice is an entry point for accessing OCNSSF supported service operations and provides the functionality of OAuth validator.

OCNSSF Egress Gateway Microservice

This microservice is responsible to route OCNSSF initiated egress messages to other NFs.

NSSF Supported Features

This section explains the NSSF supported features.

1. OAuth

- OCNSSF supports OAuth 2.0 Access Token based authorization for NF to NF authorization.
- OCNSSF can perform the task of OAuth validator for call scenarios to NS-Selection service and NS-Availability service.
- OCNSSF can act like an OAuth client for Notification messages towards AMF.

Steps to Enable OAuth in NSSF

Prerequisites to enable OAuth

- There must be an OAuth token generator for OCNSSF default token provided is NSSF.
- Generate Kubernetes secret using NRF Public key as per section in OCNSSF installation guide.
- NSSF must have Public Key of NRF.

  - Public Key should be in the format "`{nrfInstanceId}_{SigningAlgorithm}.pem`" where `nrfInstanceId` is Instance Id of NRF and SigningAlgorithm can have following values:

```
ES256: ECDSA using P-256 and SHA-256
ES384: ECDSA using P-384 and SHA-384
ES512: ECDSA using P-521 and SHA-512
RS256: RSASSA-PKCS-v1_5 using SHA-256
RS384: RSASSA-PKCS-v1_5 using SHA-384
RS512: RSASSA-PKCS-v1_5 using SHA-512
PS256: RSASSA-PSS using SHA-256 and MGF1 with SHA-256
PS384: RSASSA-PSS using SHA-384 and MGF1 with SHA-384
PS512: RSASSA-PSS using SHA-512 and MGF1 with SHA-512
```

  - Store all `.pem`files in a secret in ocnssf namespace.
- NSSF must register with all services over which OAuth validation might be supported.

OCNSSF as OAuth Validator

OCNSSF performs the following tasks after receiving a request when OAuth is enabled:

1. NSSF ensures the integrity of the token by verifying the signature using NRF's public key.
2. If integrity check is successful, NSSF verifies the claims in the token as follows:
   a. **NF-ID match**: NSSF ensures that the nf-id in claim is its self nf-id.
   b. **NF-Type match**: NSSF validates that the target nf-type is NSSF.
   c. **Token expiry Validation**: Checks the difference between current time and validity time is less than helm parameter `ingress_gateway.allowedClockSkewSeconds`.

   Sample configuration at OCNSSF to enable OAuth validator functionality:

```
ingress-gateway:
  # NFType of service producer.Mandatory Parameter
  nfType: NSSF
  # NF InsataneeId of service producer.Mandatory Parameter
  nfInstanceId: fe7d992b-0541-4c7d-ab84-c6d70b1bae31
  # Comma-seperated list of services hosted by service
producer.Mandatory Parameter
  producerScope: ns-selection,ns-availability
  # set this value if clock on the parsing NF(producer) is not
perfectly in sync with the clock on the NF(consumer) that created the
JWT.
  # Default value is 0.
  allowedClockSkewSeconds: 1000
  # Name of the secret which stores the public key(s) of NRF. Creation
Of Secret is described in Oauth Validator module below.
  nrfPublicKeyKubeSecret: ocnssf-auth-secret
  # Namespace of the NRF publicKey Secret.
  nrfPublicKeyKubeNamespace: ocnssf
  # Values can be "strict" or "relaxed".
  # "strict" means that incoming request without "Authorization"(Access
Token) header will be rejected.
  # "relaxed" means that if incoming request contains "Authorization"
header, it will be validated.If incoming request does not contain
  # "Authorization" header, validation will be ignored. Default value
is "strict"
  validationType: strict
```

OCNSSF as OAuth client

OCNSSF performs following tasks before sending a notification when OAuth is enabled:

1. OCNSSF sends `nnrf-accesstoken` GET with nf-type as AMF and nf-id as AMF-ID which is stored at NSSF during subscription and request to OCNRF.
2. OCNSSF stores the token in cache and reuses the same token till token expires.
3. OCNSSF adds authentication header using the token provided by NRF to send notification message to AMF.

Sample configuration at OCNSSF to enable OAuth client functionality:

```
 egress-gateway:
   # OAuth token provider in OCNSSF case this is NRF ,NRF's ${HOSTNAME}:
{PORT}.
   nrfAuthority: 10.75.181.00:8080
   # NFType of service consumer.Mandatory Parameter
   nfType: AMF
   # NF InstanceId of Service Consumer.Mandatory Parameter
   nfInstanceId: fe7d992b-0541-4c7d-ab84-c6d70b1b01b1
   # Flag to enable or disable oauth client. If not defined, Default value
'false' will be injected.
   oauthClientEnabled: true
```

2. HTTPS

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)

Steps to Enable HTTPS in OCNSSF

Certificate Creation

To create certificate user must have the following files:

- ECDSA private key and CA signed certificate of OCNRF (if initial algorithm is ES256)
- RSA private key and CA signed certificate of OCNRF (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. We will break the handshake up into 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 (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 will be done using a symmetric algorithm, the exact nature of which was already 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/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/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 plaintext 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.

OCNSSF 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
```

3. Rate Limiting

Rate limiting for Ingress and Egress Messages
OCNSSF uses `Bucket4j` which uses Token Bucket Algorithm to enable rate limiting. With the token-bucket algorithm, user has 3 configuration points.

The token bucket algorithm has following concepts:

bucketCapacity: The maximum number of token the bucket can hold.

duration: The amount of time between the refills.

refillRate: The number of tokens that are added to the bucket during a refill.

(where duration: in seconds (M) ,burstCapacity: (C) ,refillRate: (N))

- N tokens are added to the bucket every M seconds.
- The bucket can hold at the most C tokens. If a token arrives when the bucket is full, it is discarded.

Ingress Rate Limiting

To avoid unexpected behavior and DOS attacks ,OCNSSF allows user to enable rate limiting in ingress messages. OCNSSF allows user to configure a cap on max number of incoming messages at a given duration. User has an option to configure a max cap on number of ingress request per service.

Steps to Enable Ingress Rate Limiting
OCNSSF allows at the `max of {burstCapasity}/ {refillRate}` number of messages in a duration signified by parameter {duration}.

To enable ingress rate limiting at OCNSSF `ingress_gateway.rateLimiting.enabled` must be set to true.

Global Ingress Rate Limiting

When `globalIngressRateLimiting.enabled` is set to true then rate limiting is applied for all ingress messages.

Route Based Rate Limiting

OCNSSF provides option to configure route based rate limiting and method based rate limiting which enables NSSF to throttle messages per Service per method.

In the below example OCNSSF allows 80 GET requests on NS-Selection service for every 2 seconds.

Sample ingress rate limiting configuration:

```
#Rate limiting configuration
rateLimiting:
  enabled: true
routeRateLimiting:
  enabled: true
# Global rate limiting configuration
globalIngressRateLimiting:
  enabled: true
  duration: 2 # in seconds
  burstCapacity: 100
  refillRate: 1

routesConfig:
- id: nsselection_mapping
  uri: http://ocnssf-nsselection:5745
  path: /nnssf-nsselection/**
  order: 1
#Route level limiting configuration enabled for NS-Selection
  methodRateLimiting: # specify the list of methods u have to rate limit
  - method: GET
    burstCapacity: 80
    refillRate: 1
    duration: 2
#Route level limiting configuration not enabled for NS-Availability
- id: availability_mapping
  uri: http://ocnssf-nsavailability:5745
  path: /nnssf-nssaiavailability/**
  order: 2
- id: nsconfig_mapping
  uri: http://ocnssf-nsconfig:5755
  path: /nnssf-configuration/**
  order: 3
```

Egress Rate Limiting

OCNSSF sends notification messages to AMF based on configuration change of Supported SNSSAI/s in a TAI. Notification messages can be throttled by operator by enabling egress message rate limiting.

Steps to Enable Egress Rate Limiting
To enable rate limiting `egress-gateway.notificationRateLimit.enabled` must be set to true.

As per the below example, OCNSSF has a max cap on 200 notifications per second:

```
egress-gateway:
  notificationRateLimit:
    enabled: false
    duration: 1
    bucketCapacity: 200
    refillRate: 1
```

References

Network Slice Selection Function (NSSF) Cloud Native Installation Guide

Acronyms and Terminology

The following table provides information about the acronyms used in the document:

Table 3-1 Acronyms

Field	Description
5G-AN	5G Access Network
5GC	5G Core Network
5G-GUTI	5G Globally Unique Temporary Identifier
5QI	5G QoS Identifier
5G-S-TMSI	5G S-Temporary Mobile Subscription Identifier
5GS	5G System
5G-EIR	5G-Equipment Identity Register
(R)AN	(Radio) Access Network
AMF	Access and Mobility Management Function
AUSF	Authentication Server Function
CAPIF	Common API Framework for 3GPP northbound APIs
HTTPS	Hypertext Transfer Protocol Secure
NEF	Network Exposure Function
NF	Network Function
NRF	Network Repository Function
NSI ID	Network Slice Instance Identifier
NSSAI	Network Slice Selection Assistance Information
NSSF	Network Slice Selection Function
Network Slice	A logical network that provides specific network capabilities and network characteristics .
Network Slice instance	A set of Network Function instances and the required resources (Example: compute, storage and networking resources) which form a deployed Network Slice.
NF service	A functionality exposed by a NF through a service based interface and consumed by other authorized NFs.
NSSP	Network Slice Selection Policy
PEI	Permanent Equipment Identifier
PCF	Policy Control Function
PLMN	Public Land Mobile Network
QFI	QoS Flow Identifier
QoE	Quality of Experience
Requested NSSAI	NSSAI provided by the UE to the Serving PLMN during registration.

Table 3-1 (Cont.) Acronyms

Field	Description
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.
Configured NSSAI	NSSAI provisioned in the UE applicable to one or more PLMNs.
SEPP	Security Edge Protection Proxy
SBA	Service Based Architecture
SBI	Service Based Interface
SSC	Session and Service Continuity
SSCMSP	Session and Service Continuity Mode Selection Policy
SST	Slice/Service type
SD	Slice Differentiator
SMF	Session Management Function
SMSF	Short Message Service Function
S-NSSAI	Single Network Slice Selection Assistance Information
TA	Tracking Area
TAC	Tracking Area Code
TAI	Tracking Area Indentifier
UDM	Unified Data Management
UDR	Unified Data Repository
UDSF	Unstructured Data Storage Function
UE	User Equipment

4

NSSF Supported Services

This section includes information about the service supported by NSSF.

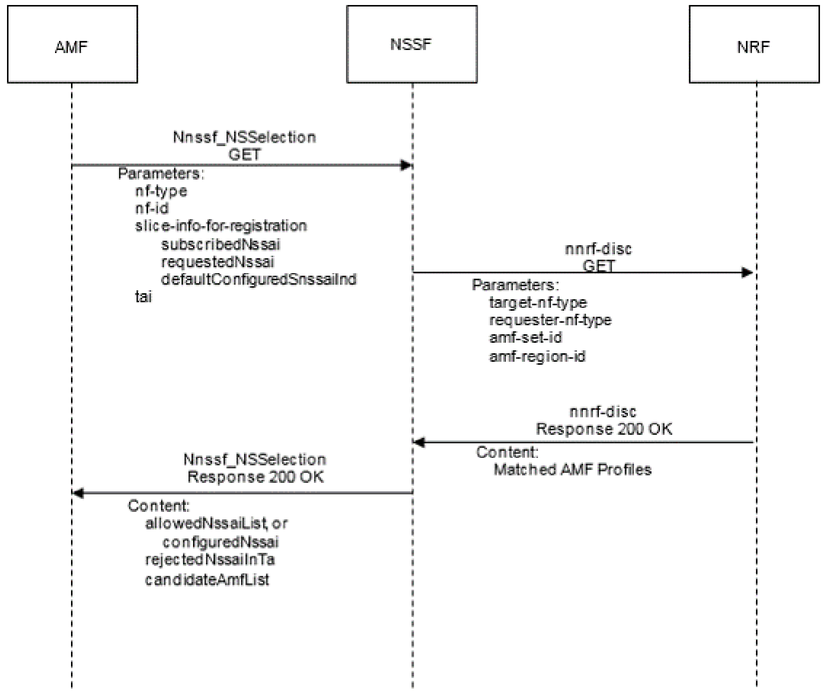
Network Slice Selection Service

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

- **Initial Register:** When the NSSF is able to find authorized network slice information for the requested network slice selection information, the response body includes a payload body containing at least the Allowed NSSAI, target AMF Set or the list of candidate AMF(s).
- **PDU Session Establishment:** When NSSF receives PDU-Session establishment request from NF consumer, NSSF determines network slice which can serve the requested SNSSAI, based on user configured policies, and responds with URL of NRF which manages to the Slice and/or Slice ID of the matching Network Slice computed.
- **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).

1. **Initial Register**
   Following diagram illustrates the procedure of Initial Register:

Figure 4-1 Initial Register



The following is performed for Initial Register:

- The AMF sends a GET request to the NSSF.

  The AMF request must include:

  – Requested NSSAI
  – the Mapping of Requested NSSAI to Configured NSSAI for the HPLMN
  – the Subscribed S-NSSAIs (with an indication if marked as default S-NSSAI)
  – any Allowed NSSAI

  The query parameters may also contain:

  – 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 this information, local configuration and other locally available information including RAN capabilities in 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.

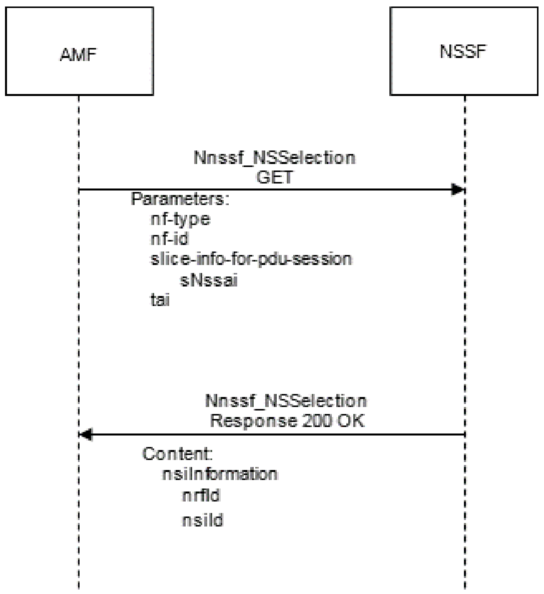
  – It determines the target AMF set to be used to serve the UE or based on configuration, the list of candidate AMF(s), possibly after querying the NRF.
  – It determines the Allowed NSSAI(s) for the applicable Access Type(s), taking also into account the availability of the Network Slice instances that 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). The 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. The 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.

2. **PDU Session Establishment**

   The PDU Session Establishment in a Network Slice to a DN allows data transmission in a Network Slice. A PDU Session is associated to an S-NSSAI and a DNN. Following diagram illustrates the procedure of PDU Session Establishment:

Figure 4-2 PDU Session Establishment



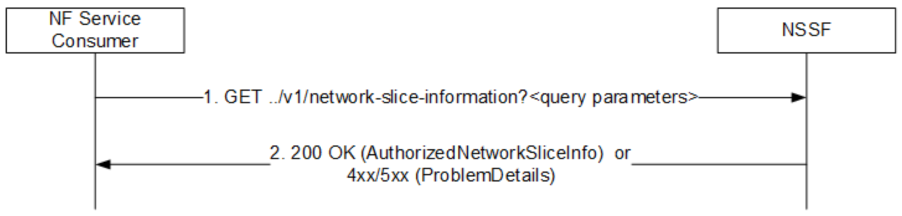
The following is performed for PDU Session Establishment:

- 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 location information.
- The NSSF determines and returns the appropriate NRF to be used to select NFs/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 CN provides to the (R)AN the S-NSSAI corresponding to this Network Slice instance to enable the R(AN) to perform access specific functions.

3. **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). Following diagram illustrates the procedure of UE-Config-Update:

Figure 4-3 UE-Config-Update


NF Service Consumer
NSSF
1. GET ../v1/network-slice-information?<query parameters>
2. 200 OK (AuthorizedNetworkSliceInfo) or 4xx/5xx (ProblemDetails)

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.
- NSSF responds with 200 OK with empty AuthorizedNetworkSliceInfo in case there is no match found.
- NSSF responds with error code in case of incorrect parameter validation.

NSSAI Availability Service

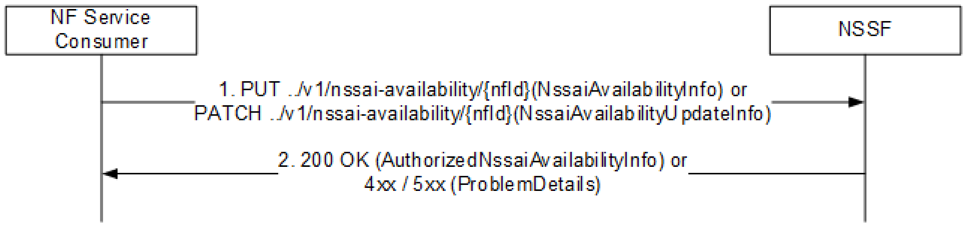
The NSSAI Availability service is identified by the service operation name, `Nnssf_NSSAIAvailability`. For the Nnssf_NSSAIAvailability service the following service operations are defined:

- **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 of 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 4-4 Update the S-NSSAIs the AMF supports per TA

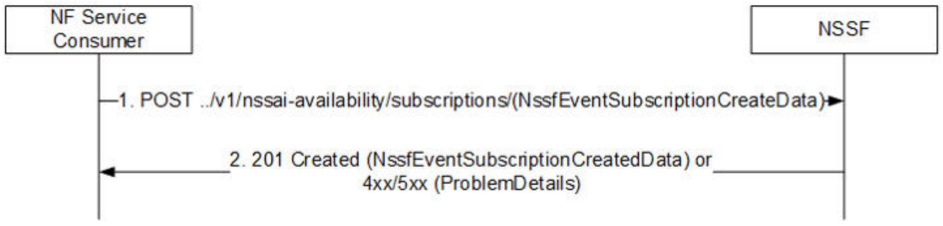

NF Service Consumer
NSSF
1. PUT ../v1/nssai-availability/{nfId}(NssaiAvailabilityInfo) or
PATCH ../v1/nssai-availability/{nfId}(NssaiAvailabilityUpdateInfo)
2. 200 OK (AuthorizedNssaiAvailabilityInfo) or
4xx / 5xx (ProblemDetails)

   - The NF service consumer (Example: AMF) sends a HTTP PUT message to NSSF with NSSAI availability information, identified by {nfId}, with NssaiAvalabilityInfo as body. Body of message contains a list of S-NSSAIs supported by AMF on a per TA basis.
   - On receiving a PUT /PATCH message, NSSF stores/updates the list in the session database.
   - Supports HTTP PATCH for NS-Availability Update
   - The NSSF authorizes the list based on NSSAI Auth rules and responds with the list of allowed S-NSSAIs for that AMF on a per TAI basis as per the request.

2. **Subscribe Service Operation**

   The Subscribe operation is used by AMF to subscribe to a notification of any changes in status of the NSSAI availability information (example: S-NSSAIs available per TA and the restricted S-NSSAI(s) per PLMN in that TA in the serving PLMN of the UE) upon this is updated by another AMF.

   Figure 4-5 Create a Subscription

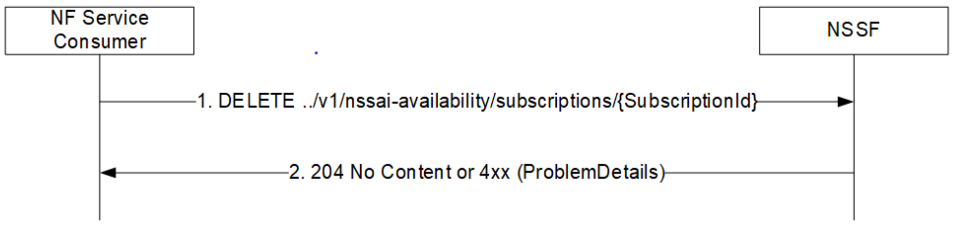

NF Service Consumer
NSSF
1. POST ../v1/nssai-availability/subscriptions/(NssfEventSubscriptionCreateData)
2. 201 Created (NssfEventSubscriptionCreatedData) or
4xx/5xx (ProblemDetails)

- AMF sends a POST request to NSSF with notification URL and a list of TAIs as JASON body.
- NSS stores the subscription request and responds with the list of allowed S-NSSAI/s per TAI for each TAI in the request. NSSF also returns a subscription-id and expiry (duration up to which NSSF ends notifications for any change in the status of Grant of S-NSSAI for subscribed TAI/s).

3. **Unsubscribe Service Operation**

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

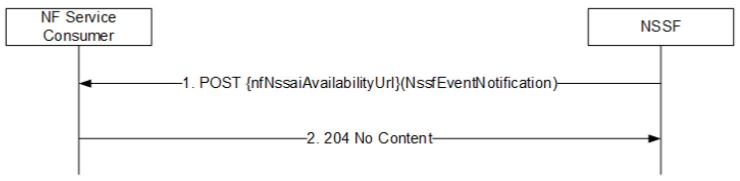
Figure 4-6 Unsubscribe a Subscription


NF Service Consumer
NSSF
1. DELETE ../v1/nssai-availability/subscriptions/{SubscriptionId}
2. 204 No Content or 4xx (ProblemDetails)

- AMF sends a Delete request to NSSF with subscription-id.
- NSSF checks for active subscription with the id and if found, deletes the subscription. NSSF responds with 204.

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 4-7 Update the AMF with any S-NSSAI restricted per TA

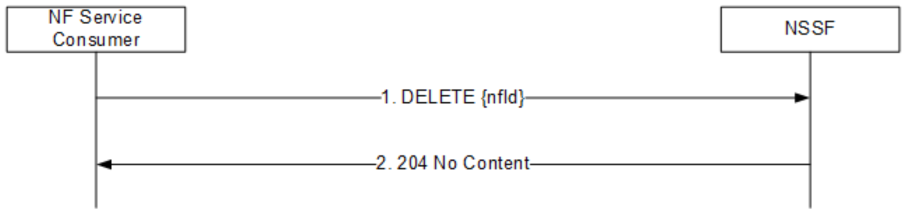


- NSSF sends notification to subscribed AMF when one or more following conditions are true:
  - There is change at Grant rules on S-NSSAI corresponding to one or more of TAIs subscribed by AMF.
  - An S-NSSAI has been added or deleted for one or more of TAIs subscribed by AMF.

5. **Delete Service Operation**

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

Figure 4-8 Delete the NSSAI Availability Information at NSSF



- The NF service consumer (example: AMF) sends a DELETE request to NSSF with {nfId}.
- The NSSF searches in session database for the NS-Availability data corresponding to nfId and deletes.

5

Configuring NSSF Using REST APIs

Managed Objects

The following NSSF managed objects can be configured using REST APIs :

NSI Profile

The NSI Profile managed object enables customer to configure Network Slice Instance profile. This allows customer to create an Network Slice, by providing a name, id, NRF URL corresponding to the slice and list of Target AMF sets which support this slice.

Table 5-1 NSI Profile - Parameters

Field Name	Type	Description (With Default Values)
name	String	Network Slice Instance Profile Name.
nrfUri	String	URI of the Network Resource Function
nsiId	String	Network Slice Intance Identifier
targetAmfSets	array (TargetAmfSet)	array of TargetAmfSet (Refer primitive data type section)
nrfNfMgtUri	String	Management URI of Network Resource Function
nrfNfMgtUri	String	Access Token URI of Network Resource Function

Customer can configure NSI Profiles by following the information provided in the table below. The supported operations are **POST**, **GET**, **DELETE**, and **PUT**. The following table provides information about the REST APIs supported by the NSI Profile managed object:

Table 5-2 Supported REST APIs - NSI Profile

Resource Name	URI	Data Type	HTTP Method	Description
NSI Profiles	/nnssf-configuration/v1/nsiprofiles	array(NssfNsiProfile)	POST	Create a network slice instance profile
			GET	Read all network slice instance profiles
NSI Profile	/nnssf-configuration/v1/nsiprofiles/{name}	NsiProfile	GET	Read a network slice instance profile
			DELETE	Delete a network slice instance profile

Table 5-2 (Cont.) Supported REST APIs - NSI Profile

Resource Name	URI	Data Type	HTTP Method	Description
			PUT	Update a network slice instance profile

NSS Rule

The NSS Rule managed object enables customer to configure policy rules, NSS Rule allows customer to allow/reject/associate a Network slice based on NSSAI(SST and SD) , PLMN(MCC and MNC) ,TAC , AMF_ID. Operator can configure salience value to prioritize one rule over other.

Table 5-3 NSS Rule Parameters

Field Name	Type	Description (With Default Values)
name	String	Network Slice Selection Rule Name
amfId	String	AMF Identifier
plmnId	String	Public Land Mobile Network ID (MCC:MNC)
tac	String	Tracking Area Code
snssai	Snssai	Single Network Slice Selection Assistance Information
salience	Integer	Order of importance, higher salience, more important
behaviour	Behaviour	Behaviour of the parameter

Customer can configure NSS Rules by following the information provided in the table below. The supported operations are **POST**, **GET**, **DELETE**, and **PUT**. The following table provides information about the REST APIs supported by the NSS Rule managed object:

Table 5-4 Supported REST APIs - NSS Rule

Resource Name	URI	Data Type	HTTP Method	Description
NSS Rules	/nnssf-configuration/v1/nssrules	array(NssfNssRule)	POST	Create a network slice selection rule
			GET	Read all network slice selection rules
NSS Rule	/nnssf-configuration/v1/nssrules/{name}	NssRule	GET	Read a network slice selection rule
			DELETE	Delete a network slice selection rule
			PUT	Update a network slice selection rule

AMF Resolution

The AMF Resolution managed object enables customer to configure mapping of list of candidate AMFs to a pair Target AMF set ID and Region ID. This enables operator to give static candidate AMF list. This configuration is used in cases where customer has disabled discovery service with NRF.

Table 5-5 AMF Resolution - Parameters

Field Name	Type	Description (With Default Values)
regionId	Integer	Region ID of the target AMF list
setId	Integer	Set ID of the target AMF list
candidateAmfList	array(candidateAmf)	Refer the primitive data type section

Customer can configure AMF Resolution by following the information provided in the table below. The supported operations are **POST**, **GET**, **DELETE**, and **PUT**. The following table provides information about the REST APIs supported by the AMF Resolution managed object:

Table 5-6 Supported REST APIS - AMF Resolution

Resource Name	URI	Data Type	HTTP Method	Description
AMF Resolutions	/nnssf-configuration/v1/amfresolutions	array(NssfAmfResolution)	POST	Create a AMF resolution
			GET	Read all AMF resolutions
AMF Resolution	/nnssf-configuration/v1/amfresoltuions/{region_id}[:{set_id}[:{instance_id}]]	NssfAmfResolution	GET	Read a AMF resolution
			DELETE	Delete a AMF resolution
			PUT	Update a AMF resolution

Configured NSSAI

The Configured NSSAI managed object enables customer to configure default NSSAI based on one or more of the following parameters PLMN, TAC and AMF-ID. This enables operator to configure default behavior when none of the rules match and UE has set default indication flag to true.

Table 5-7 Configured NSSAI - Parameters

Field Name	Type	Description (With Default Values)
amfId	Integer	AMF Identifier
plmnid	plmn	Public Land Mobile Network ID (MCC:MNC)
tac	string	Tracking Area Code

Table 5-7 (Cont.) Configured NSSAI - Parameters

Field Name	Type	Description (With Default Values)
salience	Integer	Order of importance, higher salience, more important
snssai	array(Snssai)	Refer to primitive datatype section

Customer can configure Configured NSSAI by following the information provided in the table below. The supported operations are **POST**, **GET**, **DELETE**, and **PUT**. The following table provides information about the REST APIs supported by the Configured NSSAI managed object:

Table 5-8 Supported REST APIs - Configured NSSAI

Resource Name	URI	Data Type	HTTP Method	Description
Configured NSSAIs	/nnssf-configuration/v1/configurednssais	array(NssfConfiguredNssai)	POST	Create a configured NSSAI
			GET	Read all configured NSSAIs
Configured NSSAI	/nnssf-configuration/v1/configurednssais/{amf_id}:{mcc}:{mnc}[:{tac}[:{sst}:{sd}]]]	NssfConfiguredNssai	GET	Read a configured NSSAI
			DELETE	Delete a configured NSSAI

Time Profile

The Time Profile managed object enables customer to configure time/date based slice selection policies. This allows customer to create a Time Profile and associate it to a network slice when creating a NSS Rule managed object.

Table 5-9 TimeProfile - Parameters

Field Name	Type	Description
name	String	Time Profile Name
startDate	Date	Date in the format of yy-mm-dd
endDate	Date	Date in the format of yy-mm-dd
daysOfWeek	array(Daysofweek)	Refer enumeration section
timespans	array(TimeSpan)	Refer primitive section

Customer can configure Time Profiles by following the information provided in the table below. The supported operations are **POST**, **GET**, **DELETE**, and **PUT**. The following table provides information about the REST APIs supported by the Time Profile managed object.

Table 5-10 Supported REST APIs - Time Profile

Resource Name	URI	Data Type	HTTP Method	Description
Time Profiles	/nnssf-configuration/v1/timeprofiles	array(TimeProfile)	POST	Create a time profile
			GET	Read all time profiles
Time Profile	/nnssf-configuration/v1/timeprofiles	TimeProfile	GET	Read a time profile
			DELETE	Delete a time profile
			PUT	Update a time profile

Auth NSSAI

The Auth NSSAI managed object enables customer to configure network slice authentication rules by configuring Grant status (Allowed/Rejected_PLMN, Rejected_TAC) for S-Nssai on a per TAI basis.

Table 5-11 Auth NSSAI - Parameters

Field Name	Type	Description (With Default Values)
name	String	Network Slice Authentication Rule Name
plmnId	plmnid	Public Land Mobile Network ID (MCC:MNC)
tac	string	Tracking Area Code
snssai	Snssai	Single Network Slice Selection Assistance Information
grant	Grant	Whether the requested s-NSSAI is allowed or restricted

Customer can configure Auth NSSAI by following the information provided in the table below. The supported operations are **POST**, **GET**, and **DELETE**. The following table provides information about the REST APIs supported by the Auth NSSAI managed object.

Table 5-12 Supported REST APIs - Auth NSSAI

Resource Name	URI	Data Type	HTTP Method	Description
AuthNSSAIs	/nnssf-configuration/v1/authNSSAI	array(AuthNSSAI)	POST	Create a AuthNSSAI Profile
			GET	Read all AuthNSSAI Profiles

Table 5-12 (Cont.) Supported REST APIs - Auth NSSAI

Resource Name	URI	Data Type	HTTP Method	Description
AuthNSSAI	/nnssf-configuration/v1/authNSSAI / {mcc}:{mnc}[:{tac}[:{sst}:{sd}]]	AuthNSSAI	GET	Read a AuthNSSAI Profile
			DELETE	Delete a AuthNSSAI Profile

For a sample Open API Specification, refer Open API Specification.

Primitive Tables

Behavior

Table 5-13 Behavior

Attribute	Datatype	Description
accessType	AccessType	Refer Enumeration section
nsiProfiles	array(NsiProfileMap)	Array of NsiProfile Map

TargetAmfSet

Table 5-14 TargetAmfSet

Attribute	Datatype	Description
regionId	string	region id of TargetAmfSet
setId	string	set id of TargetAmfSet
setFqdn	string	FQDN of TargetAmfSet

CandidateAmf

Table 5-15 CandidateAmf

Attribute	Datatype	Description
fqdn	string	FQDN of targetamfset
instanceId	string	Instance id of Amf

Timespan

Table 5-16 Timespan

Attribute	Datatype	Description
startTime	time	start time in hh:mm:ss
endTime	time	end time in hh:mm:ss

Plmnid

Table 5-17 Plmnid

Attribute	Datatype	Description
mcc	string	Mobile Country Code
mnc	string	Mobile Network Code

Snssai

Table 5-18 Snssai

Attribute	Datatype	Description
sst	integer	Slice /Service Type
sd	string	Slice Differentiator

Enumerations

Grant

Table 5-19 Grant

Value	Description
"ALLOWED"	Allowed signifies SNSSAI is allowed in TAI
"REJECTED_IN_TA"	S-NSSAI is not allowed for Tracking Area
"REJECTED_IN_PLMN	S-NSSAI is not allowed for PLMN

Access Type

Table 5-20 Access Type

Value	Description
"3GPP_ACCESS"	Specifies 5G network
"NON_3GPP_ACCESS"	Specifies non 5G network

DayofWeek

Table 5-21 DayofWeek

Value	Description
"MONDAY"	Monday, day of the week
"TUESDAY"	Tuesday, day of the week
"WEDNESDAY"	Wednesday, day of the week
"THURSDAY"	Thursday, day of the week
"FRIDAY"	Friday, day of the week
"SATURDAY"	Saturday, day of the week
"SUNDAY"	Sunday, day of the week

Configure NSSF using REST APIs

Before configuring NSSF using REST APIs, ensure that the NSSF is installed. For information on how to install NSSF, refer *Network Selection Slice Function Installation Guide*.

To Configure NSSF using REST APIs:

1. **Configure the NSI-Profile managed object:**
   NSI-Profile consists of network slice name and ID and NRF-ID ,Target AMF lists which are associated to the slice.

   - Request_Type: POST
   - URL: *http://{apiRoot}/nnssf-configuration/v1/nsiprofiles*
   - Body: Refer to Sample NSI-Profile-Body section for sample message/s and OpenAPI for schema.

   REST message sample - NSI Profiles

```
http://host:port/nnssf-configuration/v1/nsiprofiles
POST
Content-Type: application/json
BODY
{
    "name": "NSI001",
    "nrfUri": "https://nrf.slice11.oracle.com/nnrf-disc/v1",
    "nsiId": "SLICE1",
    "nrfNfMgtUri":"https://nrf.slice11.oracle.com/nnrf-nfm/v1",
    "nrfAccessTokenUri":"https://nrf.slice11.oracle.com/oauth2/token",
    "targetAmfSets":
    [
        {
            "regionId": "01",
            "setId": "001",
            "setFqdn": "set001.region01.amfset.
5gc.mnc311.mcc282.3gppnetwork.org"
        },
        {
            "regionId": "01",
            "setId": "002",
            "setFqdn": "set002.region01.amfset.
5gc.mnc311.mcc282.3gppnetwork.org"
        }
    ]
}

POST
Content-Type: application/json
BODY
{
    "name": "NSI002",
    "nrfUri": "https://nrf.slice2.oracle.com/nnrf-disc/v1",
    "nsiId": "SLICE2",
    "nrfNfMgtUri":"https://nrf.slice2.oracle.com/nnrf-nfm/v1",
    "nrfAccessTokenUri":"https://nrf.slice2.oracle.com/oauth2/token",
    "targetAmfSets":
    [
        {
            "regionId": "01",
            "setId": "001",
            "setFqdn": "set001.region01.amfset.
5gc.mnc311.mcc282.3gppnetwork.org"
        },
        {
            "regionId": "02",
            "setId": "002",
            "setFqdn": "set002.region01.amfset.
5gc.mnc311.mcc282.3gppnetwork.org"
        }
    ]
}
```

2. **Configure the Time Profile managed object:**
   NSI-Profile consists of network slice name and ID and NRF-ID ,Target AMF lists which are associated to the slice.
   - Request_Type: POST
   - URL: *http://host:port/nnssf-configuration/v1/timeprofiles*
   - Body: Refer to Sample TimeProfile-Body section for sample message/s and OpenApi for schema.

   REST message sample - Time Profiles

```
http://host:port/nnssf-configuration/v1/timeprofiles
POST
Content-Type: application/json
BODY
{
    "name": "WEEKDAY-BUSY",
    "startDate": "2020-01-01",    "endDate": "2020-12-01",
"daysOfWeek":
    [
        "MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY", "FRIDAY"
    ]
    "timeSpans":
    [
        {
            "startTime": "09:00:00",
            "endTime": "12:00:00"
        },
        {
            "startTime": "16:00:00",
            "endTime": "21:00:00"
        }
    ]
}
```

3. **Configure the Nssai Auth managed object:**

- Request_Type: POST
- URL: *http://localhost:5755/nnssf-configuration/v1/nssaiauth/*
- Body: Refer to Sample Nssai Auth -Body section for sample message/s and OpenApi for schema.

REST message sample - Nssai Auth

```
http://localhost:5755/nnssf-configuration/v1/nssaiauth/
POST
{
 "name": "NSSAI-ATH-1",
 "plmnId":
 {
 "mcc": "311",
 "mnc": "282"
 },
 "tac": "100001",
 "snssai":
 {
 "sst": "1",
 "sd": "EABB01"
 },
 "grant": "ALLOWED"
}
```

4. **Configure the NSS Rule managed object:**
NSS Rules are policy rules which enable operator to ALLOW/REJECT a request for Network Slice Selection request and If allowed then map to a Network Slice.

- Request_Type: POST
- URL: *http://{apiRoot}/nnssf-configuration/v1/nssrules*
- Body: Refer to Sample NSS-Rule -Body section for sample message/s and OpenApi for schema.

REST message sample - NSS Rules

```
http://host:port/nnssf-configuration/v1/nssrules
POST
Content-Type: application/json
BODY
{
    "name": "NSSRULE01",
    "amfId": "1",
    "plmnId":
    {
        "mcc": "311",
        "mnc": "282",
    },
    "tac": "100001",
    "snssai":
    {
        "sst": "1",
        "sd": "EABB01"
    },
    "salience": "0",
    "behavior":
    {
        "accessType": "3GPP_ACCESS",
        "nsiProfiles":
        [
            {
                "name": "NSI001",
                "timeProfile": "WEEKDAY-BUSY",
                "salience": 1
            },
            {
                "name": "NSI002",
                "salience": 0
            }
        ]
    }
}
```

5. **Configure the Configured NSSAI managed object:**
   Configured NSSAI enables customer to configure default configures NSSAI based on one or more of the following parameters PLMN, TAC, AMF-ID .
   - Request_Type: POST
   - URL: *http://{apiRoot}/nnssf-configuration/v1/configuredsnssais*
   - Body: Refer to Sample Configured-NSSAI-Body section for sample message/s and OpenApi for schema.

   REST message sample - Configured S-NSSAIs

```
http://host:port/nnssf-configuration/v1/configuredsnssais
POST
Content-Type: application/json
BODY
{
    "plmn":
    {
        "mcc": "311",
        "mnc": "282",
    },
    "tac": "100001"",
    "salience": 0
    "nssai":
    [
        {
            "sst": 1,
            "sd": "EABB01""
        }
    ]
}
```

6. **Configure the AMF Resolution managed object:**
   AMF Resolution enables customer to configure mapping candidate AMF list to a Target AMF set ID and Region ID.

- Request_Type: POST
- URL: *http: //{apiRoot}/nnssf-configuration/v1/amfresolutions*
- Body: Refer to Sample AMF Resolution-Body section for sample messages and Open API for schema.

REST message sample - AMF Resolutions

```
http://host:port/nnssf-configuration/v1/amfresolutions
POST
Content-Type: application/json
BODY
{
    "regionId": "01",
    "setId": "001",
    "candidateAmfList":
    [
        {
            "fqdn": "pt01.set001.region01.amfset.
5gc.mnc311.mcc282.3gppnetwork.org",
            "instanceId": "9faf1bbc-6e4a-4454-a507-aef01a101a03"
        },
        {
            "fqdn": "pt02.set001.region01.amfset.
5gc.mnc311.mcc282.3gppnetwork.org",
            "instanceId": "9faf1bbc-6e4a-4454-a507-aef01a101a04"
        }
    ]
}
```

6
NSSF Metrics

The following are NSSF Metrics:

Success Measurements

Tag	Dimensions	Description	Microservice
nsselection_rx_total	AMF Instance Id, Message Type	Count of request messages received by NSSF for the Nnssf_NSSelection service	NS-Selection
nsselection_success_tx_total	AMF Instance Id, Message Type	Count of success response messages sent by NSSF for requests for the Nnssf_NSSelection service	NS-Selection
nssaiavailability_rx_total	Operation	Count of request messages received by NSSF for the Nnssf_NSSAIAvailability service	NS-Availability
nssaiavailability_success_tx_total	Operation	Count of success response messages sent by NSSF for requests for the Nnssf_NSSAIAvailability service	NS-Availability
nsselection_policy_match	AMF Instance Id, Message Type Policy Rule Name	Count of policy matches found during processing of request messages for the Nnssf_NSSelection service	NS-Selection
nsselection_time_match	AMF Instance Id, Message Type, Time Profile Name	Count of time profile matches found during processing of request messages for the Nnssf_NSSelection service	NS-Selection
nsselection_nsi_selected	AMF Instance Id, Message Type, NSI Profile Name	Count of Network Slice Instances selected during processing of request messages for the Nnssf_NSSelection service	NS-Selection
nsselection_nrf_disc	None	Count of NRF discoveries performed during processing of request messages for the Nnssf_NSSelection service	NS-Selection
nsselection_nrf_disc_success	None	Count of successful discovery results received from NRF during processing of request messages for the Nnssf_NSSelection service	NS-Selection
nssaiavailability_tx_total	Subscription- Id	Count of notification messages sent by NSSF as part of Nnssf_NSSAIAvailability service	NS-Subscription
nssaiavailability_success_rx_total	Subscription- Id	Count of success notification response messages received by NSSF for requests for the Nnssf_NSSAIAvailability service	NS-Subscription

Error Measurements

Tag	Dimensions	Description	Micro-service
nssf_configuration_database_read_error	None	Count of errors encountered when trying to read the configuration database	NS-Selection
nssf_configuration_database_write_error	None	Count of errors encountered when trying to write to the configuration database	NS-Config
nssf_state_data_read_error	None	Count of errors encountered when trying to read the state database	NS-Selection
nssf_state_data_write_error	None	Count of errors encountered when trying to write to the state database	NS-Availability
nsselection_nrf_disc_failure	None	Count of errors encountered when trying to reach the NRF's discovery service	NS-Selection
nsselection_policy_not_found	AMF Instance Id, Message Type	Count of request messages that did not find a configured policy	NS-Selection
nsselection_nrf_retrys	Method	Count of retry attempts sent to NRF	NRF-Client
nssaiavailability_subscription_failure	Operation	Count of subscribe requests rejected by NSSF	NS-Availability
nssaiavailability_notification_failure	Subscription - Id	Count of failure notification response messages receivedby NSSF for requests for the Nnssf_NSSAIAvailability service	NS-Subscription

7 NSSF KPIs

The following are the NSSF KPIs:

Table 7-1 NSSF KPIs

KPI Name	KPI Details	Metric Used	Service Operation	Response Code
OCNSSF Ingress Request	Rate of HTTP requestes recieved at OCNRF Ingress Gateway	oc_ingressgateway_http_requests	All	Not Applicable
OCNSSF NsSelection Initial Registration success rate	Percentage of NS-Selection Initial registration messages with success response	sum(nsselection_success_tx_total{message_type=\"registartion\"})/ sum(nsselection_rx_total{ message_type= \"registartion\"}))*100"	NS-Selection	200
OCNSSF NsSelection PDU establishment success rate	Percentage of NS-Selection PDU establishment messages with success response	sum(nsselection_success_tx_total{message_type= \"pdu_session\"})/ sum(nsselection_rx_total{ message_type= \"pdu_session\"}))*100"	NS-Selection	200
OCNSSF NsSelection UE-Config Update success rate	Percentage of NS-Selection UE-Config Update messages with success response	sum(nsselection_success_tx_total{message_type= \"ue_config_update\"})/ sum(nsselection_rx_total{ message_type= \"ue_config_update \"}))*100",	NS-Selection	200
OCNSSF NsAvailability PUT success rate	Percentage of NS-Availability UPDATE PUT messages with success response	sum(nssaiavailability_success_tx_total{message_type=\"availability_update\"} {method=\"PUT"})/ sum(nssaiavailability_rx_total{message_type= \"availability_update\"} {method=\"PUT"}))*100"	NS-Availability Update	200
OCNSSF NsAvailability PATCH success rate	Percentage of NS-Availability UPDATE PATCH messages with success response	sum(nssaiavailability_success_tx_total{message_type=\"availability_update\"} {method=\"PATCH"})/ sum(nssaiavailability_rx_total{message_type= \"availability_update\"} {method=\"PATCH"}))*100"	NS-Availability Update	200

Table 7-1 (Cont.) NSSF KPIs

KPI Name	KPI Details	Metric Used	Service Operation	Response Code
OCNSSF NsAvailability Delete success rate	Percentage of NS-Availability Delete messages with success response	sum(nssaiavailability_success_tx_total{message_type=\"availability_update\"}{method=\"DELETE"})/ sum(nssaiavailability_rx_total{message_type= \"availability_update\"}{method= \"DELETE"}))*100""	NS-Availability Delete	204
OCNSSF NsAvailability Subscribe success rate	Percentage of NS-Availability Subscribe messages with success response	sum(nssaiavailability_success_tx_total{message_type=\"availability_subscribe \"}{method=\"POST"})/ sum(nssaiavailability_rx_total{message_type= \"availability_subscribe\"}{method=\"POST"}))*100"	NS-Availability Subscribe	201
OCNSSF NsAvailability Unsubscribe success rate	Percentage of NS-Availability Unsubscribe messages with success response	sum(nssaiavailability_success_tx_total{message_type=\"availability_subscribe \"}{method=\"DELETE"})/ sum(nssaiavailability_rx_total{message_type= \"availability_subscribe\"}{method= \"DELETE"}))*100"	NS-Availability Unsubscribe	204
4xx Responses (NS-Selection)	Rate of 4xx response for NS-Selection	sum(increase(oc_ingressgateway_http_responses{Status=~"4.* ",Uri=~".*nnssf-nsselection.*",Method="GET"}[5m]))	NS-Selection	4xx
4xx Responses (NS-Availability)	Rate of 4xx response for NS-Availability	sum(increase(oc_ingressgateway_http_responses{Status=~"4.* ",Uri=~".*nnssf-nsavailability.*",Method="GET"}[5m]))	NS-Availability	4xx
5xx Responses (NS-Selection)	Rate of 5xx response for NS-Selection	sum(increase(oc_ingressgateway_http_responses{Status=~"5.* ",Uri=~".*nnssf-nsselection.*",Method="GET"}[5m])	NS-Selection	5xx
5xx Responses (NS-Availability)	Rate of 5xx response for NS-Availability	sum(increase(oc_ingressgateway_http_responses{Status=~"4.* ",Uri=~".*nnssf-nsavailability.*",Method="GET"}[5m]))	NS-Availability	5xx

8

NSSF Alerts

This section includes information about alerts for OCNSSF.

Table 8-1 NSSF Alert Details

Name	Severity	Condition	Description
ocnssfPolicyNotFoundWarn	Warning	rate(nsselection_policy_match) <= 95%	Rate of messages that did not find a matching policy is above warning threshold (Threshold: <>, Current: <>)
ocnssfPolicyNotFoundMaj	Major	rate(nsselection_policy_match) <= 85%	Rate of messages that did not find a matching policy is above major threshold
ocnssfPolicyNotFoundCrit	Critical	rate(nsselection_policy_match) <= 70%	Rate of messages that did not find a matching policy is above critical threshold
ocnssfNrfDiscFailedWarn	Warning	rate(nsselection_nrf_disc_failure) >= 10%	Rate of failed NRF discovery attempts is above warning threshold
ocnssfNrfDiscFailedWarn	Major	rate(nsselection_nrf_disc_failure) >= 30%	Rate of failed NRF discovery attempts is above major threshold
ocnssfNrfDiscFailedWarn	Critical	rate(nsselection_nrf_disc_failure) >= 50%	Rate of failed NRF discovery attempts is above critical threshold
ocnssfNotificationFailureWarn	Warning	rate(nssaiavailability_notification_failure) >=10	Rate of failed attempts to send Notification to AMF
ocnssfNotificationFailureMaj	Major	rate(nssaiavailability_notification_failure) >=30	Rate of failed attempts to send Notification to AMF
ocnssfNotificationFailureCrit	Critical	rate(nssaiavailability_notification_failure) >=50	Rate of failed attempts to send Notification to AMF

For NSSF Alerts configuration, please refer to *Network Slice Selection Function (NSSF) Cloud Native Installation Guide* .

NSSF Alert Configuration

Follow the steps below for NSSFAlert configuration in Prometheus:

Note:

1. By default Namespace for OCNSSF is `ocnssf` that must be updated as per the deployment.

2. The `OCNSSF-config-1.3.0.0.0.zip` file can be downloaded from OHC. Unzip the `OCNSSF-config-1.3.0.0.0.zip` package after downloading to get `NssfAlertrules-1.3.0.yaml`file.

Procedure

1. Take a backup of current configuration map of Prometheus:
   `kubectl get configmaps _NAME_-server -o yaml -n _Namespace_ > /tmp/tempConfig.yaml`
2. Check and add OCNSSF Alert file name inside Prometheus configuration map:
   `sed -i '/etc\/config\/alertsnrf/d' /tmp/tempConfig.yaml sed -i '/rule_files:/a\ \- /etc/config/alertsnrf' /tmp/tempConfig.yaml`
3. Update configuration map with updated file name of OCNSSF alert file:
   `kubectl replace configmap _NAME_-server -f /tmp/tempConfig.yaml`
4. Add OCNSSF Alert rules in configuration map under file name of OCNSSF alert file:
   `kubectl patch configmap _NAME_-server -n _Namespace_--type merge --patch "$(cat ~/NssfAlertrules.yaml)"`

Note:

The Prometheus server takes an updated configuration map that is automatically reloaded after approximately 60 seconds. Refresh the Prometheus GUI to confirm that the OCNSSF Alerts have been reloaded.

OCNSSF Alert Config Details

Note:

By default the NameSpace is set to ocnssf. Must update it according to the requirement.

Sample

```
apiVersion: v1
data:
  alertsnssf: |
    groups:
    - name: OcnssfAlerts
      rules:
      - alert: ocnssfPolicyNotFoundWarn
        annotations:
                description: 'Policy Not Found Rate is above warning threshold
      i.e. 700 mps (current value is: {{ $value }})'
                summary: 'Policy Not Found Rate is above 70 Percent'
              expr: sum(rate(nsselection_policy_not_found_total[2m])) >= 700 <
      850
              labels:
                severity: Warning
            - alert: ocnssfPolicyNotFoundMaj
              annotations:
                description: 'Policy Not Found Rate is above major threshold
      i.e. 850 mps (current value is: {{ $value }})'
                summary: 'Policy Not Found Rate is above 85 Percent'
              expr: sum(rate(nsselection_policy_not_found_total[2m])) >= 850 <
      950
              labels:
                severity: Major
            - alert: ocnssfPolicyNotFoundCrit
              annotations:
                description: 'Policy Not Found Rate is above critical threshold
      i.e. 950 mps (current value is: {{ $value }})'
                summary: 'Policy Not Found Rate is above 95 Percent'
              expr: sum(rate(nsselection_policy_not_found_total[2m])) >= 950
              labels:
                severity: Critical
            - alert: ocnssfNrfDiscFailedWarn
              annotations:
                description: 'Rate of failed NRF discovery attempts is above
      warning threshold i.e. 500 mps (current value is {{ $value }})'
                summary: 'Failed NRF discovery Rate attempts is above 10 Percent'
              expr: sum(rate(nsselection_nrf_disc_failure_total[2m])) >= 100 <
      300
              labels:
                severity: Warning
            - alert: ocnssfNrfDiscFailedMaj
              annotations:
                description: 'Rate of failed NRF discovery attempts is above
      major threshold i.e. 700 mps (current value is {{ $value }})'
                summary: 'Failed NRF discovery Rate attempts is above 30 Percent'
              expr: sum(rate(nsselection_nrf_disc_failure_total[2m])) >= 300 <
      500
              labels:
                severity: Major
            - alert: ocnssfNrfDiscFailedCrit
              annotations:
                description: 'Rate of failed NRF discovery attempts is above
      critical threshold i.e. 900 mps (current value is {{ $value }})'
                summary: 'Failed NRF discovery Rate attempts is above 50 Percent'
              expr: sum(rate(nsselection_nrf_disc_failure_total[2m])) >= 500
              labels:
                severity: Critical
```

9 Troubleshooting Information

If you experience issues while using NSSF, refer to the topics below:

Verifying Environment

1. Check if kubectl is installed and working as expected.
2. Check if `kubectl version` command works: This must display the versions of client and server.
3. Check if `$ kubectl create namespace test` command works.
4. Check if `kubectl delete namespace test` command works.
5. Check if Helm is installed and working as expected.
6. Check if `helm version` command works: This must display the versions of client and server.

Debugging of NSSF Installation while Installing using helm

1. If the user is getting the following error:
   failed to parse ocnssf-custom-values-1.3.0.yaml: error converting YAML to JSON: yaml. Then
   a. The `ocnssf-custom-values-1.3.0.yaml` may not be created properly.
   b. The tree structure may not be followed.
   c. There may be tab spaces in the file.

   Verify that the `ocnssf-custom-values-1.3.0.yaml` is proper

   Refer *Network Slice Selection Function (NSSF) Cloud Native Installation Guide .*
2. If there is no error, helm installation will be deployed.
   Helm status can be checked using following command :

   ```
   helm status <helm release name>
   ```

Verifying NSSF Installation

To verify whether the NSSF installation is successful or not, check:

1. Verify NSSF specific pods are working as expected by executing following command
   `kubectl get pods -n <ocnssf_namespace>`

   Check whether all the pods are up and running.

   Sample output:

   ```
   NAME                                READY   STATUS   RESTARTS   AGE
   ocnssf-appinfo-55bcc48477-knc8t            1/1   Running    0      317s
   ocnssf-config-server-5d8c7968fc-d8q9z      1/1   Running    0      317s
   ocnssf-egress-595948cd5-d6hw2              1/1   Running    0      317s
   ```

```
ocnssf-ingress-54c8b668c5-nlcbf             1/1   Running    0      317s
ocnssf-nrf-clientservice-7d9c64f566-z9dnw   1/1   Running    0      317s
ocnssf-nsavailability-8646844976-8pljq      1/1   Running    0      317s
ocnssf-nsconfig-5d755b7865-nh4r9            1/1   Running    0      317s
ocnssf-nsdb-585f7bd7d-bwwjd                 1/1   Running    0      317s
ocnssf-nsselection-56674986bf-fmswv         1/1   Running    0      317s
ocnssf-nssubscription-76478c6c84-fdz7f      1/1   Running    0      317s
ocnssf-performance-6d75c7f966-57fgt         1/1   Running    0      317s
```

2. If status of any pod is shown as `ImagePullBackOff` or `ErrImagePull` then it can be due to:
   a. Incorrect ImageName provided in `ocnssf-custom-values-1.3.0.yaml`.
      Then, double check the image name and tags in `ocnssf-custom-values-1.3.0.yaml`.
   b. Docker registry is incorrectly configured.
      Then, check docker registry is properly configured in all master and slave nodes.
3. If RESTARTS count of the pods is continuously increasing, then it can happen due to the following reasons:
   a. MySQL primary and secondary hosts may not be configured properly in `ocnssf-custom-values-1.3.0.yaml`
   b. MySQL servers may not be configured properly according to the pre-installation steps mentioned in *Network Slice Selection Function (NSSF) Cloud Native Installation Guide* .

Debugging General CNE

Execute the command `kubectl get events -n <ocnssf_namespace>`to get all the events related to a particular namespace.

Collecting NSSF Logs

The following commands must be executed to get the logs from nssf specific pods:

1. Fetch the list of all pods by executing `kubectl get pods -n <ocnssf_namespace>`
2. Collect the logs from the pod and redirect to file by executing `kubectl logs <pod_name> -n <ocnssf_namespace> > <Log File>`

Example:

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

A
HTTP Response Codes

The following are HTTP Response Codes:

Table A-1 HTTP Response Codes

Service	Service Operation	HTTP Request Method	HTTP Response Code	Condition
Nnssf_NsSelection	Initial Registration	Get	400 (Bad Request)	• All semantic, syntax errors leads to this response code. • This response code indicates that the request is not valid according to protocol such as invalid json and patch items.
			401 Unauthorized	Missing Authentication
			405 (Method Not Allowed)	Method not implemented for URI
			500 (Internal Error)	• Db operation error • Memory not available
			403 (FORBIDDEN)	• Authentication failure • If Allowed SNSSAI is not matching • For a particular TAI if there is no SNSSAI in the allowed list • In case of PDU establishment • Requested SNSSAI is not allowed
			200 (OK)	Success Case
Nnssf_NsAvailability	(UPDATE)	PUT	200 (OK)	Success Case
			400 (Bad Request)	• All semantic, syntax errors leads to this response code. • This response code indicates that the request is not valid according to protocol such as invalid json and patch items.
			401 Unauthorized	Missing Authentication
			403 (FORBIDDEN)	When all SNSSAIs for all TAIs are not allowed in PLMN
			405 (Method Not Allowed)	Method not implemented for URI

Table A-1 (Cont.) HTTP Response Codes

Service	Service Operation	HTTP Request Method	HTTP Response Code	Condition
			500 (Internal Error)	Db operation error Memory not available
Nnssf_NsAvailability	(UPDATE)	PATCH	200 (OK)	Success Case
			400 (Bad Request)	• All semantic, syntax errors leads to this response code. • This response code indicates that the request is not valid according to protocol such as invalid json and patch items.
			401 Unauthorized	Missing Authentication
			403 (FORBIDDEN)	When all SNSSAIs for a all TAIs are rejected in PLMN
			405 (Method Not Allowed)	Method not implemented for URI
			404 (Not found)	AMF Availability data not found
			500 (Internal Error)	• Db operation error • Memory not available
Nnssf_NsAvailability	(Subscribe)	POST	201 (CREATED)	Success Case
			401 Unauthorized	Missing Authentication
			400 (Bad Request)	• All semantic, syntax errors leads to this response code. • This response code indicates that the request is not valid according to protocol such as invalid json and patch items. • Expiry duration smaller than Min Expiry duation.
			500 (Internal Error)	500 (Internal Error)
Nnssf_NsAvailability	(Subscribe)	DELETE	204 No Content	Success Case
			401 Unauthorized	Missing Authentication
			404 (Not found)	Subscription-ID not found
			500 (Internal Error)	Db operation error

B

Open API Specification

This appendix provides a sample of Open API specification in NSSF.

Open API 3.0

```
 openapi: 3.0.0
info:
  title: "NSSF-CONFIGURATION"
  version: v0.1
servers:
  - url: 'https://{apiRoot}/'
    variables:
      apiRoot:
        default: nssf
        description: >-
          apiRoot should be mentioned as defined in NSSF configuration
script
paths:
  '/nssf-configuration/v1/nsiprofiles':
    post:
      summary: "Create a network slice instance profile"
      tags:
        - "Create a network slice instance profile"
      requestBody:
        content:
          application/json:    # Media type
            schema:            # Request body contents
              $ref: '#/components/schemas/NssfNsiProfile'
      responses:
        '201' :
          description: Created
        '403' :
          description: Forbidden
        '409' :
          description: Conflict
        '500' :
          description: Internal Server Error
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
    get:
      summary: "Read all network slice instance profiles"
      tags:
        - "Read all network slice instance profiles"
      responses:
        '200' :
          description: OK
          content:
```

```
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/NssfNsiProfile'
      '403' :
        description: Forbidden
      '500' :
        description: Internal Server Error
      '503' :
        description: Service Unavailable
      default:
        description: Unexpected error
'/nssf-configurations/v1/nsiprofiles/{name}':
  get:
    summary: "Read a network slice instance profile"
    tags:
      - "Read a network slice instance profile"
    parameters:
      - name: name
        in: path
        description: "network slice instance profile name"
        required: true
        schema:
          type: string
    responses:
      '200' :
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/NssfNsiProfile'
      '400' :
        description: Bad Request
      '403' :
        description: Forbidden
      '404' :
        description: Not Found
      '405' :
        description: Method Not Allowed
      '409' :
        description: Conflict
      '500' :
        description: Internal Server Error
      '502' :
        description: Bad Gateway
      '503' :
        description: Service Unavailable
      default:
        description: Unexpected error
  delete:
    summary: "Delete a network slice instance profile"
    tags:
      - "Delete a network slice instance profile"
    parameters:
```

```
   - name: name
     in: path
     description: "network slice instance profile name"
     required: true
     schema:
       type: string
  responses:
   '204' :
     description: No Content
   '403' :
     description: Forbidden
   '404' :
     description: No Found
   '500' :
     description: Internal Server Error
   '503' :
     description: Service Unavailable
   default:
     description: Unexpected error
'/nssf-configuration/v1/nssrules':
 post:
  summary: "Create a network slice selection rule"
  tags:
   - "Create a network slice selection rule"
  requestBody:
   content:
     application/json:    # Media type
       schema:            # Request body contents
        $ref: '#/components/schemas/NssfNssRule'
  responses:
   '201' :
     description: Created
   '403' :
     description: Forbidden
   '409' :
     description: Conflict
   '500' :
     description: Internal Server Error
   '503' :
     description: Service Unavailable
   default:
     description: Unexpected error
 get:
  summary: "Read all network slice selection rules"
  tags:
   - "Read all network slice selection rules"
  responses:
   '200' :
     description: OK
     content:
       application/json:
        schema:
          type: array
          items:
            $ref: '#/components/schemas/NssfNssRule'
          '403' :
            description: Forbidden
          '500' :
            description: Internal Server Error
          '503' :
            description: Service Unavailable
          default:
            description: Unexpected error
  '/nssf-configuration/v1/nssrule/{name}':
    get:
      summary: "Read a network slice selection rule"
      tags:
        - "Read a network slice selection rule"
      parameters:
        - name: name
          in: path
          description: "network slice selection rule name"
          required: true
          schema:
            type: string
      responses:
        '200' :
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/NssfNssRule'
        '400' :
          description: Bad Request
        '403' :
          description: Forbidden
        '404' :
          description: Not Found
        '405' :
          description: Method Not Allowed
        '409' :
          description: Conflict
        '500' :
          description: Internal Server Error
        '502' :
          description: Bad Gateway
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
    delete:
      summary: "Delete a network slice selection rule"
      tags:
        - "Delete a network slice selection rule"
      parameters:
        - name: name
          in: path
          description: "network slice selection rule name"
          required: true
          schema:
                type: string
          responses:
            '204' :
              description: No Content
            '403' :
              description: Forbidden
            '404' :
              description: No Found
            '500' :
              description: Internal Server Error
            '503' :
              description: Service Unavailable
            default:
              description: Unexpected error
      '/nssf-configuration/v1/nssaiauth':
        post:
          summary: "Create a network slice authentication rule"
          tags:
            - "Create a network slice authentication rule"
          requestBody:
            content:
              application/json:    # Media type
                schema:            # Request body contents
                  $ref: '#/components/schemas/NssfNssaiAuth'
          responses:
            '201' :
              description: Created
            '403' :
              description: Forbidden
            '409' :
              description: Conflict
            '500' :
              description: Internal Server Error
            '503' :
              description: Service Unavailable
            default:
              description: Unexpected error
        get:
          summary: "Read all network slice authentication rules"
          tags:
            - "Read all network slice authentication rules"
          responses:
            '200' :
              description: OK
              content:
                application/json:
                  schema:
                    type: array
                    items:
                      $ref: '#/components/schemas/NssfNssaiAuth'
            '403' :
              description: Forbidden
            '500' :
              description: Internal Server Error
            '503' :
```

```
          description: Service Unavailable
        default:
          description: Unexpected error
  '/nssf-configuration/v1/timeprofiles':
    post:
      summary: "Create a time profile"
      tags:
        - "Create a time profile"
      requestBody:
        content:
          application/json:    # Media type
            schema:            # Request body contents
              $ref: '#/components/schemas/Nssftimeprofile'
      responses:
        '201' :
          description: Created
        '403' :
          description: Forbidden
        '409' :
          description: Conflict
        '500' :
          description: Internal Server Error
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
    get:
      summary: "Read all time profiles"
      tags:
        - "Read all time profiles"
      responses:
        '200' :
          description: OK
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Nssftimeprofile'
        '403' :
          description: Forbidden
        '500' :
          description: Internal Server Error
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
  '/nssf-configurations/v1/timeprofiles/{name}':
    get:
      summary: "Read a time profile"
      tags:
        - "Read a time profile"
      parameters:
        - name: name
          in: path
          description: "time profile name"
          required: true
          schema:
            type: string
      responses:
        '200' :
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Nssftimeprofile'
        '400' :
          description: Bad Request
        '403' :
          description: Forbidden
        '404' :
          description: Not Found
        '405' :
          description: Method Not Allowed
        '409' :
          description: Conflict
        '500' :
          description: Internal Server Error
        '502' :
          description: Bad Gateway
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
    delete:
      summary: "Delete a time profile"
      tags:
        - "Delete a time profile"
      parameters:
        - name: name
          in: path
          description: "time profile name"
          required: true
          schema:
            type: string
      responses:
        '204' :
          description: No Content
        '403' :
          description: Forbidden
        '404' :
          description: No Found
        '500' :
          description: Internal Server Error
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
  '/nssf-configuration/v1/nssaiAuth/{name}':
    get:
      summary: "Read a network slice authentication rule"
      tags:
        - "Read a network slice authentication rule"
      parameters:
        - name: name
          in: path
          description: "network slice authentication rule name"
          required: true
          schema:
            type: string
      responses:
        '200' :
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/NssfNssaiAuth'
        '400' :
          description: Bad Request
        '403' :
          description: Forbidden
        '404' :
          description: Not Found
        '405' :
          description: Method Not Allowed
        '409' :
          description: Conflict
        '500' :
          description: Internal Server Error
        '502' :
          description: Bad Gateway
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
    delete:
      summary: "Delete a network slice authentication rule"
      tags:
        - "Delete a network slice authentication rule"
      parameters:
        - name: name
          in: path
          description: "network slice authentication rule name"
          required: true
          schema:
            type: string
      responses:
        '204' :
          description: No Content
        '403' :
          description: Forbidden
        '404' :
          description: No Found
        '500' :
          description: Internal Server Error
              '503' :
                description: Service Unavailable
              default:
                description: Unexpected error
        '/nssf-configuration/v1/amfresolutions':
          post:
            summary: "Create a Amf Resolution"
            tags:
              - "Create a Amf Resolution"
            requestBody:
              content:
                application/json:    # Media type
                  schema:            # Request body contents
                    $ref: '#/components/schemas/NssfAmfResolution'
            responses:
              '201' :
                description: Created
              '403' :
                description: Forbidden
              '409' :
                description: Conflict
              '500' :
                description: Internal Server Error
              '503' :
                description: Service Unavailable
              default:
                description: Unexpected error
          get:
            summary: "Read all Amf Resolutions"
            tags:
              - "Read all Amf Resolutions"
            responses:
              '200' :
                description: OK
                content:
                  application/json:
                    schema:
                      type: array
                      items:
                        $ref: '#/components/schemas/NssfAmfResolution'
              '403' :
                description: Forbidden
              '500' :
                description: Internal Server Error
              '503' :
                description: Service Unavailable
              default:
                description: Unexpected error
        '/nssf-configurations/v1/amfresolutions/{region_id}[:{set_id}[:
      {instance_id}]]':
          get:
            summary: "Read a Amf Resolution"
            tags:
              - "Read a Amf Resolution"
            parameters:
```

```
        - name: region_id
          in: path
          description: "Amf Region ID"
          required: true
          schema:
            type: string
        - name: set_id
          in: path
          description: "Amf Set ID"
          required: true
          schema:
            type: string
        - name: instance_id
          in: path
          description: "Amf instance ID"
          required: true
          schema:
            type: string
      responses:
        '200' :
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/NssfAmfResolution'
        '400' :
          description: Bad Request
        '403' :
          description: Forbidden
        '404' :
          description: Not Found
        '405' :
          description: Method Not Allowed
        '409' :
          description: Conflict
        '500' :
          description: Internal Server Error
        '502' :
          description: Bad Gateway
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
    delete:
      summary: "Delete a Amf Resolution"
      tags:
        - "Delete a Amf Resolution"
      parameters:
        - name: region_id
          in: path
          description: "Amf region ID"
          required: true
          schema:
            type: string
        - name: set_id
```

```
          in: path
          description: "Amf set ID"
          required: true
          schema:
            type: string
        - name: instance_id
          in: path
          description: "Amf instance ID"
          required: true
          schema:
            type: string
      responses:
        '204' :
          description: No Content
        '403' :
          description: Forbidden
        '404' :
          description: No Found
        '500' :
          description: Internal Server Error
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
  '/nssf-configuration/v1/configurednssais':
    post:
      summary: "Create a Configured S-NSSAI "
      tags:
        - "Create a Configured S-NSSAI "
      requestBody:
        content:
          application/json:    # Media type
            schema:            # Request body contents
              $ref: '#/components/schemas/NssfConfiguredNssai'
      responses:
        '201' :
          description: Created
        '403' :
          description: Forbidden
        '409' :
          description: Conflict
        '500' :
          description: Internal Server Error
        '503' :
          description: Service Unavailable
        default:
          description: Unexpected error
    get:
      summary: "Read all Configured S-NSSAI "
      tags:
        - "Read all Configured S-NSSAI "
      responses:
        '200' :
          description: OK
          content:
                application/json:
                  schema:
                    type: array
                    items:
                      $ref: '#/components/schemas/NssfConfiguredNssai'
            '403' :
              description: Forbidden
            '500' :
              description: Internal Server Error
            '503' :
              description: Service Unavailable
            default:
              description: Unexpected error
  '/nssf-configurations/v1/configurednssais/{amf_id}:{mcc}:{mnc}[:{tac}[:
{sst}:{sd}]]':
    get:
      summary: "Read a Amf Resolution"
      tags:
        - "Read a Amf Resolution"
      parameters:
        - name: amf_id
          in: path
          description: "Amf ID"
          required: true
          schema:
            type: string
        - name: mcc
          in: path
          description: "Mobile country code"
          required: true
          schema:
            type: string
        - name: mnc
          in: path
          description: "Mobile Network code"
          required: true
          schema:
            type: string
        - name: tac
          in: path
          description: "Tracking Area code"
          required: true
          schema:
            type: string
        - name: sst
          in: path
          description: "Slice service type"
          required: true
          schema:
            type: integer
        - name: sd
          in: path
          description: "Slice descriptor"
          required: true
          schema:
              type: string
              pattern: '^[A-Fa-f0-9]{6}$'
        responses:
          '200' :
            description: OK
            content:
              application/json:
                schema:
                  $ref: '#/components/schemas/NssfConfiguredNssai'
          '400' :
            description: Bad Request
          '403' :
            description: Forbidden
          '404' :
            description: Not Found
          '405' :
            description: Method Not Allowed
          '409' :
            description: Conflict
          '500' :
            description: Internal Server Error
          '502' :
            description: Bad Gateway
          '503' :
            description: Service Unavailable
          default:
            description: Unexpected error
      delete:
        summary: "Delete a Amf Resolution"
        tags:
          - "Delete a Amf Resolution"
        parameters:
          - name: amf_id
            in: path
            description: "Amf ID"
            required: true
            schema:
              type: string
          - name: mcc
            in: path
            description: "Mobile country code"
            required: true
            schema:
              type: string
          - name: mnc
            in: path
            description: "Mobile Network code"
            required: true
            schema:
              type: string
          - name: tac
            in: path
            description: "Tracking Area code"
            required: true
            schema:
                type: string
            - name: sst
              in: path
              description: "Slice service type"
              required: true
              schema:
                type: integer
            - name: sd
              in: path
              description: "Slice descriptor"
              required: true
              schema:
                type: string
                pattern: '^[A-Fa-f0-9]{6}$'
          responses:
            '204' :
              description: No Content
            '403' :
              description: Forbidden
            '404' :
              description: No Found
            '500' :
              description: Internal Server Error
            '503' :
              description: Service Unavailable
            default:
              description: Unexpected error
components:
  schemas:
    NssfNssaiAuth:
      type: object
      properties:
        name:
          type: string
          description:  "Authentication Rule Name"
          minLength: 1
          maxLength: 255
          example: "AUTH-RULE-1"
        plmnId:
          $ref: '#/components/schemas/PlmnId'
        tac:
          type: string
          description: "AMF Identifier"
          minLength: 1
          maxLength: 255
        snssai:
          $ref: '#/components/schemas/Snssai'
        grant:
          type: string
          enum:
            - ALLOWED
            - RESTRICTED
    NssfNsiProfile:
      type: object
      properties:
            name:
              type: string
              description:  "Network Slice Instance Profile Name"
              minLength: 1
              maxLength: 255
              example: "Slice01"
            nrfUri:
              type:  string
              description: "URI of the Network Resource Function"
              minLength: 1
              maxLength: 255
              example: nrf.oracle.com
            nsiId:
              type: string
              description: "Network Slice Intance Identifier"
              minLength: 1
              maxLength: 255
            targetAmfSets:
              type: array
              description: "List of Target AMF Sets mapped to this Network 
Slice Instance"
              items:
                  $ref: '#/components/schemas/NssfTargetAmfSet'
              minItems: 1
          required:
            - name
            - nrfUri
            - targetAmfSets
        NssfTargetAmfSet:
          type: object
          properties:
            regionId:
              type: string
              description:  "Target AMF Region Id"
              minLength: 1
              maxLength: 2
              example: "01"
            setId:
              type: string
              description:  "Target AMF Set Id"
              minLength: 1
              maxLength: 3
              example: "001"
            setFqdn:
              type: string
              description: "Target AMF Set Fqdn"
              pattern: "^(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\\-]*[a-zA-Z0-9])\
\.){2,}([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\\-]*[A-Za-z0-9]){2,}$"
              example: "set001.region01.amfset.
5gc.mnc311.mcc282.3gppnetwork.org"
          required:
            - regionId
            - setId
        NssfNssRule:
          type: object
          properties:
            name:
              type: string
              description:  "Network Slice Selection Rule Name"
              minLength: 1
              maxLength: 255
              example: "NSS-Rule01"
            amfId:
              type:  string
              description: "AMF Identifier"
              minLength: 1
              maxLength: 255
            plmnId:
              $ref: '#/components/schemas/PlmnId'
            tac:
              type: string
              description: "AMF Identifier"
              minLength: 1
              maxLength: 255
            snssai:
              $ref: '#/components/schemas/Snssai'
            salience:
              type: integer
              description: "Order of importance, higher salience, more
important"
              minimum: 0
              maximum: 65535
            behavior:
              $ref: '#/components/schemas/NssfNssRuleBehavior'
          required:
            - name
            - nrfUri
            - snssai
            - behavior
        PlmnId:
          type: object
          properties:
            mcc:
              type: string
              description: "Mobile Country Code"
              minLength: 1
              maxLength: 3
            mnc:
              type: string
              description: "Mobile Network Code"
              minLength: 1
              maxLength: 3
          required:
            - mcc
            - mnc
        Snssai:
          type: object
          properties:
            sst:
              type: integer
          minimum: 0
          maximum: 255
        sd:
          type: string
          pattern: '^[A-Fa-f0-9]{6}$'
      required:
        - sst
    NssfNssRuleBehavior:
      type: object
      properties:
        grant:
          type: string
          enum:
            - ALLOWED
            - RESTRICTED
          description: "Whether the requested S-NSSAI is allowed or
restricted"
        accessType:
          type: string
          enum:
            - 3GPP_ACCESS
            - NON_3GPP_ACCESS
          description: "Access Type in which the grant applies"
        nsiProfiles:
          type: array
          items:
            properties:
              name:
                type: string
                description: "Network Slice Instance profile name"
              salience:
                type: integer
                description: "Order of importance, higher salience, more
important"
            required:
              - name
      required:
        - accessType
    Nssftimeprofile:
      type: object
      properties:
        name:
          type: string
          description:  "Network Slice Instance Profile Name"
          minLength: 1
          maxLength: 255
          example: "TimeProfile01"
        startDate:
          type: string
          description:  "Start Date format yyyy-mm-dd"
          example: "2044-11-01"
        endDate:
          type: string
          description:  "end Date format yyyy-mm-dd"
          example: "2044-11-09"
    daysOfWeek:
      type: array
      description: "List of days on which profile is active"
      items:
        $ref: '#/components/schemas/DaysOfWeek'
    timeSpans:
      type: array
      items:
        properties:
          startTime:
            type: string
            description: "Start time format hh:mm:ss"
          endTime:
            type: string
            description: "end time format hh:mm:ss"
        required:
          - startTime
          - endTime
  required:
    - name
    - startTime
    - endTime
DaysOfWeek:
  description: "Days of Week"
  enum:
            - MONDAY
            - TUESDAY
            - WEDNESDAY
            - THURSDAY
            - FRIDAY
            - SATURDAY
            - SUNDAY
NssfAmfResolution:
  type: object
  properties:
    reqionId:
      type: string
      description:  "Region Id of AMF"
      minLength: 2
      maxLength: 3
      example: "101"
    setId:
      type: string
      description:  "Set Id of AMF"
      minLength: 2
      maxLength: 3
      example: "101"
    candidateAmfList:
      type: array
      items:
        properties:
          fqdn:
            type: string
            description: "AMF FQDN"
          instanceId:
```

```
                    type: string
                    description: "NF isnstance ID of AMF"
                  required:
                    - instanceId
              required:
                - reqionId
                - setId
                - candidateAmfList
            NssfConfiguredNssai:
              type: object
              properties:
                amfId:
                  type:  string
                  description: "AMF Identifier"
                  minLength: 1
                  maxLength: 255
                plmnId:
                  $ref: '#/components/schemas/PlmnId'
                tac:
                  type: string
                  description: "TAC Identifier"
                  minLength: 1
                  maxLength: 255
                nssai:
                  type: array
                  description: "List of Configured S-Nssais"
                  items:
                      $ref: '#/components/schemas/Snssai'
                  minItems: 1
                salience:
                  type: integer
                  description: "Order of importance, higher salience, more
important"
                  minimum: 0
                  maximum: 65535
              required:
                - nssai
```