TEAS
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

B. Wu
D. Dhody
Huawei Technologies
R. Rokui
Ciena
T. Saad
J. Mullooly
Cisco Systems, Inc

16 March 2024

A YANG Data Model for the RFC 9543 Network Slice Service
draft-ietf-teas-ietf-network-slice-nbi-yang-10

Abstract

This document defines a YANG data model for RFC 9543 Network Slice Service. The model can be used in the Network Slice Service interface between a customer and a provider that offers RFC 9543 Network Slice Services.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 17 September 2024.

1. Introduction

~~This document defines a YANG [RFC7950] data model for [RFC9543] Network Slice Service.~~ [RFC9543] discusses common framework and interface for Network Slice using IETF technologies. The Network Slice Services may be referred to as RFC 9543 Network Slice Services. In this document, we simply use the term "Network Slice Service" to refer to this concept.

This document defines a YANG [RFC7950] data model for [RFC9543] Network Slice Service. The Network Slice Service Model (NSSM) can be used in the Network Slice Service Interface exposed by a provider to its customers (including ~~of~~ for provider's internal use) in order to manage (e.g., subscribe, delete, or change) Network Slice Services. The agreed service will then trigger the appropriate Network Slice operation, such as instantiating, modifying, or deleting a Network Slice.

The NSSM focuses on the requirements of a Network Slice Service from the point of view of the customer, not how it is implemented within a provider network. The module is classified as customer service model (Section 2 of [RFC8309]). As discussed in [RFC9543], the mapping between a Network Slice Service and its realization is implementation and deployment specific.

The NSSM is a service model (Section 3.5.1 of [I-D.ietf-netmod-rfc8407bis]).

The NSSM conforms to the Network Management Datastore Architecture (NMDA) [RFC8342].

Editorial Note: (To be removed by RFC Editor)

This document contains several placeholder values that need to be replaced with finalized values at the time of publication. Please apply the following replacements:

* "AAAA" -- the assigned RFC value for this draft both in this draft and in the YANG models under the revision statement.

* The "revision" date in model, in the format XXXX-XX-XX, needs to be updated with the date the draft gets approved.

2. Conventions used in this document

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14, [RFC2119], [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC6241] and are used in this specification:

* client

* configuration data

* state data

This document makes use of the terms defined in [RFC7950].

The tree diagrams used in this document follow the notation defined in [RFC8340].

This document also makes use of the terms defined in [RFC9543]:

* Attachment Circuit (AC): See Section 3.2 of [RFC9543].

* Connectivity Construct: See Sections 3.2 and 4.2.1 of [RFC9543].

* Customer: See Section 3.2 of [RFC9543].

* Customer Higher-level Operation System: See Section 6.3.1 of [RFC9543].

* Service Demarcation Point (SDP): See Sections 3.2 and 5.2 [RFC9543].

In addition, this document defines the following term:

* Connection Group: Refers to one or more connectivity constructs that are grouped for administrative purposes, such as the following:

  - Combine multiple connectivity constructs to support a set of well-known connectivity service types, such as bidirectional unicast service, multipoint-to-point (MP2P) service, or hub-and-spoke service.

  - Assign the same Service Level Objectives (SLOs/ Service Level Expectations (SLEs) policies to multiple connectivity constructs unless SLO/SLE policy is explicitly overridden at the individual connectivity construct level.

  - Share specific SLO limits within multiple connectivity constructs.

2.1. Acronyms

The following acronyms are used in the document:

A2A	Any-to-any
AC	Attachment Circuit
CE	Customer Edge
MTU	Maximum Transmission Unit
NSC	Network Slice Controller
NSSM	Network Slice Service Model
P2P	Point-to-point
P2MP	Point-to-multipoint
PE	Provider Edge
QoS	Quality of Service
SDP	Service Demarcation Point
SLE	Service Level Expectation
SLO	Service Level Objective

3. Network Slice Service Overview

As defined in Section 3.2 of [RFC9543], a Network Slice Service is specified in terms of a set of Service Demarcation Points (SDPs), a set of one or more connectivity constructs between subsets of these SDPs, and a set of Service Level Objectives (SLOs) and Service Level Expectations (SLEs) for each SDP sending to each connectivity construct. A communication type (point- to-point (P2P), point-to-multipoint (P2MP), or any-to-any (A2A)) is specified for each connectivity construct.

The SDPs serve as the Network Slice Service ingress/egress points. An SDP is identified by a unique identifier in the context of a Network Slice Service.

Examples of Network Slice Services that contain only one connectivity construct are shown in Figure 1.

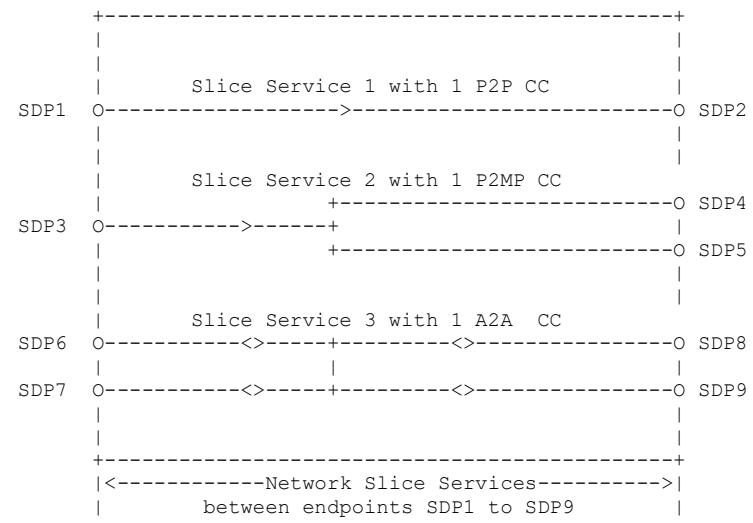


Figure 1: Examples of Network Slice Services of Single Connectivity Construct

An example of Network Slice Services that contain multiple connectivity constructs is shown in Figure 2.

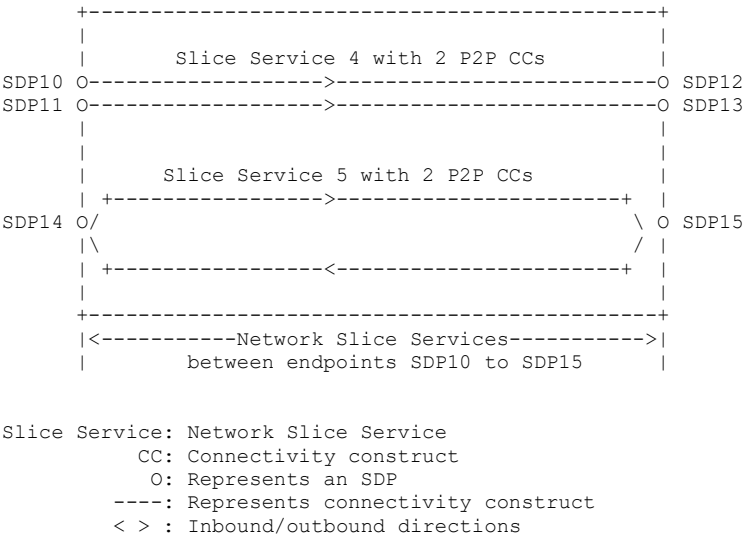


Figure 2: Examples of Network Slice Services of Multiple Connectivity Constructs

As shown in Figure 2, the "Network Slice Service 4" contains two P2P connectivity constructs between the set of SDPs. The "Network Slice Service 5" is a bidirectional unicast service between "SDP14" and "SDP15" that consists of two unidirectional P2P connectivity constructs.

4. Network Slice Service Model (NSSM) Usage

The NSSM can be used by a provider to expose its Network Slice Services, and by a customer to manage its Network Slices Services (e.g., request, delete, or modify). The details about how service requests are handled by ~~the~~ a provider (specifically, a controller), including which network operations are triggered, are internal to the provider. The details of the Network Slices realization are hidden from customers. Readers can refer to [I-D.ietf-teas-5g-ns-ip-mpls] for a realization example.

The Network Slices are applicable to use cases, such as (but not limited to) 5G, network wholesale services, network infrastructure sharing among operators, Network Function Virtualization (NFV) connectivity, and Data Center interconnect. [I-D.ietf-teas-ietf-network-slice-use-cases] provides a more detailed description of the use cases ~~usecases~~ for Network Slices.

~~An~~ A Network Slice Controller (NSC) is an entity that exposes the Network Slice Service Interface to customers to manage Network Slice Services. Typically, an NSC receives requests from its customer-facing interface (e.g., from a management system). During service

creation, this interface can convey data objects that the Network Slice Service customer provides, describing the needed Network Slices Service in terms of SDPs, the associated connectivity constructs, and the service objectives that the customer wishes to be fulfilled. Depending ~~of~~on whether the requirements and authorization checks are successfully met, these service requirements are then translated into technology-specific actions that are implemented in the underlying network(s) using a network-facing interface. The details of how the Network Slices are put into effect are out of scope for this document.

As shown in Figure 3, the NSSM is used by the customer's higher level operation system to communicate with an NSC for life cycle management of Network Slice Services including both enablement and monitoring. For example, in the 5G End-to-end network slicing ~~use-case~~use case the 5G network slice orchestrator acts as the higher layer system to manage the Network Slice Services. The interface is used to support dynamic Network Slice management to facilitate end-to-end 5G network slice services.

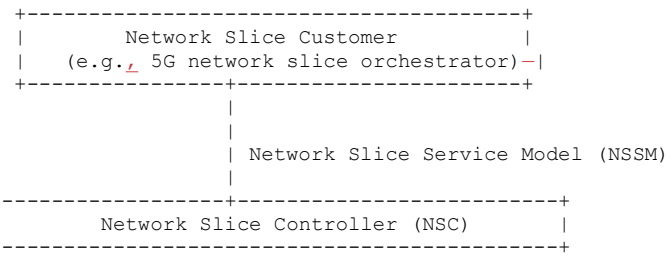


Figure 3: Network Slice Service Reference Architecture

Note: The NSSM can be used recursively (hierarchical mode), i.e., an NSS can map to child NSSes. As described in Section A.5 of [RFC9543], the Network Slice Service can support a recursive composite architecture that allows one layer of Network Slice Services to be used by other layers.

5. Network Slice Service Model (NSSM) Description

The NSSM, "ietf-network-slice-service", includes two main data nodes: "slo-sle-templates" and "slice-service" (~~see~~ Figure 4).

```
module: ietf-network-slice-service
  +--rw network-slice-services
     +--rw slo-sle-templates
     |  +--rw slo-sle-template* [id]
     |        ...
     +--rw slice-service* [id]
        +--rw id                                string
        +--rw description?                      string
        +--rw service-tags
        |     ...
        +--rw (slo-sle-policy)?
        |     ...
```

```
           +--rw compute-only?                     empty
           +--rw status
           |      ...
           +--rw sdps
           |      ...
           +--rw connection-groups
           |      ...
           +--rw custom-topology
                  ...
```

Figure 4: The NSSM Overall Tree Structure

The "slo-sle-templates" container is used by an NSC to maintain a set of common ~~network~~ Network ~~slice~~ Slice SLO and SLE templates that apply to one or several Network Slice Services. Refer to Section 5.1 for further details on the properties of ~~a~~ NSS templates.

The "slice-service" list includes the set of Network Slice Services that are maintained by a provider for a given customer. "slice-service" is the data structure that abstracts the Network Slice Service. Under the "slice-service", the "sdp" list is used to abstract the SDPs. The "connection-group" is used to abstract connectivity constructs between SDPs. Refer to Section 5.2 for further details on the properties of an NSS.

~~Figure 4 describes the overall tree structure of the NSSM.~~

```
~~module: ietf-network-slice-service~~
~~  +--rw network-slice-services~~
~~     +--rw slo-sle-templates~~
~~     |  +--rw slo-sle-template* [id]~~
~~     |        ...~~
~~     +--rw slice-service* [id]~~
~~        +--rw id                                string~~
~~        +--rw description?                      string~~
~~        +--rw service-tags~~
~~        |      ...~~
~~        +--rw (slo-sle-policy)?~~
~~        |      ...~~
~~        +--rw compute-only?                     empty~~
~~        +--rw status~~
~~        |      ...~~
~~        +--rw sdps~~
~~        |      ...~~
~~        +--rw connection-groups~~
~~        |      ...~~
~~        +--rw custom-topology~~
~~               ...~~
```

~~Figure 4: The NSSM Overall Tree Structure~~

5.1. SLO and SLE Templates

The "slo-sle-templates" container (Figure 5) is used by a Network Slice Service provider to define and maintain a set of common Network Slice Service templates that apply to one or several Network Slice Services. The templates are assumed to be known to both the customers and the provider. The exact definition of the templates is deployment specific ~~to each provider~~.

```
+--rw slo-sle-templates
|  +--rw slo-sle-template* [id]
|     +--rw id              string
|     +--rw description?    string
|     +--rw template-ref?   slice-template-ref
|     +--rw slo-policy
|     |  +--rw metric-bound* [metric-type]
|     |  |  +--rw metric-type          identityref
|     |  |  +--rw metric-unit          string
|     |  |  +--rw value-description?   string
|     |  |  +--rw percentile-value?    percentile
|     |  |  +--rw bound?               uint64
|     |  +--rw availability?   identityref
|     |  +--rw mtu?            uint32
|     +--rw sle-policy
|        +--rw security*              identityref
|        +--rw isolation*             identityref
|        +--rw max-occupancy-level?   uint8
|        +--rw path-constraints
|           +--rw service-functions
|           +--rw diversity
|              +--rw diversity-type?
|                      te-types:te-path-disjointness
```

Figure 5: Slo Sle Templates Subtree Structure

The NSSM provides the ~~identifiers of~~ SLO and SLE template identifiers ~~s~~ and the common attributes defined in Section 5.1 of [RFC9543]. Considering that there are many attributes defined and some attributes could vary with service requirements, e.g., bandwidth~~,~~ or latency, standard templates as well as custom "service-slo-sle-policy" are defined. A ~~Customer~~ customer ~~can~~ may choose either a standard template provided by the operator or a custom "service-slo-sle-policy".

Commenté [MB1]: What is meant by "standard template" here?

1. Standard template: The exact definition of the templates is deployment specific ~~to the provider~~. The attributes configuration of a standard template is optional. When specifying attributes, a standard template can use "template-ref" to inherit some attributes of ~~the~~ a predefined standard template and override the specific attributes.

2. Custom "service-slo-sle-policy": More description is provided in Section 5.2.3.

Figure 6 shows an example where two standard network slice templates ~~can be~~are retrieved by the customers.

```
========== NOTE: '\' line wrapping per RFC 8792 ===========

{
  "ietf-network-slice-service:network-slice-services": {
    "slo-sle-templates": {
      "slo-sle-template": [
        {
          "id": "PLATINUM-template",
          "description": "Two-way bandwidth: 1 Gbps,\
                          95th percentile latency 50ms",
          "slo-policy": {
            "metric-bound": [
              {
                "metric-type": "ietf-nss:two-way-delay-percentile",
                "metric-unit": "milliseconds",
                "percentile-value": "95.000",
                "bound": "50"
              }
            ]
          },
          "sle-policy": {
            "isolation": ["ietf-nss:traffic-isolation"]
          }
        },
        {
          "id": "GOLD-template",
          "description": "Two-way bandwidth: 1 Gbps,\
                          maximum latency 100ms",
          "slo-policy": {
            "metric-bound": [
              {
                "metric-type": "ietf-nss:two-way-delay-maximum",
                "metric-unit": "milliseconds",
                "bound": "100"
              }
            ]
          },
          "sle-policy": {
            "isolation": ["ietf-nss:traffic-isolation"]
          }
        }
      ]
    }
  }
}
```

Figure 6 : Example of Template Retrieval

Figure 6 ~~use~~ uses folding as defined in [RFC8792] for long lines.

5.2. Network Slice Services

The "slice-service" is the data structure that abstracts a Network Slice Service. Each "slice-service" is uniquely identified ~~by "id" specified in the context of~~within an NSC by “id”.

Commenté [MB2]: Missing prefix

a mis en forme : Anglais (États-Unis)

a mis en forme : Anglais (États-Unis)

A Network Slice Service has the following main data nodes:

* "description": Provides a textual description of ~~an~~ a Network Slice Service.

* "service-tags": Indicates a management tag (e.g., "customer-name") that is used to correlate the operational information of Customer Higher-level Operation System and Network Slices. It might be used by a Network Slice Service provider to provide additional information to an NSC during the operation of the Network Slices. ~~E.g.~~For example, adding tags with "customer-name" when multiple actual customers use a same Network Slice Service. Another ~~use-case~~use case for "service-tag" might be for a provider to provide additional attributes to an NSC which might be used during the realization of Network Slice Services such as type of services (e.g., use Layer 2 or Layer 3 technology for the realization). These additional attributes can also be used by an NSC for various purposes such as monitoring and assurance of the Network Slice Services where the NSC can issue notifications to the customer system. ~~Note that a~~All these attributes are optional.

* "slo-sle-policy": Defines SLO and SLE policies for the "slice-service". More details are provided in Section 5.2.3.

* "compute-only": Is used to check the feasibility of service before instantiating a Network Slice Service in a network. More details are provided in Section 5.2.6.

* "status": ~~Is used to show~~ indicates ~~the~~ both operational and administrative status of a Network Slice Service. Mismatches between the admin/oper status ~~It~~ can be used as an indicator to detect Network Slice Service anomalies.

* "sdps": Represents a set of SDPs that are involved in the Network Slice Service. More details are provided in Section 5.2.1.

* "connection-groups": Abstracts the connections to the set of SDPs of the Network Slice Service.

* "custom-topology": Represents custom topology constraints for the Network Slice Service. More details are provided in Section 5.2.5

5.2.1. Service Demarcation Points

A Network Slice Service involves two or more SDPs. A Network Slice Service can be modified by adding new "sdp"s.

```
+--rw sdps
   +--rw sdp* [id]
      +--rw id                        string
      +--rw description?              string
      +--rw geo-location
```

```
|     ...
+--rw node-id?                  string
+--rw sdp-ip-address*           inet:ip-address
+--rw tp-ref?                   leafref
+--rw service-match-criteria
|     ...
+--rw incoming-qos-policy
|     ...
+--rw outgoing-qos-policy
|     ...
+--rw sdp-peering
|     ...
+--rw ac-svc-name*              string
+--rw ce-mode?                  boolean
+--rw attachment-circuits
|     ...
+--rw status
|     ...
+--ro sdp-monitoring
      ...
```

Figure 7: SDP Subtree Structure

Section 5.2 of [RFC9543] describes four possible ways in which an SDP may be placed:

* Within ~~the~~ a CE

* Provider-facing ports on ~~the~~ a CE

* Customer-facing ports on ~~the~~ a PE

* Within the PE

Although there are four options, they can be categorized into two categories:
CE-based or PE-based.

In the four options, the Attachment Circuit (AC) may be part of the Network Slice Service or may be external to it. Based on the AC definition in Section 5.2 of [RFC9543], the customer and provider may agree on a per {Network Slice Service, connectivity construct, and SLOs/SLEs} basis to police or shape traffic on the AC in both the ingress (CE to PE) direction and egress (PE to CE) direction, which ensures that the traffic is within the capacity profile that is agreed in a Network Slice Service. Excess traffic is dropped by default, unless specific out-of-profile policies are agreed between the customer and the provider.

To abstract the SDP options and SLOs/SLEs profiles, an SDP has the following characteristics:

* "id": Uniquely identifies the SDP within an NSC. The identifier is a string that allows any encoding for the local administration of the Network Slice Service.

* "geo-location": Indicates SDP location information, which helps the NSC to identify an SDP.

Commenté [MB3]: Now that the AC spec is stable and will be WGLCed in OPSAWG, I suggest we use a clean design here and properly use the references exposed in the ACaaS model.

* "node-id": A reference to the node that hosts the SDP, which helps the NSC to identify an SDP. This document assumes that higher-level systems can obtain the node information, PE and CE, prior to the service requests. For example, Service Attachment Points (SAPs) ~~SAP Network~~ [RFC9408] can obtain PE-related node information. The implementation details are left to the NSC provider.

* "sdp-ip-address": The SDP IP ~~information~~address, which helps the NSC to identify an SDP.

* "tp-ref": A reference to a Termination Point (TP) in the custom topology defined in Section 5.2.5.

* "service-match-criteria": Defines matching policies for the Network Slice Service traffic to apply on a given SDP.

* "incoming-qos-policy" and "outgoing-qos-policy": Sets the incoming and outgoing QoS policies to apply on a given SDP, including QoS policy and specific ingress and egress traffic limits to ensure access security. When applied in the incoming direction, the policy is applicable to the traffic that passes through the AC from the customer network or from another provider's network to the Network Slice. When applied in the outgoing direction, the policy is applied to the traffic from the Network Slice towards the customer network or towards another provider's network. If an SDP has multiple ACs, the "rate-limits" of "attachment-circuit" can be set to an AC specific value, but the rate cannot exceed the "rate-limits" of the SDP. If an SDP only contains a single AC, then the "rate-limits" of "attachment-circuit" is the same with the SDP. The definition of AC refers to Section 5.2 [RFC9543].

* "sdp-peering": Specifies the peers and peering protocols for an SDP to exchange control-plane information, e.g., ~~-~~ Layer 1 signaling protocol or Layer 3 routing protocols, etc. As shown in Figure 8

```
+--rw sdp-peering
|  +--rw peer-sap-id*   string
|  +--rw protocols
```

Figure 8: SDP Peering Subtree Structure

  - "peer-sap-id": Indicates the references to the remote endpoints of attachment circuits. This information can be used for correlation purposes, such as identifying Service Attachment Points (SAPs) defined in [RFC9408], which defines a model of an abstract view of the provider network topology that contains the points from which the services can be attached.

  - "protocols": Serves as an augmentation target. Appendix A gives the example protocols of BGP, static routing, etc.

* "ac-svc-name": Indicates the names of AC services, for association purposes, to refer to the ACs that have been created. When both

"ac-svc-name" and the attributes of "attachment-circuits" are defined, the "ac-svc-name" takes precedence.

* "ce-mode": A flag node that marks the SDP as CE type.

* "attachment-circuits": Specifies the list of ACs by which the service traffic is received. This is an optional SDP attribute. When an SDP has multiple ACs and some AC specific attributes are needed, each "attachment-circuit" can specify attributes, such as interface specific IP addresses, service MTU, etc.

* "status": Enables the control of the administrative status and report the operational status of the SDP. These status values can be used as indicator to detect SDP anomalies.

* "sdp-monitoring": Provides SDP bandwidth statistics.

Depending on the requirements of different cases, "service-match-criteria" can be used for the following purposes:

* Specify the AC type: physical or logical connection

* Distinguish the SDP traffic if the SDP is located in the CE or PE

* Distinguish the traffic of different connection groups (CGs) or connectivity constructs (CCs) when multiple CGs/CCs of different SLO/SLE may be set up between the same pair of SDPs, as illustrated in Figure 9. Traffic needs to be explicitly mapped into the Network Slice's specific connectivity construct. The policies, "service-match-criteria", are based on the values in which combination of layer 2 and layer 3 header and payload fields within a packet to identify to which {Network Slice Service, connectivity construct, and SLOs/SLEs} that packet is assigned.

* Define specific out-of-profile policies: The customer may choose to use an explicit "service-match-criteria" to map any SDP's traffic or a subset of the SDP's traffic to a specific connection group or connectivity construct. If a subset of traffic is matched (e.g., "dscp-match") and mapped to a connectivity construct, the customer may choose to add a subsequent "match-any" to explicitly map the remaining SDP traffic to a separate connectivity construct. If the customer chooses to implicitly map remaining traffic and if there are no additional connectivity constructs where the "sdp-id" source is specified, then that traffic will be dropped.

Commenté [MB4]: Please check if you meant "sdp/id"

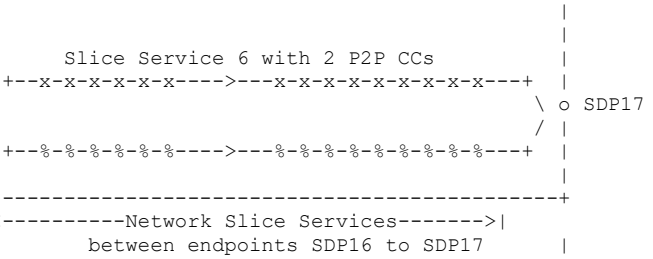
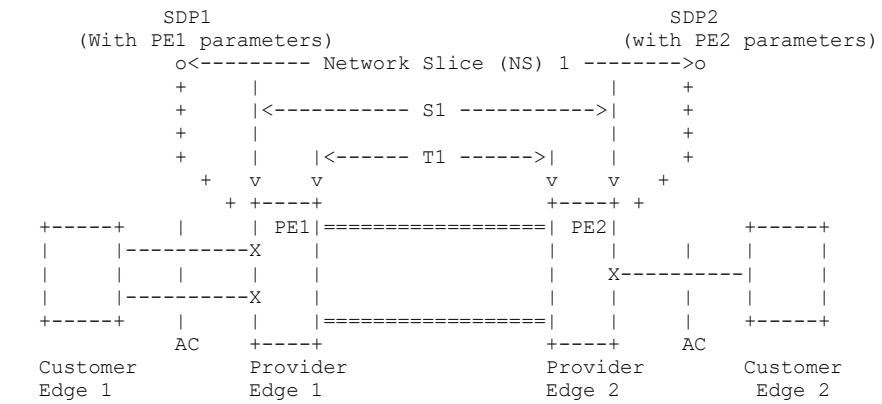


Figure 9: Application of Match Criteria

If an SDP is placed at the port of a CE or PE, and there is only one single connectivity construct with a source at the SDP, traffic can be implicitly mapped to this connectivity construct since the AC information (e.g., VLAN tag) can be used to unambiguously identify the traffic and the SDP is the only source of the connectivity-construct. Appendix B.1 shows an example of both the implicit and explicit approaches. While explicit matching is optional in some use cases, it provides a more clear and readable implementation, but the choice is left to the operator.

To illustrate the use of SDP options, Figure 10 and Figure 11 are two examples. How an NSC realize the mapping is out of scope for this document.

* SDPs at customer-facing ports on the PEs: As shown in Figure 10~~,~~, a customer of the Network Slice Service would like to connect two SDPs to satisfy specific service needs, e.g., network wholesale services. In this case, the Network Slice SDPs are mapped to customer-facing ports of PE nodes. The NSC uses "node-id" (PE device ID), "attachment-circuits" ~~(ACs)~~ or "ac-svc-name" to map SDPs to the customer-facing ports on the PEs.



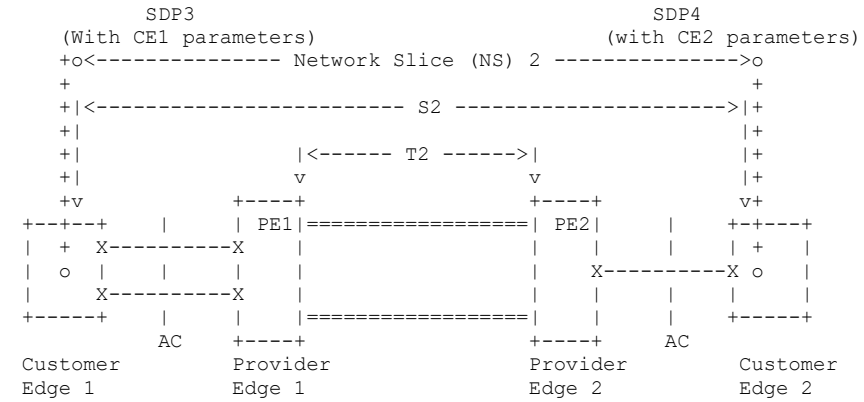
Legend:
o: Representation of an SDP
+: Mapping of an SDP to customer-facing ports on the PE
X: Physical interfaces used for realization of the NS Service
S1: L0/L1/L2/L3 services used for realization of NS Service
T1: Tunnels used for realization of NS Service

Figure 10: An Example of SDPs Placing at PEs

* SDPs within CEs: As shown in Figure ~~11 ,~~11, a customer of the Network Slice Service would like to connect two SDPs to provide connectivity between transport portion of 5G RAN to 5G Core network functions. In this scenario, the NSC uses "node-id" (CE device ID), "geo-location", "sdp-ip-address" (IP address of SDP for management), "service-match-criteria" (VLAN tag), "attachment-circuits" or ~~or~~ "ac-svc-name" (CE ACs) to map SDPs to the CE. The

NSC can use these CE parameters (and optionally other information

to uniquely identify a CE within an NSC, such as "peer-sap-id" [RFC9408]) to retrieve the corresponding PE device, interface and AC mapping details to complete the Network Slice Service provisioning.



Legend:
o: Representation of an SDP
+: Mapping of an SDP to CE
X: Physical interfaces used for realization of the NS Service
S2: L0/L1/L2/L3 services used for realization of the NS Service
T2: Tunnels used for realization of NS Service

Figure 11: An Example of SDPs Placing at CEs

5.2.2. Connectivity Constructs

Section 4.2.1 of [RFC9543] defines the basic connectivity construct (CC) and CC types of a Network Slice Service, including P2P, P2MP, and A2A.

A Network Slice Service involves one or more connectivity constructs. The "connection-groups" container is used to abstract CC, CC groups, and their SLO-SLE policies and the structure is shown in Figure 12.

```
+--rw connection-groups
   +--rw connection-group* [id]
      +--rw id                                  string
      +--rw connectivity-type?
      |       identityref
      +--rw (slo-sle-policy)?
      |  +--:(standard)
      |  |     ...
      |  +--:(custom)
      |        ...
      +--rw service-slo-sle-policy-override?
      |       identityref
      +--rw connectivity-construct* [id]
```

```
        |  +--rw id
        |  |       uint32
        |  +--rw (type)?
        |  |     ...
        |  +--rw (slo-sle-policy)?
        |  |     ...
        |  +--rw service-slo-sle-policy-override?
        |  |       identityref
        |  +--rw status
        |  |     ...
```

Figure 12: Connection Groups Subtree Structure

Commenté [MB5]: Why this is not string as used for the connection group?

The description of the "connection-groups" data nodes is as follows:

* "connection-group": Represents a group of CCs. In the case of hub and spoke connectivity of the Slice Service, it may be inefficient when there are a large number of SDPs with the multiple CCs. As illustrated in Appendix B.3, "connectivity-type" of "ietf-vpn-common:hub-spoke" and "connection-group-sdp-role" of "ietf-vpn-common:hub-role" or "ietf-vpn-common:spoke-role" can be specified [RFC9181]. Another use is for optimizing "slo-sle-policy" configurations, treating CCs with the same SLO and SLE characteristics as a connection group such that the connectivity construct can inherit the SLO/SLE from the group if not explicitly defined.

* "connectivity-type": Indicates the type of the connection group, extending "vpn-common:vpn-topology" specified [RFC9181] with the NS connectivity type, e.g., P2P and P2MP.

* "connectivity-construct": Represents single connectivity construct, and "slo-sle-policy" under it represents the per-connectivity construct SLO and SLE requirements.

* "slo-sle-policy" and "service-slo-sle-policy-override": The details of "slo-sle-policy" ~~is~~ are defined in Section 5.2.3. In addition to "slo-sle-policy" nodes of "connection-group" and "connectivity-construct", a leaf node "service-slo-sle-policy-override" is provided for scenarios with complex SLO-SLE requirements to completely override all or part of ~~an~~ a "slo-sle-policy" with new values. For example, if a particular "connection-group" or a "connectivity-construct" has a unique bandwidth or latency setting, that are different from those defined in the Slice Service, a new set of SLOs/SLEs with full or partial override can be applied. In the case of partial override, only the newly specified parameters are replaced from the original template, while maintaining on pre-existing parameters not specified. While a full override removes all pre-existing parameters, and in essence starts a new set of SLOs/SLEs which are specified.

5.2.3. SLO and SLE Policy

As defined in Section 5 of [RFC9543], the SLO and SLE policy of the Network Slice Services define some common attributes.

"slo-sle-policy" is used to represent these SLO and SLE policies.

During the creation of a Network Slice Service, the policy can be specified either by a standard SLO and SLE template or a customized SLO and SLE policy.

The policy can apply to per-network Slice Service, per-connection group "connection group", or per-connectivity construct "connectivity-construct". Since there are multiple mechanisms for assigning a policy to a single connectivity construct, an override precedence order among them is as follows:

* Connectivity-construct at an individual sending SDP

* Connectivity-construct

* Connection-group

* Slice-level

That is, the policy assigned through the sending SDP has highest precedence, and the policy assigned by the slice level has lowest precedence. Therefore, the policy assigned through the sending SDP takes precedence over the policy assigned through the connection-construct entry. Appendix B.5 gives an example of the preceding policy, which shows a Slice Service having an A2A connectivity as default and several specific SLO connections.

The SLO attributes include performance metric attributes, availability, and MTU. The SLO structure is shown in Figure 13.

```
+--rw slo-policy
|  +--rw metric-bound* [metric-type]
|  |  +--rw metric-type
|  |  |       identityref
|  |  +--rw metric-unit          string
|  |  +--rw value-description?   string
|  |  +--rw percentile-value?
|  |  |       percentile
|  |  +--rw bound?               uint64
|  +--rw availability?   identityref
|  +--rw mtu?            uint16
```

Figure 13: SLO Policy Subtree Structure

The list "metric-bound" supports the generic performance metric variations and the combinations and each "metric-bound" could specify a particular "metric-type". "metric-type" is defined with YANG identity and supports the following options:

"one-way-bandwidth": Indicates the guaranteed minimum bandwidth between any two SDPs. ~~And t~~The bandwidth is unidirectional.

"two-way-bandwidth": Indicates the guaranteed minimum bandwidth between any two SDPs. ~~And t~~The bandwidth is bidirectional.

"one-way-delay-maximum": Indicates the maximum one-way latency between two SDPs, defined in [RFC7679].

"two-way-delay-maximum": Indicates the maximum round-trip latency

between two SDPs, defined in [RFC2681].

"one-way-delay-percentile": Indicates the percentile objective of the one-way latency between two SDPs (See [RFC7679]).

"two-way-delay-percentile": Indicates the percentile objective of the round-trip latency between two SDPs (See [RFC2681]).

"one-way-delay-variation-maximum": Indicates the jitter constraint of the slice maximum permissible delay variation, and is measured by the difference in the one-way latency between sequential packets in a flow, as defined in [RFC3393].

"two-way-delay-variation-maximum": Indicates the jitter constraint of the slice maximum permissible delay variation, and is measured by the difference in the two-way latency between sequential packets in a flow, as defined in [RFC3393].

"one-way-delay-variation-percentile": Indicates the percentile objective of the delay variation, and is measured by the difference in the one-way latency between sequential packets in a flow, as defined in [RFC3393].

"two-way-delay-variation-percentile": Indicates the percentile objective of the delay variation, and is measured by the difference in the two-way latency between sequential packets in a flow, as defined in [RFC5481].

"one-way-packet-loss": Indicates maximum permissible packet loss rate (See [RFC7680], which is defined by the ratio of packets dropped to packets transmitted between two SDPs.

"two-way-packet-loss": Indicates maximum permissible packet loss rate (See [RFC7680], which is defined by the ratio of packets dropped to packets transmitted between two SDPs.

"availability": Specifies service availability defined as the ratio of uptime to the sum of uptime and downtime, where uptime is the time the Network Slice is available in accordance with the SLOs associated with it.

"mtu": Refers to the service MTU. If the customer sends packets that are longer than the requested service MTU, the network may discard it (or for IPv4, fragment it). Depending on the service layer, the value can be an L3 service MTU (Section 7.6.6 [RFC9182]) or an L2 service MTU (Section 7.4 of [RFC9291]).

As shown in Figure 14, the following SLEs data nodes are defined.

"security": The security leaf-list defines the list of security Functions that the customer requests the operator to apply to traffic between the two SDPs, including authentication, encryption, etc., which is defined in Section 5.1.2.1 [RFC9543].

"isolation": Specifies the isolation types that a customer expects, as defined in Section 8 of [RFC9543].

"max-occupancy-level": Specifies the number of flows that the

Commenté [MB6]: The service is slicing here. I guess you meant that the MTU is L3 if the L3 techniques are used for the realization and L2 otherwise.

I would personnally simplify and always assume this is L2 MTU.

operator admits (See Section 5.1.2.1 of [RFC9543]).

"path-constraints": Specifies the path constraints the customer requests for the Network Slice Service, including geographic restrictions and diversity which is defined in Section 5.1.2.1 of [RFC9543].

```
+--rw sle-policy
   +--rw security*              identityref
   +--rw isolation*             identityref
   +--rw max-occupancy-level?   uint8
   +--rw path-constraints
      +--rw service-functions
      +--rw diversity
         +--rw diversity-type?
                 te-types:te-path-disjointness
```

Figure 14: SLE Policy Subtree Structure

Figure 15 shows an example with a network slice "slo-policy".

```
{
  " ietf-network-slice-service:network-slice-services": {
    "slice-service": [{
      "id": "exp-slice",
      "service-slo-sle-policy": {
        "description": "video-service-policy",
        "slo-policy": {
          "metric-bound": [
            {
              "metric-type": "ietf-nss:one-way-bandwidth",
              "metric-unit": "Mbps",
              "bound": "1000"
            },
            {
              "metric-type": "ietf-nss:two-way-delay-maximum",
              "metric-unit": "milliseconds",
              "bound": "10"
            }
          ],
          "availability": "ietf-nss:level-4",
          "mtu": "1500"
        }
      }]
  }
}
```

Figure 15: An Example of a Slice Service of SLO Policies

5.2.4. Network Slice Service Performance Monitoring

The operation and performance status of Network Slice Services is also a key component of the NSSM. The model provides SLO monitoring information with the following granularity:

* Per SDP: The incoming and outgoing bandwidths of an SDP are

Commenté [MB7]: Does not match the DM

a mis en forme : Anglais (États-Unis)

Commenté [MB8]: Please check the structure

Commenté [MB9]: Seems "sdps" are missing

specified in "sdp-monitoring" under the "sdp".

* Per connectivity construct: The delay, delay variation, and packet loss status are specified in "connectivity-construct-monitoring" under the "connectivity-construct".

* Per connection group: The delay, delay variation, and packet loss status are specified in "connection-group-monitoring" under the "connection-group".

[RFC8639] and [RFC8641] define a subscription mechanism and a push mechanism for YANG datastores. These mechanisms currently allow the user to subscribe to notifications on a per-client basis and specify either periodic or on-demand notifications. By specifying subtree filters or xpath filters to "sdp", "connectivity-construct", or "connection-group", so that only interested contents will be sent. The example in Figure 24 shows the way for a customer to subscribe to the monitoring information for a particular Network Slice Service. .

~~Additionaly~~Additionally, a customer can use the NSSM to obtain a snapshot of the Network Slice Service performance status through [RFC8040] or [RFC6241] interfaces. For example, retrieve the per-connectivity-~~construc~~construct data by specifying "connectivity-construct" as the filter in the RESTCONF GET request.

5.2.5. Custom Topology Constraints

~~The~~ A Slice Service customer might request for some level of control over the topology or resources constraints. "custom-topology" is defined as an augmentation target that references the context topology. The leaf "network-ref" under this container is used to reference a predefined topology as a customized topology constraint for ~~an~~ a Network Slice Service. Section 1 of [RFC8345] defines a general abstract topology concept to accommodate both the provider's resource capability and the customer's preferences. The abstract topology is a topology that contains abstract topological elements (nodes, links, and termination points).

This document defines only the minimum attributes of a custom topology, which can be extended based on the implementation requirements.

The following nodes are defined for the custom topology:

"custom-topology": This container serves as an augmentation target for the Slice Service topology context, which can be multiple. This node is located directly under the "slice-service" list.

"network-ref": This leaf is under the container "custom-topology", which is defined to reference a predefined topology as a customized topology constraint for a Network Slice Service, e.g., a ~~Service Attachment Points (~~SAP~~s)~~ topology to request SDP feasibility checks on ~~a~~ SAPs network described in Section 3 of [RFC9408], an abstract Traffic Engineering (TE) topology defined in ~~section~~Section ~~-~~ 3.13 of [RFC8795] to customize the service paths in a

Network Slice Service.

"tp-ref": A reference to a Termination Point (TP) in the custom topology, under the list "sdp", can be used to associate an SDP with a TP of the customized topology. The example TPs could be parent termination points of the SAP topology.

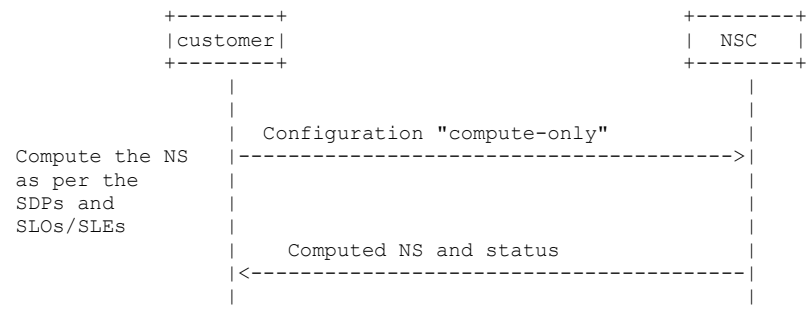
5.2.6. Network Slice Service Compute

A Network Slice Service is, by default, provisioned so that it can instantiate and trigger service delivery. A Network Slice Service customer may request to check the feasibility of a request before instantiating or modifying a Network Slice Service. In such a case, the Network Slice Service is configured in "compute-only" mode to distinguish it from the default behavior.

Commenté [MB10]: Why this mention?

A "compute-only" Network Slice Service is configured as usual with the associated per slice SLOs/SLEs. The NSC computes the feasible connectivity constructs to the configured SLOs/SLEs. This computation does not create the Network Slice or reserve any resources in the provider's network, it simply computes the resulting Network Slice based on the request. The Network Slice "admin-status" and the connection groups or connectivity construct list are used to convey the result. For example, "admin-compute-only" can be used to show the status. Customers can query the "compute-only" connectivity constructs attributes, or can subscribe to be notified when the connectivity constructs status change.

The "compute-only" applies only if the data model is used with a protocol that does not natively support such operation, e.g., [RFC8040]. When using NETCONF, <edit-config> operation (Section 7.2 of [RFC6241]), "test-only" of the <test-option> parameter also applies.



NS: Network Slice

Figure 16: An Example of Network Slice Service Compute

6. Network Slice Service Module

The "ietf-network-slice-service" module uses types defined in [RFC6991], [RFC8345], [RFC9179], [RFC9181], [RFC8776], and [RFC7640].

```
<CODE BEGINS> file "ietf-network-slice-service@2024-03-17.yang"
module ietf-network-slice-service {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-network-slice-service";
  prefix ietf-nss;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Types";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-geo-location {
    prefix geo;
    reference
      "RFC 9179: A YANG Grouping for Geographic Locations";
  }
  import ietf-vpn-common {
    prefix vpn-common;
    reference


      "RFC 9181: A Common YANG Data Model for Layer 2 and Layer 3
                 VPNs";
  }
  import ietf-network {
    prefix nw;
    reference
      "RFC 8345: A YANG Data Model for Network Topologies";
  }
  import ietf-network-topology {
    prefix nt;
    reference
      "RFC 8345: A YANG Data Model for Network
                 Topologies, Section 6.2";
  }
  import ietf-te-types {
    prefix te-types;
    reference
      "RFC 8776: Traffic Engineering Common YANG Types";
  }

  organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
     Working Group";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/teas/>
     WG List:  <mailto:teas@ietf.org>

     Editor: Bo Wu
           <lana.wubo@huawei.com>
     Editor: Dhruv Dhody
           <dhruv.ietf@gmail.com>
```

```
          Editor: Reza Rokui
                  <rrokui@ciena.com>
          Editor: Tarek Saad
                  <tsaad@cisco.com>
          Editor: John Mullooly
                  <jmullool@cisco.com>";
      description
        "This YANG module defines a service model for the RFC 9543 Network
Slice
         Service.

         Copyright (c) 2024 IETF Trust and the persons identified as
         authors of the code.  All rights reserved.

         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and subject to
         the license terms contained in, the Revised BSD License set
         forth in Section 4.c of the IETF Trust's Legal Provisions
         Relating to IETF Documents
         (https://trustee.ietf.org/license-info).

         This version of this YANG module is part of RFC AAAA; see the
         RFC itself for full legal notices.";

      revision 2024-03-17 {
        description
          "Initial revision.";
        reference
          "RFC AAAA: A YANG Data Model for the RFC 9543 Network Slice Service
A YANG Data Model for Network Slice Service";
      }

      /* Identities */

      identity service-tag-type {
        description
          "Base identity of Network Slice Service tag type.";
      }

      identity service-tag-customer {
        base service-tag-type;
        description
          "The Network Slice Service customer name tag type,
           e.g., adding tags with 'customer-name' when multiple actual
           customers use a same Network Slice Service.";
      }

      identity service-tag-service {
        base service-tag-type;
        description
          "The Network Slice Service tag type, which can indicate the
           technical constraints used during service realization,
           for example, Layer 2 or Layer 3 technologies.";
      }

      identity service-tag-opaque {
        base service-tag-type;
        description
```

Commenté [MB11]: May be worth defining more what a tag is about

Commenté [MB12]: No need to prefix with the base identity name.

Commenté [MB13]: idem

```
      "An opaque type, which can be used for future use,
       such as filtering of services.";
  }

  identity attachment-circuit-tag-type {
    description
      "Base identity for the attachment circuit tag type.";
  }

  identity vlan-id {
    base attachment-circuit-tag-type;
    description
      "Identity for VLAN ID tag type, 802.1Q dot1Q.";
    reference
      "IEEE Std 802.1Q: IEEE Standard for Local and Metropolitan
                        Area Networks--Bridges and Bridged
                        Networks";
  }

  identity cvlan-id {
    base attachment-circuit-tag-type;
    description
      "Identity for C-VLAN ID tag type, 802.1ad QinQ VLAN IDs.";
    reference
      "IEEE Std 802.1ad: IEEE Standard for Local and Metropolitan
                         Area Networks---Virtual Bridged Local
                         Area Networks---Amendment 4: Provider
                         Bridges";
  }

  identity svlan-id {
    base attachment-circuit-tag-type;
    description
      "Identity for S-VLAN ID tag type, 802.1ad QinQ VLAN IDs.";
    reference
      "IEEE Std 802.1ad: IEEE Standard for Local and Metropolitan
                         Area Networks---Virtual Bridged Local
                         Area Networks---Amendment 4: Provider
                         Bridges";
  }

  identity ip-address-mask {
    base attachment-circuit-tag-type;
    description
      "Identity for IP address mask tag type.";
  }

  identity service-isolation-type {
    description
      "Base identity for Network Slice Service isolation type.";
  }

  identity traffic-isolation {
    base service-isolation-type;
    description
      "Specify the requirement for separating the traffic of the
       customer's Network Slice Service from other services,
       which may be provided by the service provider using VPN
```

Commenté [MB14]: Please check this is listed in the references./

Commenté [MB15]: idem

```
     technologies, such as L3VPN, L2VPN, EVPN, etc.";
}

identity service-security-type {
  description
    "Base identity for Network Slice Service security type.";
}

identity authentication {
  base service-security-type;
  description
    "Indicates that the Slice Service requires authentication.";
}

identity integrity {
  base service-security-type;
  description
    "Indicates that the Slice Service requires data integrity.";
}

identity encryption {
  base service-security-type;
  description
    "Indicates that the Slice Service requires data encryption.";
}

identity point-to-point {
  base vpn-common:vpn-topology;
  description
    "Identity for point-to-point Network Slice
     Service connectivity.";
}

identity point-to-multipoint {
  base vpn-common:vpn-topology;
  description
    "Identity for point-to-multipoint Network Slice
     Service connectivity.";
}

identity multipoint-to-multipoint {
  base vpn-common:vpn-topology;
  description
    "Identity for multipoint-to-multipoint Network Slice
     Service connectivity.";
}

identity multipoint-to-point {
  base vpn-common:vpn-topology;
  description
    "Identity for multipoint-to-point Network Slice
     Service connectivity.";
}

identity sender-role {
  base vpn-common:role;
  description
    "Indicates that an SDP is acting as a sender.";
```

```
}

identity receiver-role {
  base vpn-common:role;
  description
    "Indicates that an SDP is acting as a receiver.";
}

identity service-slo-metric-type {
  description
    "Base identity for Network Slice Service SLO metric type.";
}

identity one-way-bandwidth {
  base service-slo-metric-type;
  description
    "SLO bandwidth metric. Minimum guaranteed bandwidth between
     two SDPs at any time and is measured unidirectionally.";
}

identity two-way-bandwidth {
  base service-slo-metric-type;
  description
    "SLO bandwidth metric. Minimum guaranteed bandwidth between
     two SDPs at any time.";
}

identity shared-bandwidth {
  base service-slo-metric-type;
  description
    "The shared SLO bandwidth bound. It is the limit on the
     bandwidth that can be shared amongst a group of
     connectivity constructs of a Slice Service.";
}

identity one-way-delay-maximum {
  base service-slo-metric-type;
  description
    "The SLO objective of this metric is the upper bound of network
     delay when transmitting between two SDPs.";
  reference
    "RFC_7679: A One-Way Delay Metric for IP Performance
     _________Metrics (IPPM)";
}

identity one-way-delay-percentile {
  base service-slo-metric-type;
  description
    "The SLO objective of this metric is percentile objective of
     network delay when transmitting between two SDPs.
     The metric is defined in RFC7679.";
  reference
    "RFC_7679: A One-Way Delay Metric for IP Performance
     Metrics (IPPM)";
}

identity two-way-delay-maximum {
  base service-slo-metric-type;
```

```
  description
    "SLO two-way delay is the upper bound of network delay when
     transmitting between two SDPs";
  reference
    "RFC_2681: A Round-trip Delay Metric for IPPM";
}

identity two-way-delay-percentile {
  base service-slo-metric-type;
  description
    "The SLO objective of this metric is the percentile
     objective of network delay when the traffic transmitting
     between two SDPs.";
  reference
    "RFC_2681: A Round-trip Delay Metric for IPPM";
}

identity one-way-delay-variation-maximum {
  base service-slo-metric-type;
  description
    "The SLO objective of this metric is maximum bound of the
     difference in the one-way delay between sequential packets
     between two SDPs.";
  reference
    "RFC_3393: IP Packet Delay Variation Metric for IP Performance
          Metrics (IPPM)";

}


identity one-way-delay-variation-percentile {
  base service-slo-metric-type;
  description
    "The SLO objective of this metric is the percentile objective
     in the one-way delay between sequential packets between two
     SDPs.";
  reference
    "RFC_3393: IP Packet Delay Variation Metric for IP Performance
          Metrics (IPPM)";
}

identity two-way-delay-variation-maximum {
  base service-slo-metric-type;
  description
    "SLO two-way delay variation is the difference in the
     round-trip delay between sequential packets between two SDPs.";
  reference
    "RFC_5481: Packet Delay Variation Applicability Statement";
}

identity two-way-delay-variation-percentile {
  base service-slo-metric-type;
  description
    "The SLO objective of this metric is the percentile objective
     in the round-trip delay between sequential packets between
     two SDPs.";
  reference
    "RFC_5481: Packet Delay Variation Applicability Statement";
```

a mis en forme : Anglais (États-Unis)

```
}

identity one-way-packet-loss {
  base service-slo-metric-type;
  description
    "This metric type refers to the ratio of packets dropped
     to packets transmitted between two SDPs in one-way.";
  reference
    "RFC_7680: A One-Way Loss Metric for IP Performance
     _________ Metrics (IPPM)";
}

identity two-way-packet-loss {
  base service-slo-metric-type;
  description
    "This metric type refers to the ratio of packets dropped
     to packets transmitted between two SDPs in two-way.";
  reference
    "RFC_7680: A One-Way Loss Metric for IP Performance
     _________ Metrics (IPPM)";
}

    /*
     * Identity for availability-type
     */

identity availability-type {
  description
    "Base identity for availability.";
}

identity level-1 {
  base availability-type;
  description
    "Specifies the availability level 1: 99.9999%";
}

identity level-2 {
  base availability-type;
  description
    "Specifies the availability level 2: 99.999%";
}

identity level-3 {
  base availability-type;
  description
    "Specifies the availability level 3: 99.99%";
}

identity level-4 {
  base availability-type;
  description
    "Specifies the availability level 4: 99.9%";
}

identity level-5 {
  base availability-type;
  description
```

Commenté [MB16]: If you keep it, please do the same of the other identities.

Commenté [MB17]: Usually called 5-nines. I would change the labels to align with the practice: 2-nines, ..5 nines, etc.

```
    "Specifies the availability level 5: 99%";
}

identity service-match-type {
  description
    "Base identity for Network Slice Service traffic
     match type.";
}

identity phy-interface-match {
  base service-match-type;
  description
    "Uses the physical interface as match criteria for
     Slice Service traffic.";
}

identity vlan-match {
  base service-match-type;
  description
    "Uses the VLAN ID as match criteria for the Slice Service
     traffic.";
}

identity label-match {
  base service-match-type;
  description
    "Uses the MPLS label as match criteria for the Slice Service
     traffic.";
}

identity source-ip-prefix-match {
  base service-match-type;
  description
    "Uses source IP prefix as match criteria for the Slice Service
     traffic. Examples of 'value' of this match type are
     '192.0.2.0/24' and '2001:db8::1/64'.";
}

identity destination-ip-prefix-match {
  base service-match-type;
  description
    "Uses destination IP prefix as match criteria for the Slice
     Service traffic. Examples of 'value' of this match type are
     '203.0.113.1/32', and '2001:db8::2/128'.";
}

identity dscp-match {
  base service-match-type;
  description
    "Uses DSCP field in the IP packet header as match criteria
     for the Slice Service traffic.";
}

identity acl-match {
  base service-match-type;
  description
    "Uses Access Control List (ACL) as match criteria
     for the Slice Service traffic.";
```

Commenté [MB18]: I think you can get rid of "match" suffix for all these identities.

```
    reference
      "RFC 8519: YANG Data Model for Network Access Control
                 Lists (ACLs)";
  }

  identity any-match {
    base service-match-type;
    description
      "Matches any Slice Service traffic.";
  }

  identity slo-sle-policy-override {
    description
      "Base identity for SLO/SLE policy override options.";
  }

  identity full-override {
    base slo-sle-policy-override;
    description
      "The SLO/SLE policy defined at the child level overrides a
       parent SLO/SLE policy, which means that no SLO/SLEs are
       inherited from parent if a child SLO/SLE policy exists.";
  }

  identity partial-override {
    base slo-sle-policy-override;
    description
      "The SLO/SLE policy defined at the child level updates the
       parent SLO/SLE policy. For example, if a specific SLO is
       defined at the child level, that specific SLO overrides
       the one inherited from a parent SLO/SLE policy, while all
       other SLOs in the parent SLO-SLE policy still apply.";
  }

  /* Typedef */

  typedef percentage {
    type decimal64 {
      fraction-digits 5;
      range "0..100";
    }
    description
      "Percentage to 5 decimal places.";
  }

  typedef percentile {
    type decimal64 {
      fraction-digits 3;
      range "0..100";
    }
    description
      "The percentile is a value between 0 and 100
       to 3 decimal places, e.g., 10.000, 99.900 ,99.990, etc.
       For example, for a given one-way delay measurement,
       if the percentile is set to 95.000 and the 95th percentile
       one-way delay is 2 milliseconds, then the 95 percent of
       the sample value is less than or equal to 2 milliseconds.";
  }
```

Commenté [MB19]: Not cited as a reference

```
    typedef slice-template-ref {
      type leafref {
        path "/ietf-nss:network-slice-services"
           + "/ietf-nss:slo-sle-templates"
           + "/ietf-nss:slo-sle-template"
           + "/ietf-nss:id";
      }
      description
        "This type is used by data models that need to reference
         Network Slice templates.";
    }

    /* Groupings */

    grouping service-slos {
      description
        "A reusable grouping for directly measurable objectives of
         a Slice Service.";
      container slo-policy {
        description
          "Contains the SLO policy.";
        list metric-bound {
          key "metric-type";
          description
            "List of Slice Service metric bounds.";
          leaf metric-type {
            type identityref {
              base service-slo-metric-type;
            }
            description
              "Identifies SLO metric type of the Slice Service.";
          }
          leaf metric-unit {
            type string;
            mandatory true;
            description
              "The metric unit of the parameter. For example,
               for time units, where the options are hours, minutes,
               seconds, milliseconds, microseconds, and nanoseconds;
               for bandwidth units, where the options are bps, Kbps,
               Mbps, Gbps; for the packet loss rate unit,
               the options can be percentage.";
          }
          leaf value-description {
            type string;
            description
              "The description of the provided value.";
          }
          leaf percentile-value {
            type percentile;
            description
              "The percentile value of the metric type.";
          }
          leaf bound {
            type uint64;
            description
              "The bound on the Slice Service connection metric.
```

Commenté [MB20]: What about refs for referencing a slice service? I would add a new typedef for that as well.

```
            When set to zero, this indicates an unbounded
            upper limit for the specific metric-type.";
        }
      }
      leaf availability {
        type identityref {
          base availability-type;
        }
        description
          "Service availability level";
      }
      leaf mtu {
        type uint32;
        units "bytes";
        description
          "The MTU specifies the maximum length of data
           packets of the Slice Service.
           The value needs to be less than or equal to the
           minimum MTU value of all 'attachment-circuits'
           in the SDPs.";
      }
    }
  }

  grouping service-sles {
    description
      "A reusable grouping for indirectly measurable objectives of
       a Slice Service.";
    container sle-policy {
      description
        "Contains the SLE policy.";
      leaf-list security {
        type identityref {
          base service-security-type;
        }
        description
          "The security functions that the customer requests
           the operator to apply to traffic between the two SDPs.";
      }
      leaf-list isolation {
        type identityref {
          base service-isolation-type;
        }
        description
          "The Slice Service isolation requirement.";
      }
      leaf max-occupancy-level {
        type uint8 {
          range "1..100";
        }
        description
          "The maximal occupancy level specifies the number of flows
           to be admitted and optionally a maximum number of
           countable resource units (e.g., IP or MAC addresses)
           a Network Slice Service can consume.";
      }
      container path-constraints {
        description
```

Commenté [MB21]: Indicates this is about L2 MTU (or whatever choice you made)

```
        "Container for the policy of path constraints
         applicable to the Slice Service.";
      container service-functions {
        description
          "Container for the policy of service function
           applicable to the Slice Service.";
      }
      container diversity {
        description
          "Container for the policy of disjointness
           applicable to the Slice Service.";
        leaf diversity-type {
          type te-types:te-path-disjointness;
          description
            "The type of disjointness on Slice Service, i.e.,
             across all connectivity constructs.";
        }
      }
    }
  }
}

grouping slice-service-template {
  description
    "A reusable grouping for Slice Service templates.";
  container slo-sle-templates {
    description
      "Contains a set of Slice Service templates.";
    list slo-sle-template {
      key "id";
      description
        "List for SLO and SLE template identifiers.";
      leaf id {
        type string;
        description
          "Identification of the Service Level Objective (SLO)
           and Service Level Expectation (SLE) template to be used.
           Local administration meaning.";
      }
      leaf description {
        type string;
        description
          "Describes the SLO and SLE policy template.";
      }
      leaf template-ref {
        type slice-template-ref;
        description
          "The reference to a standard template. When set it
           -indicates the base template over which further
           -SLO/SLE policy changes are made.";
      }
      uses service-slos;
      uses service-sles;
    }
  }
}

grouping service-slo-sle-policy {
```

```
      description
        "Slice service policy grouping.";
      choice slo-sle-policy {
        description
          "Choice for SLO and SLE policy template.
           Can be standard template or customized template.";
        case standard {
          description
            "Standard SLO template.";
          leaf slo-sle-template {
            type slice-template-ref;
            description
              "Standard SLO and SLE template to be used.";
          }
        }
        case custom {
          description
            "Customized SLO and SLE template.";
          container service-slo-sle-policy {
            description
              "Contains the SLO and SLE policy.";
            leaf description {
              type string;
              description
                "Describes the SLO and SLE policy.";
            }
            uses service-slos;
            uses service-sles;
          }
        }
      }
    }

    grouping bw-rate-limits {
      description
        "Grouping for bandwidth rate limits.";
      reference
        "RFC 7640: Traffic Management Benchmarking";
      leaf cir {
        type uint64;
        units "bps";
        description
          "Committed Information Rate. The maximum number of bits
           that a port can receive or send during one-second over an
           interface.";
      }
      leaf cbs {
        type uint64;
        units "bytes";
        description
          "Committed Burst Size. CBS controls the bursty nature
           of the traffic. Traffic that does not use the configured
           CIR accumulates credits until the credits reach the
           configured CBS.";
      }
      leaf eir {
        type uint64;
        units "bps";
```

```
        description
          "Excess Information Rate, i.e., excess frame delivery
           allowed not subject to SLA. The traffic rate can be
           limited by EIR.";
      }
      leaf ebs {
        type uint64;
        units "bytes";
        description
          "Excess Burst Size. The bandwidth available for burst
           traffic from the EBS is subject to the amount of
           bandwidth that is accumulated during periods when
           traffic allocated by the EIR policy is not used.";
      }
      leaf pir {
        type uint64;
        units "bps";
        description
          "Peak Information Rate, i.e., maximum frame delivery
           allowed. It is equal to or less than sum of CIR and EIR.";
      }
      leaf pbs {
        type uint64;
        units "bytes";
        description
          "Peak Burst Size.";
      }
    }

  grouping service-qos {
    description
      "Grouping for the Slice Service QoS policy.";
    container incoming-qos-policy {
      description
        "The QoS policy imposed on ingress direction of the traffic ,
         from the customer network or from another provider's
         network.";
      leaf qos-policy-name {
        type string;
        description
          "The name of the QoS policy that is applied to the
           attachment circuit. The name can reference a QoS
           profile that is pre-provisioned on the device.";
      }
      container rate-limits {
        description
          "Container for the asymmetric traffic control.";
        uses bw-rate-limits;
        container classes {
          description
            "Container for service class bandwidth control.";
          list cos {
            key "cos-id";
            description
              "List of Class of Services.";
            leaf cos-id {
              type uint8;
              description
```

Commenté [MB22]: I would reuse the "grouping bandwidth-parameters" from the common AC module.

```
              "Identifier of the CoS, indicated by
               a Differentiated Services Code Point
               (DSCP) or a CE-CLAN CoS (802.1p)
               value in the service frame.";
            reference
              "IEEE Std 802.1Q: Bridges and Bridged
                                Networks";
          }
          uses bw-rate-limits;
        }
      }
    }
  }
  container outgoing-qos-policy {
    description
      "The QoS policy imposed on egress direction of the traffic,
       towards the customer network or towards another
       provider's network.";
    leaf qos-policy-name {
      type string;
      description
        "The name of the QoS policy that is applied to the
         attachment circuit. The name can reference a QoS
         profile that is pre-provisioned on the device.";
    }
    container rate-limits {
      description
        "The rate-limit imposed on outgoing traffic.";
      uses bw-rate-limits;
      container classes {
        description
          "Container for classes.";
        list cos {
          key "cos-id";
          description
            "List of Class of Services.";
          leaf cos-id {
            type uint8;
            description
              "Identifier of the CoS, indicated by
               a Differentiated Services Code Point
               (DSCP) or a CE-CLAN CoS (802.1p)
               value in the service frame.";
            reference
              "IEEE Std 802.1Q: Bridges and Bridged
                                Networks";
          }
          uses bw-rate-limits;
        }
      }
    }
  }
}

grouping service-slo-sle-policy-override {
  description
    "Slice Service policy override grouping.";
  leaf service-slo-sle-policy-override {
```

```
        type identityref {
          base slo-sle-policy-override;
        }
        default "ietf-nss:full-override";
        description
          "SLO/SLE policy override option.";
      }
    }

    grouping one-way-performance-metrics {
      description
        "One-way PM metrics grouping.";
      leaf one-way-min-delay {
        type yang:gauge64;
        description
          "One-way minimum delay or latency in microseconds.";
      }
      leaf one-way-max-delay {
        type yang:gauge64;
        description
          "One-way maximum delay or latency in microseconds.";
        reference
          "RFC 7679: A One-Way Delay Metric for IP Performance
           Metrics (IPPM)";
      }
      leaf one-way-delay-variation {
        type yang:gauge64;
        description
          "One-way delay variation in microseconds.";
        reference
          "RFC 3393: IP Packet Delay Variation Metric for IP Performance
           Metrics (IPPM)";
      }
      leaf one-way-packet-loss {
        type percentage;
        description
          "The ratio of packets dropped to packets transmitted between
           two endpoints.";
        reference
          "RFC 7680: A One-Way Loss Metric for IP Performance
           Metrics (IPPM)";
      }
    }

    grouping two-way-performance-metrics {
      description
        "Two-way packet PM metrics grouping.";
      leaf two-way-min-delay {
        type yang:gauge64;
        description
          "Two-way minimum delay or latency in microseconds.";
        reference
          "RFC 2681: A Round-trip Delay Metric for IPPM";
      }
      leaf two-way-max-delay {
        type yang:gauge64;
        description
          "Two-way maximum delay or latency in microseconds.";
```

```
      reference
        "RFC_2681: A Round-trip Delay Metric for IPPM";
    }
    leaf two-way-delay-variation {
      type yang:gauge64;
      description
        "Two-way delay variation in microseconds.";
      reference
        "RFC_5481: Packet Delay Variation Applicability Statement";
    }
    leaf two-way-packet-loss {
      type percentage;
      description
        "The ratio of packets dropped to packets transmitted between
         two endpoints.";
    }
  }

  grouping connectivity-construct-monitoring-metrics {
    description
      "Grouping for connectivity construct monitoring metrics.";
    uses one-way-performance-metrics;
    uses two-way-performance-metrics;
  }

  /* Main Network Slice Services Container */

  container network-slice-services {
    description
      "Contains a list of Network Slice Services";
    uses slice-service-template;
    list slice-service {
      key "id";
      description
        "A Slice Service is identified by a service id.";
      leaf id {
        type string;
        description
          "A unique Slice Service identifier within an NSC.";
      }
      leaf description {
        type string;
        description
          "Textual description of the Slice Service.";
      }
      container service-tags {
        description
          "Container for a list of service tags for management
           purposes, such as policy constraints
           (e.g., Layer 2 or Layer 3 technology realization),
           classification (e.g., customer names, opaque values).";
        list tag-type {
          key "tag-type";
          description
            "The service tag list.";
          leaf tag-type {
            type identityref {
              base service-tag-type;
```

```
            }
            description
              "Slice Service tag type, e.g., realization technology
               constraints, customer name, or other customer-defined
               opaque types.";
          }
          leaf-list value {
            type string;
            description
              "The tag values, e.g., 5G customer names when multiple
               customers share the same Slice Service in 5G scenario,
               or Slice realization technology (such as Layer 2 or
               Layer 3).";
          }
        }
      }
      uses service-slo-sle-policy;
      leaf compute-only {
        type empty;
        description
          "When present, the slice is computedthis is a feasibility
check. That is, No no resources are
           committed or reserved in the network.";
      }
      uses vpn-common:service-status;
      container sdps {
        description
          "Slice Service SDPs.";
        list sdp {
          key "id";
          min-elements 2;
          description
            "List of SDPs in this Slice Service.";
          leaf id {
            type string;
            description
              "The unique identifier of the SDP within the scope of
               an NSC.";
          }
          leaf description {
            type string;
            description
              "Provides a description of the SDP.";
          }
          uses geo:geo-location;
          leaf node-id {
            type string;
            description
              "A unique identifier of an edge node of the SDP
               within the scope of the NSC.";
          }
          leaf-list sdp-ip-address {
            type inet:ip-address;
            description
              "IPv4 or IPv6 address of the SDP.";
          }
          leaf tp-ref {
            type leafref {
```

```
      path
        "/nw:networks/nw:network[nw:network-id="
      + "current()/../../../custom-topology/network-ref]/"
      + "nw:node/nt:termination-point/nt:tp-id";
    }
    description
      "A reference to Termination Point (TP) in the custom
       topology";
    reference
      "RFC 8345: A YANG Data Model for Network Topologies";
  }
  container service-match-criteria {
    description
      "Describes the Slice Service match criteria.";
    list match-criterion {
      key "index";
      description
        "List of the Slice Service traffic match criteria.";
      leaf index {
        type uint32;
        description
          "The identifier of a match criteria.";
      }
      leaf match-type {
        type identityref {
          base service-match-type;
        }
        mandatory true;
        description
          "Indicates the match type of the entry in the
           list of the Slice Service match criteria.";
      }
      leaf-list value {
        type string;
        description
          "Provides a value for the Slice Service match
           criteria, e.g. IP prefix and VLAN ID.";
      }
      leaf target-connection-group-id {
        type leafref {
          path
            "../../../../../ietf-nss:connection-groups"
          + "/ietf-nss:connection-group"
          + "/ietf-nss:id";
        }
        mandatory true;
        description
          "Reference to the Slice Service connection group.";
      }
      leaf connection-group-sdp-role {
        type identityref {
          base vpn-common:role;
        }
        default "vpn-common:any-to-any-role";
        description
          "Specifies the role of SDP in the connection group
           When the service connection type is MP2MP,
           such as hub and spoke service connection type.
```

```
            In addition, this helps to create connectivity
            construct automatically, rather than explicitly
            specifying each one.";
        }
        leaf target-connectivity-construct-id {
          type leafref {
            path
              "../../../../../ietf-nss:connection-groups"
            + "/ietf-nss:connection-group[ietf-nss:id="
            + "current()/../target-connection-group-id]"
            + "/ietf-nss:connectivity-construct/ietf-nss:id";
          }
          description
            "Reference to a Network Slice connection
             construct.";
        }
      }
    }
    uses service-qos;
    container sdp-peering {
      description
        "Describes SDP peering attributes.";
      leaf-list peer-sap-id {
        type string;
        description
          "Indicates the reference to the remote endpoints of
           the attachment circuits. This information can be used
           for correlation purposes, such as identifying SAPs
           of provider equipments when requesting a service with
           CE based SDP attributes.";
        reference
          "RFC 9408: A YANG Network Data Model for Service
           Attachment Points (SAPs)";
      }
      container protocols {
        description
          "Serves as an augmentation target.
           Protocols can be augmented into this container,
           e.g., BGP, static routing.";
      }
    }
    leaf-list ac-svc-name {
      type string;
      description
        "Indicates the attachment circuit service names for
         association purposes, to refer to ACs that have been
         created before the slice creation.";
      reference
        "draft-ietf-opsawg-teas-attachment-circuit-02:
         YANG Data Models for
         'Attachment Circuits'-as-a-Service (ACaaS)";
    }
    leaf ce-mode {
      type boolean;
      description
        "Indicates that SDP is on the CE.";
    }
    container attachment-circuits {
```

Commenté [MB23]: Now that the AC spec is stable, I think using the ref exposed by the AC model would be cleaner.

```
description
  "List of attachment circuits.";
list attachment-circuit {
  key "id";
  description
    "The Network Slice Service SDP attachment circuit
     related parameters.";
  leaf id {
    type string;
    description
      "The identifier of attachment circuit.";
  }
  leaf description {
    type string;
    description
      "The attachment circuit's description.";
  }
  leaf ac-svc-name {
    type string;
    description
      "Indicates an attachment circuit (AC) service name
       for association purposes, to refer to an AC that
       has been created before the slice creation.
       This node can override 'ac-svc-name' of
       the parent SDP.";
    reference
      "draft-ietf-opsawg-teas-attachment-circuit-02:
       YANG Data Models for
       'Attachment Circuits'-as-a-Service (ACaaS)";
  }
  leaf ac-node-id {
    type string;
    description
      "The attachment circuit node ID in the case of
       multi-homing.";
  }
  leaf ac-tp-id {
    type string;
    description
      "The termination port ID of the
       attachment circuit.";
  }
  leaf ac-ipv4-address {
    type inet:ipv4-address;
    description
      "The IPv4 address of the AC.";
  }
  leaf ac-ipv4-prefix-length {
    type uint8;
    description
      "The IPv4 subnet prefix length expressed in bits.";
  }
  leaf ac-ipv6-address {
    type inet:ipv6-address;
    description
      "The IPv6 address of the AC.";
  }
  leaf ac-ipv6-prefix-length {
```

Commenté [MB24]: Idem as above

```
    type uint8;
    description
      "The IPv6 subnet prefix length expressed in bits.";
  }
  leaf mtu {
    type uint32;
    units "bytes";
    description
      "Maximum size of the Slice Service data packet
       that can traverse an SDP.";
  }
  container ac-tags {
    description
      "Container for the attachment circuit tags.";
    list ac-tag {
      key "tag-type";
      description
        "The attachment circuit tag list.";
      leaf tag-type {
        type identityref {
          base attachment-circuit-tag-type;
        }
        description
          "The attachment circuit tag type.";
      }
      leaf-list value {
        type string;
        description
          "The attachment circuit tag values.
           For example, the tag may indicate
           multiple VLAN identifiers.";
      }
    }
  }
  uses service-qos;
  container sdp-peering {
    description
      "Describes SDP peering attributes.";
    leaf peer-sap-id {
      type string;
      description
        "Indicates a reference to the remote endpoints
         of an attachment circuit. This information can
         be used for correlation purposes, such as
         identifying a service attachment point (SAP)
         of a provider equipment when requesting a
         service with CE based SDP attributes.";
      reference
        "RFC_9408: A YANG Network Data Model for
         _________Service Attachment Points (SAPs)";
    }
    container protocols {
      description
        "Serves as an augmentation target.
         Protocols can be augmented into this container,
         e.g., BGP or static routing.";
    }
  }
```

```
        uses vpn-common:service-status;
      }
    }
    uses vpn-common:service-status;
    container sdp-monitoring {
      config false;
      description
        "Container for SDP monitoring metrics.";
      leaf incoming-bw-value {
        type yang:gauge64;
        units "bps";
        description
          "Indicates the absolute value of the incoming
           bandwidth at an SDP from the customer network or
           from another provider's network.";
      }
      leaf incoming-bw-percent {
        type percentage;
        units "percent";
        description
          "Indicates a percentage of the incoming bandwidth
           at an SDP from the customer network or
           from another provider's network.";
      }
      leaf outgoing-bw-value {
        type yang:gauge64;
        units "bps";
        description
          "Indicates the absolute value of the outgoing
           bandwidth at an SDP towards the customer network or
           towards another provider's network.";
      }
      leaf outgoing-bw-percent {
        type percentage;
        units "percent";
        description
          "Indicates a percentage of the outgoing bandwidth
           at an SDP towards the customer network or towards
           another provider's network.";
      }
    }
  }
}
container connection-groups {
  description
    "Contains connection groups.";
  list connection-group {
    key "id";
    description
      "List of connection groups.";
    leaf id {
      type string;
      description
        "The connection group identifier.";
    }
    leaf connectivity-type {
      type identityref {
        base vpn-common:vpn-topology;
```

```
  }
  default "vpn-common:any-to-any";
  description
    "Connection group connectivity type.";
}
uses service-slo-sle-policy;
/* Per connection group SLO/SLE policy
 * overrides the per Slice SLO/SLE policy.
 */
uses service-slo-sle-policy-override;
list connectivity-construct {
  key "id";
  description
    "List of connectivity constructs.";
  leaf id {
    type uint32;
    description
      "The connectivity construct identifier.";
  }
  choice type {
    default "p2p";
    description
      "Choice for connectivity construct type.";
    case p2p {
      description
        "P2P connectivity construct.";
      leaf p2p-sender-sdp {
        type leafref {
          path "../../../../sdps/sdp/id";
        }
        description
          "Reference to a sender SDP.";
      }
      leaf p2p-receiver-sdp {
        type leafref {
          path "../../../../sdps/sdp/id";
        }
        description
          "Reference to a receiver SDP.";
      }
    }
    case p2mp {
      description
        "P2MP connectivity construct.";
      leaf p2mp-sender-sdp {
        type leafref {
          path "../../../../sdps/sdp/id";
        }
        description
          "Reference to a sender SDP.";
      }
      leaf-list p2mp-receiver-sdp {
        type leafref {
          path "../../../../sdps/sdp/id";
        }
        description
          "Reference to a receiver SDP.";
      }
```

```
              }
              case a2a {
                description
                  "A2A connectivity construct.";
                list a2a-sdp {
                  key "sdp-id";
                  description
                    "List of included A2A SDPs.";
                  leaf sdp-id {
                    type leafref {
                      path "../../../../../sdps/sdp/id";
                    }
                    description
                      "Reference to an SDP.";
                  }
                  uses service-slo-sle-policy;
                }
              }
            }
            uses service-slo-sle-policy;
            /* Per connectivity construct SLO/SLE policy
             * overrides the per slice SLO/SLE policy.
             */
            uses service-slo-sle-policy-override;
            uses vpn-common:service-status;
            container connectivity-construct-monitoring {
              config false;
              description
                "SLO status per connectivity construct.";
              uses connectivity-construct-monitoring-metrics;
            }
          }
          container connection-group-monitoring {
            config false;
            description
              "SLO status per connection group.";
            uses connectivity-construct-monitoring-metrics;
          }
        }
      }
      container custom-topology {
        description
          "Serves as an augmentation target.
           Container for custom topology, which is indicated by the
           referenced topology predefined, e.g., an abstract RFC8345
           topology.";
        uses nw:network-ref;
      }
    }
  }
}
<CODE ENDS>
```

Figure 17: Network Slice Service YANG Module

7. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in these YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/ vulnerability in the "ietf-network-slice-service" module:

* /ietf-network-slice-service/network-slice-services/slo-sle-templates

This subtree specifies the Network Slice Service SLO templates and SLE templates. Modifying the configuration in the subtree will change the related Network Slice Service configuration in the future. By making such modifications, a malicious attacker may degrade the Slice Service functions configured at a certain time in the future.

* /ietf-network-slice-service/network-slice-services/slice-service

The entries in the list above include the whole network configurations corresponding with the Network Slice Service which the higher management system requests, and indirectly create or modify the PE or P device configurations. Unexpected changes to these entries could lead to service disruption and/or network misbehavior.

Some of the readable data nodes in these YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability in the "ietf-network-slice-service" module:

* /ietf-network-slice-service/network-slice-services/slo-sle-templates

Unauthorized access to the subtree may disclose the SLO and SLE templates of the Network Slice Service.

* /ietf-network-slice-service/network-slice-services/slice-service

Unauthorized access to the subtree may disclose the operation status information of the Network Slice Service.

Commenté [MB25]: Service tags exposes privacy data such as customer name, etc. This should be called out as well.

8. IANA Considerations

This document request to register the following URI in the IETF XML registry [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-network-slice-service
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document requests to register the following YANG module in the YANG Module Names registry [~~RFC7950~~RFC6020].

Name: ietf-network-slice-service
Namespace: urn:ietf:params:xml:ns:yang:ietf-network-slice-service
Prefix: ietf-nss
Maintained by IANA?~~:~~ N
Reference: RFC AAAA