GWD-R-P
NSI-WG
nsi-wg@ogf.org

Jeroen van der Ham, UvA
John MacAuley, ESnet
May 27, 2017

Network Service Interface Topology Representation

Status of This Document
Group Working Draft (GWD), candidate Recommendations Proposed (R-P).

This document should be read in conjunction with OGF GFD.212 Network Service Interface Connection Service, v2.0.

Abstract
This document describes Network Service Interface (NSI) Connection Services (CS) use of the Network Markup Language (NML) base schema version 1 (update 1) for modeling NSI topology constructs. In addition, the normative definition of a Service Termination Point (STP) is provided, along with the introduction of Service Definition Documents describing the services offered by a network.

Contents

1 Introduction

The Network Services Interface provides an API that allows applications to monitor, control, interrogate, and support network resources that are made available by the provider of the network. The NSI Connection Service [GFD.212] deals specifically with the request and management of network Connections on transport networks. By design NSI is inherently agnostic to the technology used within the transport plane, instead utilizing high-level service abstractions to provide the basic building blocks needed to model an end-user service across a network. This technology agnostic approach is built into the NSI topology representation and is supported through the use of Service Definitions [GFD.212].

The NSI Connection Service protocol is implemented by Network Service Agent (NSA) entities within the network for the purpose of connection creation on behalf of the end user application. A Connection Service can be requested by any application that has implemented an NSI CS Requester Agent (RA). Similarly, any network provider who has implemented an NSI Provider Agent (PA) can service the request. An Aggregator Agent (AG) combines provider and requester functions along with path finding capabilities to coordinate Connection Service requests between multiple RA and PA entities within the network.

Each service is managed by an exchange of NSI messages between agents. These messages operate using a set of service primitives. Service primitives are the set of instructions that allow the requester to set up and manage a service. Each service request will result in the allocation of a service identifier for the new service instance.

Network Service Agents (NSA) that perform path finding in support of service creation require topology descriptions that model the interconnected network and the services offered by the network. This document describes how the Network Markup Language [GFD.206] is used to model NSI topology concepts in support of the NSI Connection Service [GFD.212] and NSI Signaling and Path Finding [GFD.217].

[Section 3 describes the NSI topology representation extension of the Network Markup Language base schema version 1. Only section 3 and 4, and appendices A and B are normative and considered part of the recommendation. – needs updating]

This document should be read in conjunction with GFD-R.212 Network Service Interface Connection Service version 2.0 [GFD.212], Open Grid forum GFD-I.213, Network Services Framework v2.0 [GFD.213] and OGF GFD-I.217 NSI Signaling and Path Finding [GFD.217].

1.1 Scope and Context

The NSI topology representation is modeled using existing Network Markup Language (NML) schema definitions, a new NSI specific schema definition, and specific NSI object naming schemes. NSI topology covers concepts relevant for supporting the NSI Connection Service that are outside of the scope of NML. Figure 1 shows the relationships between the XML documents defined as part of the NSI protocol suite.

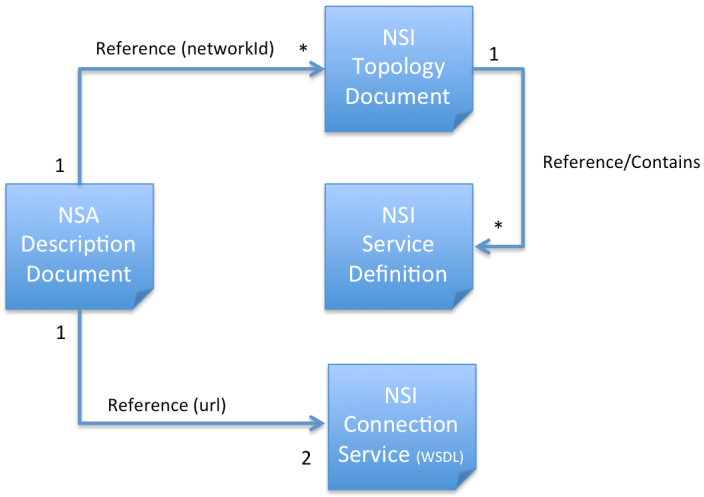


Figure 1 – NSI XML document relationships.

The Network Service Agent (NSA) Description Document [GFD.220] provides metadata in support of NSA self-description. This metadata includes NSA administration and management information, NSI related interface versions and protocol endpoints, NSA features, control plane connectivity, and managed network identifiers for mapping to associated topology documents.

The NSI Topology Document contains network topology described using NML schema and approved schema extensions. Each NSI Topology Document describes a single NSI managed network, and includes a reference to the services supported by that network via the NSI Service Definition [GFD.212].

The NSI Service Definitions document describes the technical aspects of the services being offered by a managed network. Each service is assigned a unique identifier that is then referenced by NSI topology elements, and used in the NSI CS reservation request to identify the type of service being requested.

All three of these documents are then disseminated to NSI actors throughout the network via the NSI Document Distribution Service [NSI-DDS].

The NSI topology representation defined in this document is based on the concepts and requirements described in the GFD-I.213 NSI Framework Document v2.0 [GFD.213], NSI Connection Service version 2.0 [GFD.212], and OGF GFD-I.217 NSI Signaling and Path Finding [GFD.217]. The Network Markup Language (NML) base schema version 1 [GFD.206] with errata applied is used as a syntax for describing the NSI Topology Document. All other mechanisms related to the dissemination and use of NML and NSI topology are out of scope of this document.

2 Notational Conventions

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are to be interpreted as described in [RFC 2119]. Words defined in the glossary are capitalized (e.g. Connection). NSI protocol messages and their attributes are written in camel case and italics (e.g. *reserveConfirmed*).

3 Overview

NSI Topology is a topological representation of the service connection capabilities of the Transport Plane. It exists for two primary purposes: first to facilitate path finding by NSA on behalf of user initiated reservation requests; and second to determine the reservation segment parameters to be sent in a reservation request to PA managing Transport Plane resources involved in a connection. A detailed description of NSI Topology concepts is provided in [GFD.213], Section 4 - "Representing Network Resources", however, in this section a summary is provided to define terminology used throughout the document.

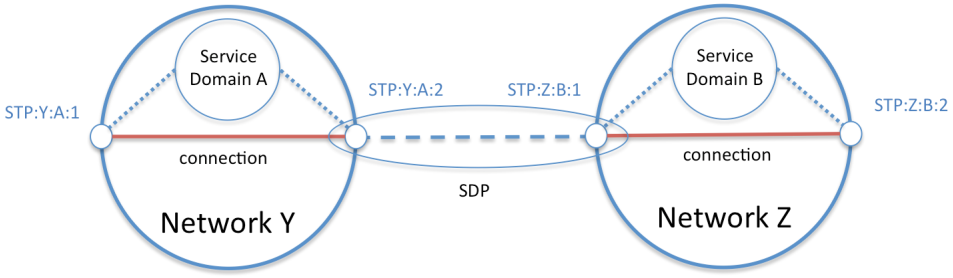


Figure 2 – Inter-Network representation of a Connection.

It is important to note that in its current instantiation NSI Topology represents the "potential" service connection capability. NSI Topology is relatively static and is not updated to reflect network resources no longer available due to an existing or pending service reservation. There is always a possibility that a set of resources present in the current NSI Topology instance may already be consumed, and therefore, unavailable for future reservation requests. As a result it is important for any path finding process to be iterative.

3.1 Service

A service is a "connection" between two points in a Network with certain predefined and dynamically specified characteristics that will deliver a "payload" from Network ingress to Network egress unmodified. Service/connection instances are not currently modeled in NSI Topology and only exist as runtime entities within the NSI Connection Service.

3.2 Service Termination Points

A Service Termination Point (STP) is an abstract reference to a network resource that is capable of terminating an NSI Connection. This is often referred to as the "service endpoint" or "service termination" of a connection. An STP may represent a specific physical object (such as port) or represent an abstract point in a network topology where a connection could logically terminate. Depending on the service being offered an STP could represent a unidirectional or bidirectional service termination. When we refer to an "STP" we are referring to the abstract notion of the endpoint regardless of any physical association.

NSI Topology exposes a set of STP objects that can be used in an NSI Connection Service request to identify the source, destination, and intermediate points of the desired Connection. A prerequisite for an STP is that it must be a member of a Service Domain with a single known Service Type, and both the STP and Service Domain must be members of the same Network. A Network Service Agent (NSA) associated with the network can then advertise the topology for use by pathfinders interconnected on the NSI Service Plane. Note that the choice as to which resources to advertise is subject to local policy of the advertising domain.

In Figure 2 above, STP:Y:A:1 and STP:Y:A:2 are endpoints that are members of the same Service Domain A within Network Y. This meets the criteria for interconnection using the NSI Connection Service. Similarly, STP:Z:B:1 and STP:Z:B:2 are endpoints that are members of the same Service Domain B within Network Z so also meet the criteria for interconnection. In addition, STP:Y:A:2 and STP:Z:B:1 are defined as two ends of a Service Demarcation Point (SDP) so will interconnect the connection segments from Network Y and Network Z providing an end-to-end service in the Transport Plane between endpoints STP:Y:A:1 and STP:Z:B:2.

[Do we want to mention/enforce the restriction of an STP can only be in a single ServiceDomain?]

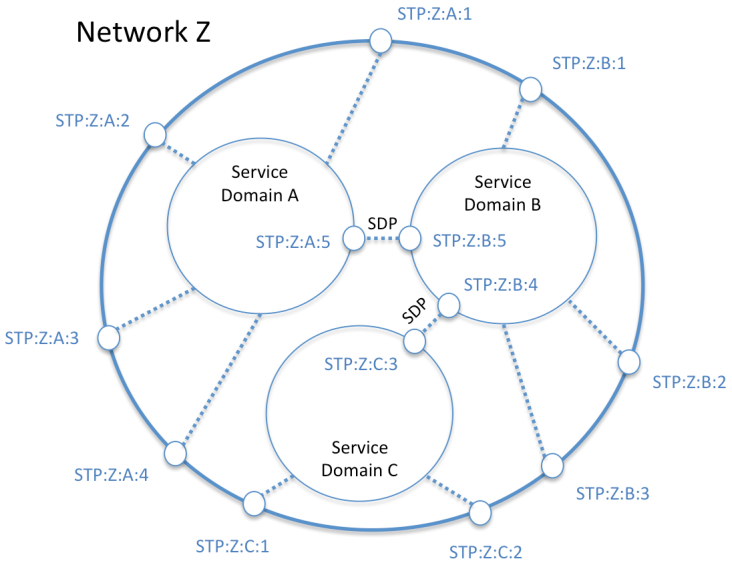


Figure 3 – Service Domains.

3.3 Service Domains

A Service Domain collects a group of STP within a Network the support the same Service Type and which can be fully interconnected without restriction. An associated Service Definition (SD) describing the service being offered defines the Service Type of a Service Domain. Service Domains are grouped into Network topologies that can be advertised by at most one NSA; however, an NSA can advertise multiple Network topologies.

As shown in Figure 3, multiple Service Domains within a single network can be used to model internal network structure if desired. In this case, externally visible STP are used to form inter-network SDP between peer networks, while internal STP are used to form internal SDP to model intra-network topology between Service Domains within a single network.

Using Figure 3 as an example, an external path finder could issue a request to connect external STP:Z:A:3 to STP:Z:B:2 and delegate internal path finding to the uPA managing the network, or if the path finder would like to provide additional guidance, it could specify a more detailed path such as STP:Z:A:3 to STP:Z:A:5 and STP:Z:B:5 to STP:Z:B:2.

In a Service Domain any STP can be connected to any other STP with the following constraints:

- Unidirectional inbound STP can only be connected to unidirectional outbound STP.
- Bidirectional ports can only be interconnected to other bidirectional ports.

3.4 Service Definitions

A document that describes the predefined service characteristics and data elements associated with a service being offered by a Network. A Service Domain referencing a Service Definition must support the associated Service Type for all member STP. See [GDF-212] for a more detailed description.

3.5 Service Type

A predefined identifier that uniquely identifies the type of service offered by a Service Domain and is associated with a Service Definition document. The Service Type identifier is specified in the NSI Connection Service *reserve* operation to uniquely identify the service type being requested. This will also dictate the service parameters included in the *reserve* operation.

3.6 Networks

A Network is a grouping of related NSI Topology objects (STP, Service Domains, Service Definitions, Adaptations, etc.) sharing a common network identifier and managed by a single NSA. An NSA may be associated with multiple Networks, but a Network can only be associated with a single NSA.

3.7 Adaptations

An Adaptation defines the (de) encapsulation or (de) adaptation of one Service Type into another Service Type if the Network is capable of offering the service. Furthermore, an Adaptation has directionality i.e. adaptation and de-adaptation. Both unidirectional and bidirectional Adaptations are supported, with bidirectional Adaptations containing a symmetrical pair of adaptation and de-adaptation functions.

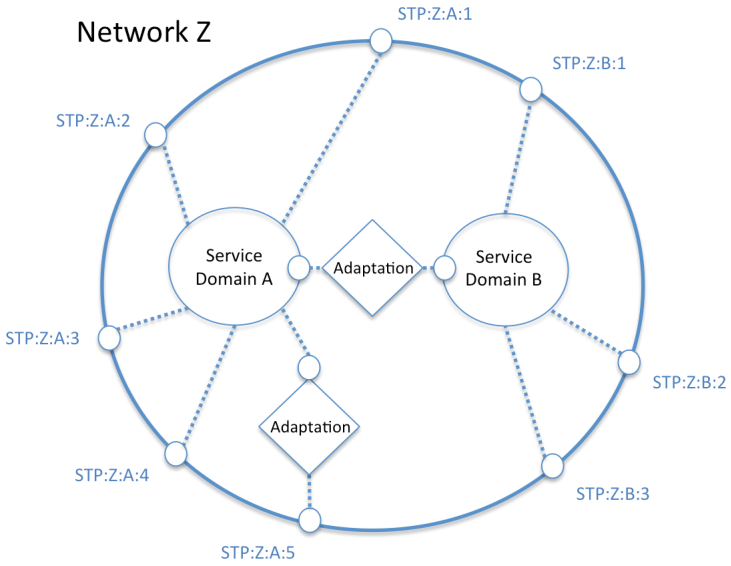


Figure 4 – Service Adaptations.

The Adaptation can be placed between two Service Domains of different Service Types within the same Network if that Network supports the ability to adapt between the two services, or the Adaptation can be placed between an inter-domain STP and Service Domain if the service adaptation occurs on ingress/egress to the Network. An Adaptation can also be defined between

STPs of the same service type in the case where encapsulation/adaptation of the input service type results in the same output service type.

3.8 Service Demarcation Point (SDP)

SDP are formed when a pair of STP of matching capabilities is defined as adjacent (and connectable) between two Service Domains. The formation of an SDP between two Service Domains is an agreement to carry service "payload" from the egress STP in the first Service Domain to the ingress STP in the second Service Domain. The SDP concept is show in both Figure 2 and Figure 3 above.

4 Terminology

NSI Topology concepts are implemented using standard NML topology schema [GFD.206] with NSI specific extensions, and constrained naming rules for object identifiers. The following table maps NSI constructs to the equivalent NML objects.

NSI Construct	NML Construct
STP	Port, PortGroup, and BidirectionalPort objects.
Service Domain	SwitchingService object.
SDP	isAlias relationship on Port/PortGroup pointing to peer Port/PortGroup.
Service Definition	Annotation on SwitchingService object + independent Service Definition XML document.
Service Type	A URL attribute of Service Definition providing a unique Service Type identifier.
Adaptation	AdaptationService and DeadaptationService objects.
Service	Currently not modeled in NML.

Table 1 – NSI to NML topology mapping.

Figure 5 below shows a model representation of the NSI to NML mapping with NML relationships shown where possible.

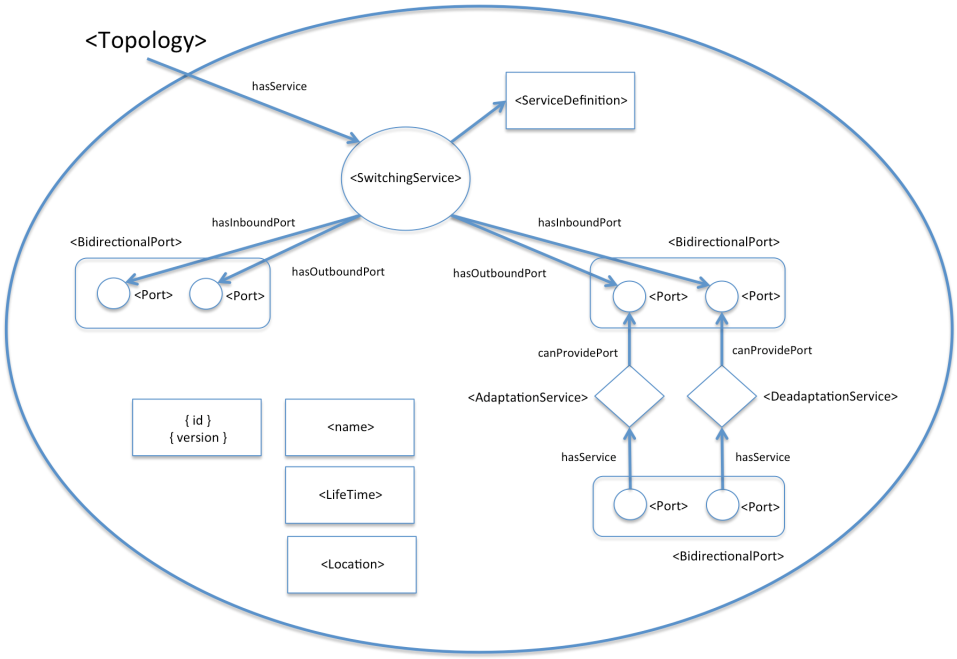


Figure 5 – NML schema used by NSI Topology.

5 NSI Topology Identifier Format

An STP identifier (*stpId*) is used to identify the source and destination service endpoints of a connection request within NSI Connection Services. This STP identifier and all other NSI topology identifiers must be globally unique, such that no two topology constructs have the same identifier. The recommended way of constructing such a topology identifier is by using the network resource URN (NURN) identified by the "urn:ogf:network:" prefix as defined in [GFD.202]. By using the NURN construct, combined with specific naming rules defined here, every NSI topology element can be uniquely identified in a global namespace.

The NSI Connection Service specification [GFD.212] puts a specific requirement on the Service Termination Point Identifier (*stpId*), defining the *stpId* as a three-part identifier composed of a network identifier part (*networkId*), a local identifier part (*localId*), and a qualifying label part:

```
<STP identifier> ::= <networkId> ":" <localId> <label>
<label> ::= "?" <labelType> "=" <labelValue> | "?"<labelType> | ""
<labelType> ::= <string>
<labelValue> ::= <string>
```

To maintain syntactic correctness with the NURN specification such that an *stpId* is also a valid NURN formatted URN, we define a specific Augmented BNF [RFC 5234] to restrict the format of the NSI Network and STP identifiers such that they remain NURN compliant, but provide the needed imbedded structure for separating the *networkId* and *localId* components. To do this we must also relax the following NURN restriction: "**OPAQUE-PART** is opaque, and must not be parsed or interpreted by any organization except for the organization that assigned the URN." NSAs within the connected control plane are permitted, when needed, to parse an NURN into component parts as defined below.

```
NSAID = "urn:ogf:network:" ORGID ":" LOCALPART
NETWORKID = "urn:ogf:network:" ORGID ":" LIMITEDLOCALPART
STPID = NETWORKMEMBER
SERVICEDOMAINID = NETWORKMEMBER
SERVICEADAPTATIONID = NETWORKMEMBER
SERVICEDEFINITIONID = NETWORKMEMBER

ORGID = FQDN ":" DATE ; ID of assigning organisation
FQDN = 1*(ALPHA / DIGIT / "-" / ".") ; Domain name
DATE = YEAR *1(MONTH *1DAY) ; Date of creation of ORGID
YEAR = 4DIGIT
MONTH = 2DIGIT
DAY = 2DIGIT
LIMITEDLOCALPART = *(ALPHA / DIGIT / SYMBOL)
ALPHA =  %x41-5A / %x61-7A   ; A-Z / a-z
DIGIT =  %x30-39 ; 0-9
SYMBOL = "+" / "," / "-" / "." / ";" / "=" / "_"

NETWORKMEMBER = NETWORKID ":" LOCALPART *1QUERY *1FRAGMENT
LOCALPART = *(ALPHA / DIGIT / OTHER)
OTHER = ALLOWED / EXTENSION
ALLOWED = "+" / "," / "-" / "." / ":" / ";" / "=" / "_"
EXTENSION = "!" / "$" / "(" / ")" / "*" / "@" / "~" / "&"
QUERY = "?" *QFCHAR
FRAGMENT = "#" *QFCHAR
QFCHAR = ALPHA / DIGIT / OTHER
```

The full length of an ID must not exceed 255 characters as defined in [GFD202].

ALLOWED and **SYMBOL** characters may be used for the assignment of NURNs. **EXTENSION** characters should not be used to assign NURNs. To allow for future extensions, parsers should accept NURNs with **EXTENSION** characters as per [GFD202].

The **QUERY** part is used by NSI to specify technology specific labels as part of the NURN as per [GFD.212].

The **FRAGMENT** parts must not be present in any assigned NURN and is reserved for future standardization as per [GFD.202].

An NURN must not contain percentage-encoded characters ("%" HEXDIG HEXDIG). It should also be noted that the following characters (which are either allowed by the URI or URN specification) must not be used in the **LOCALPART** or **LIMITEDLOCALPART** of an NURN: "%", "/", "?", "#", and "".

DOMAIN is a fully qualified domain name (FQDN) of the URN assigning organization in LDR format [RFC5890]. Valid examples are example.net and example.xn--jxalpdlp.

DATE is a date (either year, year+month or year+month+day). The combination of **DOMAIN** and **DATE** forms the organization identifier, **ORGID**.

5.1 Examples

The following are examples of legal network identifiers:

1. urn:ogf:network:grnet.gr:2013:topology
2. urn:ogf:network:surfnet.nl:1990:production7
3. urn:ogf:network:caltech.edu:2014:

The following are examples of illegal network identifiers:

```
1. urn:ogf:network:es.net:2013
2. urn:ogf:network:cipo.rnp.br:
3. urn:ogf:network:example.net:2013:production:east
```

In example #1 there is no following colon after the **DATE** component. This is required even when there is no **LIMITEDLOCALPART**. In example #2 there is no required **DATE** component. In example #3 the **LIMITEDLOCALPART** contains a colon ":" which is not a member of the character set.

The following are examples of legal STP identifiers for a network identifier of "urn:ogf:network:example.net:2013:topology":

```
1. urn:ogf:network:example.net:2013:topology:CLIENT_port
2. urn:ogf:network:example.net:2013:topology:EAST_PORT_TO_CUSTOMER:in
3. urn:ogf:network:example.net:2013:topology:PORT+TO-CUSTOMER:east:bidirectional
```

The following are examples of legal STP identifiers for a network identifier of "urn:ogf:network:example.net:2013:":

```
1. urn:ogf:network:example.net:2013::CLIENT_port
2. urn:ogf:network:example.net:2013::EAST_PORT_TO_CUSTOMER:in
3. urn:ogf:network:example.net:2013::PORT+TO-CUSTOMER:east:bidirectional?vlan=1791
```

The following are examples of illegal STP identifiers for a network identifier of "urn:ogf:network:example.net:2013:topology":

```
1. urn:ogf:network:example.net:2013:client_port
2. urn:ogf:network:example.net:2013::client_port
2. urn:ogf:network:example.net:2013:topology:client_port:xe-8/2/0:*
```

In example #1 and #2 the network identifier's **LIMITEDLOCALPART** is missing from the STP identifier. In example #3 the "/" and "*" characters are illegal **LOCALPART** characters based on the NURN definition.

6 NML document structure

NSI Topology is described in an XML instance document following the rules of NML schema. An NSI Topology document following the rules of the NML schema will consists of six sections:

- Namespace definitions.
- Document identification information consisting of a unique document *id*, a document *version*, common *name*, and a document *Lifetime*.
- *BidirectionalPort* definitons.
- NSI *serviceDefinitions*.
- Service Relations (SwitchingService, Adaptations, etc.).
- *Inbound* and *Outbound* unidirectional Port definitions.

The number of sections and the content of each section may change over time as additional NSI capabilities are modeled using NML Topology. The following is a simple instance representation of the *nml:Topology* element:

```
<nml:Topology id="" version="" xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#">
  <nml:name>{0,1}</nml:name>
```

```
  <nml:Lifetime>{0,1}</nml:Lifetime>
  <nml:Location id="">{0,1}</nml:Location>
  <nml:Link encoding="" id="" noReturnTraffic="" version="">{0,unbounded}</nml:Link>
  <nml:Port encoding="" id="" version="">{0,unbounded}</nml:Port>
  <nml:Node id="" version="">{0,unbounded}</nml:Node>
  <nml:SwitchingService encoding="" id="" labelSwapping="" labelType="" version="">
        {1,1}
  </nml:SwitchingService>
  <nml:AdaptationService adaptationFunction="" id="" version="">
        {1,1}
  </nml:AdaptationService>
  <nml:DeadaptationService adaptationFunction="" id="" version="">
        {1,1}
  </nml:DeadaptationService>
  <nml:Topology id="" version="">{1,1}</nml:Topology>
  <nml:PortGroup encoding="" id="" version="">{1,1}</nml:PortGroup>
  <nml:LinkGroup id="" version="">{1,1}</nml:LinkGroup>
  <nml:BidirectionalPort id="" version="">{1,1}</nml:BidirectionalPort>
  <nml:BidirectionalLink id="" version="">{1,1}</nml:BidirectionalLink>
  <nml:Relation type="">{0,unbounded}</nml:Relation>
</nml:Topology>
```

6.1 A note to the reader

There is an attribute naming inconsistency within the NML schema that has caused confusion in the past within NSA implementations and their generated NML Topology documents. NML uses standard camel case naming rules:

- Names of type definitions are capitalized using upper camel case rules e.g. `PortType`.
- Names of elements are capitalized using upper camel case rules e.g. `PortGroup`.
- Names of attributes are capitalized using lower camel case rules e.g. `labelSwapping`.

Unfortunately, there was one error in the NML schema where the use of lower camel case is inconsistent for the same attribute name present in two different type definitions:

- In the SwitchingServiceType type definition there is an attribute called "`labelType`".
- In the LabelType type definition there is an attribute called "`labeltype`".

These two attributes have different capitalization but represent the same value. Within this document an attempt is made to use the capitalization of the attribute name as it appears in the schema when referring to the attribute in question. This should not be considered an error.

6.2 Namespace definitions

The outer *Topology* element contains namespace definitions used by NML and NSI to define XML elements within the instance document. The namespace definitions used in a standard NSI topology document are:

NML base schema namespace:

```
xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#"
```

NML Ethernet schema namespace:

```
xmlns:eth="http://schemas.ogf.org/nml/2012/10/ethernet"
```

NSI Service Definition schema namespace:

```
xmlns:sd="http://schemas.ogf.org/nsi/2013/12/services/definition"
```

Ethernet schema namespace:

```
xmlns:eth="http://schemas.ogf.org/nml/2012/10/ethernet"
```

These namespaces will be used throughout this document to identify the source of the element/attribute definition. It should be noted that other namespaces could be defined within the NSI Topology document in addition to these three. Unknown namespaces and associated elements should be ignored by the NSA and not considered an error.

6.3 Network Identifier

An NSI Topology is named by a globally unique network identifier, or *networkId*, which maps to a managing NSA via metadata stored in the NSA Description Document [GFD.220]. This root *networkId* is part of all named topology components within the associated NML document. The *networkId* is stored in the *id* attribute of the enclosing NML *Topology* element (inherited from *nml:NetworkObject*) as follows:

```
<nml:Topology
        xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#"
        xmlns:eth="http://schemas.ogf.org/nml/2012/10/ethernet"
        xmlns:sd="http://schemas.ogf.org/nsi/2013/12/services/definition"
    id="urn:ogf:network:netherlight.net:2013:production7" version="2016-09-16T18:16:04.560Z">
```

6.4 Document Version

An NSI Topology is versioned using an *xsd:dateTime* string representing the time the topology document was generated. This *xsd:dateTime* string will be used to compare versions, with newer versions having an later date than older versions. Each time a new document version is generated a new version string must be generated to indicate a change in topology. The version of an NML document is stored using the NML *version* attribute in the NML *Topology* element (inherited from *nml:NetworkObject*) as follows:

```
<nml:Topology
        xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#"
        xmlns:eth="http://schemas.ogf.org/nml/2012/10/ethernet"
        xmlns:sd="http://schemas.ogf.org/nsi/2013/12/services/definition"
    id="urn:ogf:network:netherlight.net:2013:production7" version="2016-09-16T18:16:04.560Z">
```

Version is formatted as ISO 8601 calendar date, and should be a basic (compact) representation with UTC timezone (YYYYMMDDThhmmss Z) [ISO8601].

6.5 Topology Name

An NSI Topology document contains a *name* element that must be populated with a user visible string identifying the topology. This string can be used for display purposes. The NML Topology *nml:name* element (inherited from *nml:NetworkObject*) is populated as follows:

```
<nml:Topology
        xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#"
        xmlns:eth="http://schemas.ogf.org/nml/2014/01/ethernet"
        xmlns:sd="http://schemas.ogf.org/nsi/2013/12/services/definition"
    id="urn:ogf:network:netherlight.net:2013:production7" version="2016-09-13T17:52:21.000Z">
```

```
    <!-- User friendly and displayable name for the Network. -->
    <nml:name>SURFnet production7</nml:name>
       ...
</nml:Topology>
```

6.6 Lifetime

An NSI Topology document has a Lifetime that defines the interval between which the document is said to be active or valid. This *nml:Lifetime* element is defined for the entire topology document (inherited from *nml:NetworkObject*) and not the individual topology objects within the document. It is possible to assign one of more *nml:Lifetime* elements to individual topology objects but this use is beyond the scope of this document. The NSI Topology document should not be used if deemed inactive by the *nml:Lifetime* element.

```
<nml:Topology
        xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#"
        xmlns:eth="http://schemas.ogf.org/nml/2014/01/ethernet"
        xmlns:sd="http://schemas.ogf.org/nsi/2013/12/services/definition"
    id="urn:ogf:network:netherlight.net:2013:production7" version="2016-09-13T17:52:21.000Z">

    <!-- User friendly and displayable name for the Network. -->
    <nml:name>SURFnet production7</nml:name>
    <nml:Lifetime>
        <nml:start>2016-09-13T17:52:21.000Z</nml:start>
        <nml:end>2016-11-13T17:52:16.326Z</nml:end>
    </nml:Lifetime>
       ...
</nml:Topology>
```

A *nml:Lifetime* element may have the following attributes:

- *start* is the start time and date formatted as ISO 8601 calendar date, and should be a basic (compact) representation with UTC timezone (YYYYMMDDThhmmss Z) [ISO 8601].
- *end* is the end time and date formatted as ISO 8601 calendar date, and should be a basic (compact) representation with UTC timezone (YYYYMMDDThhmmss Z) [ISO 8601].

6.7 The Service Termination Point

An NSI Topology document models a Service Termination Point (STP) using three separate constructs. A single unidirectional STP is modeled using the *nml:Port* element. This single port can be defined with an associated single value label using the *nml:Label* element if desired. A bidirectional STP is then modeled using an *nml:BidirectionalPort* element referencing paired outbound and inbound unidirectional *nml:Port* elements. This is shown below in Figure 6.

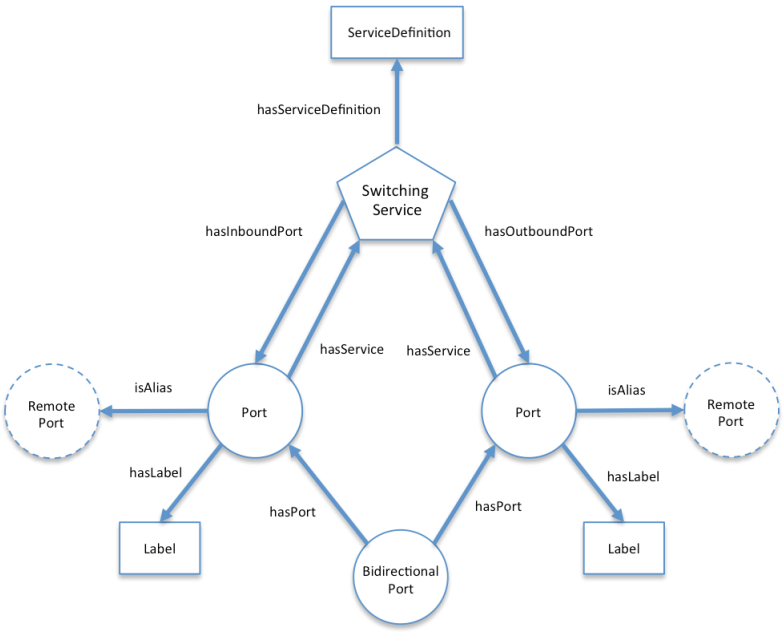


Figure 6 – Modeling an STP in NML using a Port element.

In some cases an STP will be associated with a set of technology labels. Rather than describing each of these available labels as individual STPs, we introduce the STP Bundle as a way to describe a set of related STP within the NSI Topology. A unidirectional STP bundle is modeled using the *nml:PortGroup* element with an associated multi-valued label using the *nml:LabelGroup* element. A bidirectional STP bundle is then modeled using an *nml:BidirectionalPort* element referencing paired outbound and inbound unidirectional *nml:PortGroup* elements. This is shown below in Figure 7.

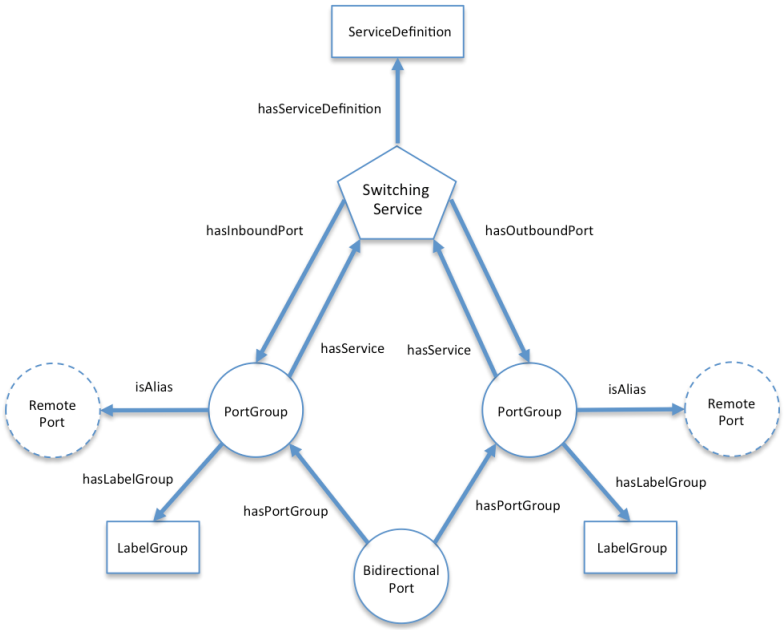


Figure 7 - Modeling an STP bundle in NML using a PortGroup element.

6.7.1 Defining a unidirectional STP

In NML we use the *nml:Port* or *nml:PortGroup* elements along with the *hasInboundPort* and *hasOuboundPort* relations to model unidirectional STP.

```
<nml:Relation type="" xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#">
  <nml:Port encoding="" id="" version="">
    <nml:name>{0,1}</nml:name>
    <nml:Lifetime>{0,1}</nml:Lifetime>
    <nml:Location id="">{0,1}</nml:Location>
    <nml:Label labeltype="">{0,1}</nml:Label>
    <nml:Relation type="">{0,unbounded}</nml:Relation>
  </nml:Port>
</nml:Relation>

<nml:Relation type="" xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#">
  <nml:PortGroup encoding="" id="" version="">
    <nml:name>{0,1}</nml:name>
    <nml:Lifetime>{0,1}</nml:Lifetime>
    <nml:Location id="">{0,1}</nml:Location>
    <nml:LabelGroup labeltype="">{0,unbounded}</nml:LabelGroup>
    <nml:Port encoding="" id="" version="">{0,unbounded}</nml:Port>
    <nml:PortGroup encoding="" id="" version="">{0,unbounded}</nml:PortGroup>
    <nml:Relation type="">{0,unbounded}</nml:Relation>
  </nml:PortGroup>
</nml:Relation>
```

For an inbound *nml:Port* or *nml:PortGroup* the *type* attribute within the *nml:Relation* element is set to:

```
type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort"
```

For an outbound port the *type* attribute is set to:

```
type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort"
```

NSI Topology uses the following attributes/elements from the *nml:Port/nml:PortGroup* elements:

encoding
: (optional) Assigned the technology encoding of the port. In NSI we use this attribute to identify the abstract type of STP for matching rules within the *SwitchingService* element when generating *ServiceDomain* constructs. Encoding is not required if all *SwitchingService* elements are populated with their corresponding *Port/PortGroup* elements. For Ethernet services the following encoding URL will be used:

```
encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
```

id
: Assigned the unique *stpId* of the STP this *Port/PortGroup* is modeling.

version
: Although use is not currently defined in NSI topology, this attribute can hold a unique version identifier allowing for multiple versions of the same *Port/PortGroup* element to be present in the NML topology document. This would need to be used in concert with the *Lifetime* element to provide a time sequence of when a version of the STP is active.

nml:name
: (optional) A user visible string for the STP.

nml:Lifetime
: (optional) Although use is not currently defined in NSI topology, this element could be used to provide a limited availability for the STP in the context of the NML topology document.

nml:Location
: (optional) A geographical location for the STP.

nml:Label/nml:LabelGroup
: (optional) A set of one or more labels used to generate a range of STP. The *labeltype* attribute will use the following URL if supporting IEEE 802.1Q Ethernet:

```
labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan"
```

The following *nml:Port* element definitions define two unidirectional STP of the specified *encoding* type:

```
<!-- The hasInboundPort relationship models a single inbound unidirectional
     port that is mapped to the NSI unidirectional STP object. -->
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
    <nml:Port encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
        id="urn:ogf:network:netherlight.net:2013:production7:uva-in"/>
</nml:Relation>

<!-- The hasOutboundPort relationship models a single outbound unidirectional
     port that is mapped to the NSI unidirectional STP object. -->
```

```
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
    <nml:Port encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
        id="urn:ogf:network:netherlight.net:2013:production7:uva-out"/>
</nml:Relation>
```

Similarly, the following *nml:PortGroup* element represents the same STP as the *nml:Port* element definition:

```
<!-- The hasInboundPort relationship models a single inbound unidirectional
     port that is mapped to the NSI unidirectional STP object. -->
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
    <nml:PortGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
        id="urn:ogf:network:netherlight.net:2013:production7:uva-in"/>
</nml:Relation>

<!-- The hasOutboundPort relationship models a single outbound unidirectional
     port that is mapped to the NSI undirectional STP object. -->
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
    <nml:PortGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
        id="urn:ogf:network:netherlight.net:2013:production7:uva-out"/>
</nml:Relation>
```

From these two NML port definitions we would generate the following unidirectional STP identifiers:

```
urn:ogf:network:netherlight.net:2013:production7:uva-in
urn:ogf:network:netherlight.net:2013:production7:uva-out
```

Multiple inbound (or outbound) STP *nml:Port* declarations can be grouped into a single *hasInboundPort* (or *hasOutboundPort*) *nml:Relation* element if desired, or can be defined as individual *nml:Relation* elements.

6.7.2 Defining a unidirectional STP with a label

Endpoints in a network often have a technology label associated with them such as a VLAN or a wavelength. If an STP has a single label, we can we can annotate the previous *nml:Port* element with a *nml:Label* element allowing for the specification of a single label:

```
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
    <nml:Port id="urn:ogf:network:netherlight.net:2013:production7:uva-in">
        <Label labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">200</Label>
    </Port>
</nml:Relation>

<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
    <nml:Port id="urn:ogf:network:netherlight.net:2013:production7:uva-out">
        <Label labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">200</Label>
    </Port>
</nml:Relation>
```

From these two *nml:Port* definitions we would generate the following unidirectional STP identifiers with the *nml:Label* annotated as a query parameter on the STP URN:

```
urn:ogf:network:netherlight.net:2013:production7:uva-in?vlan=200
urn:ogf:network:netherlight.net:2013:production7:uva-out?vlan=200
```

A similar definition can be created using the *nml:PortGroup* element.

6.7.3 Defining STP Bundles

In some cases an STP will be associated with a set of technology labels. Rather than describing each of these available labels as individual STPs, we introduce the STP bundle as a way to describe a set of related STP within the NSI Topology. A *nml:LabelGroup* supports comma and hyphen separated ranges such as “1-1770,1780-2000,2002,2006”.

For the STP bundle we use the *nml:PortGroup* element:

```
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
    <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-in">
        <LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
            1000-1019
        </LabelGroup>
    </PortGroup>
</nml:Relation>

<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
    <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-out">
        <LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
            1000-1019
        </LabelGroup>
    </PortGroup>
</nml:Relation>
```

From these two *nml:PortGroup* definitions we would generate individual unidirectional STP definitions for each of the labels specified in the *nml:LabelGroup* element.

```
urn:ogf:network:netherlight.net:2013:production7:nordunet-in?vlan=1000
urn:ogf:network:netherlight.net:2013:production7:nordunet-in?vlan=1001
urn:ogf:network:netherlight.net:2013:production7:nordunet-in?vlan=1002
.
.
.
urn:ogf:network:netherlight.net:2013:production7:nordunet-out?vlan=1019
```

and

```
urn:ogf:network:netherlight.net:2013:production7:nordunet-out?vlan=1000
urn:ogf:network:netherlight.net:2013:production7:nordunet-out?vlan=1001
urn:ogf:network:netherlight.net:2013:production7:nordunet-our?vlan=1002
.
.
.
urn:ogf:network:netherlight.net:2013:production7:nordunet-out?vlan=1019
```

Multiple inbound (or outbound) STP *nml:PortGroup* declarations can be grouped into a single *hasInboundPort* (or *hasOutboundPort*) *nml:Relation* element if desired, or can be defined as individual *nml:Relation* elements.

6.7.4 Bidirectional STP

In NML we use the *nml:BidirectionalPort* element to group *nml:Port* and *nml:PortGroup* definitions into bidirectional STP.

```
<nml:BidirectionalPort id="" version=""
        xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#">
```

```
  <nml:name>{0,1}</nml:name>
  <nml:Lifetime>{0,1}</nml:Lifetime>
  <nml:Location id="">{0,1}</nml:Location>
  <nml:Port encoding="" id="" version="">{1,1}</nml:Port>
  <nml:Port encoding="" id="" version="">{1,1}</nml:Port>
  <nml:PortGroup encoding="" id="" version="">{1,1}</nml:PortGroup>
  <nml:PortGroup encoding="" id="" version="">{1,1}</nml:PortGroup>
</nml:BidirectionalPort>
```

NSI Topology uses the following attributes/elements from the *nml:BidirectionalPort* element:

id
: Assigned the unique *stpId* of the STP this *BidirectionalPort* is modeling.

version
: (optional) Use is not currently defined in NSI topology. This attribute can hold a unique version identifier allowing for multiple versions of the same *nml:BidirectionalPort* element to be present in the NML topology document. This would need to be used in concert with the *nml:Lifetime* element to provide a time sequence of when a version of the STP is active.

nml:name
: (optional) A user visible string for the STP.

nml:Lifetime
: (optional) Use is not currently defined in NSI topology. This element could be used to provide a limited availability for the STP in the context of the NML topology document.

nml:Location
: (optional) A geographical location for the STP.

```
<!-- BidirectionalPort elements map to the NSI bidirectional STP object. -->
<nml:BidirectionalPort id="urn:ogf:network:netherlight.net:2013:production7:uva">
  <!-- User friendly and displayable name for the STP object. -->
  <nml:name>Asd001A_8700_07 4/1 UvA (SNE)</nml:name>

  <!-- This bidirectional port is composed of two individual unidirectional
       PortGroup objects. -->
  <nml:Port id="urn:ogf:network:netherlight.net:2013:production7:uva-in"/>
  <nml:Port id="urn:ogf:network:netherlight.net:2013:production7:uva-out"/>
</nml:BidirectionalPort>
```

Here is the a similar *nml:BidirectionalPort* definition but using *nml:PortGroup* member references based on the element definition used for the member unidirectional STP.

```
<!-- BidirectionalPort elements map to the NSI bidirectional STP object. -->
<nml:BidirectionalPort id="urn:ogf:network:netherlight.net:2013:production7:uva">
  <!-- User friendly and displayable name for the STP object. -->
  <nml:name>Asd001A_8700_07 7/2 NORDUnet (nl-sar2-nordunet xe-0/0/3)</nml:name>

  <!-- This bidirectional port is composed of two individual unidirectional
       PortGroup objects. -->
  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-in"/>
  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-out"/>
</nml:BidirectionalPort>
```

6.7.5 Underspecified STP

Within the NSI protocol itself the concept of an underspecified STP was introduced to allow for a range of labels to be specified in a reservation request using a single STP URN. This is accomplished by encoding the desired label range in the query string portion of the URN as follows:

```
urn:ogf:network:netherlight.net:2013:production7:nordunet?vlan=1000-1019
```

When these underspecified STP are present in a reservation request a Network Service Agent is free to pick any individual STP from the specified label range. The confirmation back to the requester will contain the fully specified STP selected by the NSA for the request.

If a label schema is defined as containing a set of values, but the underspecified STP contains the label with no values as follows:

```
urn:ogf:network:netherlight.net:2013:production7:nordunet?vlan
```

Then a Network Service Agent is free to pick any valid label value supported by that root STP.

The behavior defined when a root STP is specified with no label or label values is specific to the Service Type of the STP. For example, if the following STP represented a tagged Ethernet service, then specifying the root STP could request all contents on the underlying Ethernet port or only untagged traffic.

```
urn:ogf:network:netherlight.net:2013:production7:nordunet
```

The specific behaviors would be defined in the Service Definition document associated with the STP's Service Type.

6.8 Service Demarcation Point

SDP are formed when a pair STPs of matching capabilities are considered adjacent. The *nml:Port* and *nml:PortGroup* elements may contain an *isAlias* relationship indicating connectivity to another *nml:Port* or *nml:PortGroup*. NSI Topology uses the *isAlias* relationship to create an SDP between adjacent STP derived from the *nml:Port* or *nml:PortGroup* elements. The rules for creating SDP from a set of STP are specific to the type of STP (encoding) and associated labels. This will be explained in more detail using the following example using *nml:PortGroup* definitions from two different network topologies:

```
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
  <nml:PortGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
      id="urn:ogf:network:netherlight.net:2013:production7:esnet-in">
    <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
      1000-1019
    </nml:LabelGroup>
    <!-- This isAlias relationship identifies the outbound unidirectional port
         connected on the far end of this port. In this specific case the port
         is hosted in a different network. -->
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
      <nml:PortGroup id="urn:ogf:network:es.net:2013::amst-cr5:3_1_1:+:out"/>
    </nml:Relation>
  </nml:PortGroup>
</nml:Relation>

<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
  <nml:PortGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
```

```
      id="urn:ogf:network:netherlight.net:2013:production7:esnet-out">
    <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
      1000-1019
    </nml:LabelGroup>
    <!-- This isAlias relationship identifies the inbound unidirectional port
         connected on the far end of this port. In this specific case the port
         is hosted in a different network. -->
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
      <nml:PortGroup id="urn:ogf:network:es.net:2013::amst-cr5:3_1_1:+:in"/>
    </nml:Relation>
  </nml:PortGroup>
</nml:Relation>
```

The above *isAlias* definition states that port `urn:ogf:network:netherlight.net:2013:production7:esnet-in` is connected to port `urn:ogf:network:es.net:2013::amst-cr5:3_1_1:+:out`, and similarly, port `urn:ogf:network:netherlight.net:2013:production7:esnet-out` is connected to `urn:ogf:network:es.net:2013::amst-cr5:3_1_1:+:in`. However, for an SDP to be formed we have to have STP generated from port definitions on from both network topologies declare the same *isAlias* relationship as with this example from the peer:

```
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
  <nml:PortGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
      id="urn:ogf:network:es.net:2013::amst-cr5:3_1_1:+:in">
    <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
      1000-1019
    </nml:LabelGroup>
    <!-- This isAlias relationship identifies the outbound unidirectional port
         connected on the far end of this port. In this specific case the port
         is hosted in a different network. -->
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:esnet-out"/>
    </nml:Relation>
  </nml:PortGroup>
</nml:Relation>

<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
  <nml:PortGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
      id="urn:ogf:network:es.net:2013::amst-cr5:3_1_1:+:out">
    <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
      1000-1019
    </nml:LabelGroup>
    <!-- This isAlias relationship identifies the inbound unidirectional port
         connected on the far end of this port. In this specific case the port
         is hosted in a different network. -->
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:esnet-in"/>
    </nml:Relation>
  </nml:PortGroup>
</nml:Relation>
```

STP of type "*http://schemas.ogf.org/nml/2012/10/ethernet*" related via an *isAlias* element have the following rules for creating associated SDP:

1. If no Labels are defined then a single SDP is created between the two STP.
2. If a matching label is present in each of the two *nml:Port* definitions, then these two STP with matching labels can form an SDP.
3. If one or more labels are present in each of two *nml:PortGroup* definitions, only STP with matching labels can form an SDP (intersection of the Inbound and Outbound port labels is performed).

6.9 Service Domain (SD)

A Service Domain is modeled through the use of the *nml:SwitchingService* element that describes the ability to create new Links from any of its inbound Ports to any of its outbound Ports. In NSI Topology it is used to describe the ability to interconnect any member inbound STP to any member outbound STP for the associated Service Definition.

A single *nml:SwitchingService* declaration can expand into many Service Domains depending if label swapping is supported or not by the network (a domain per label value if not). Figure 8 below shows this mapping based on *labelSwapping* support.

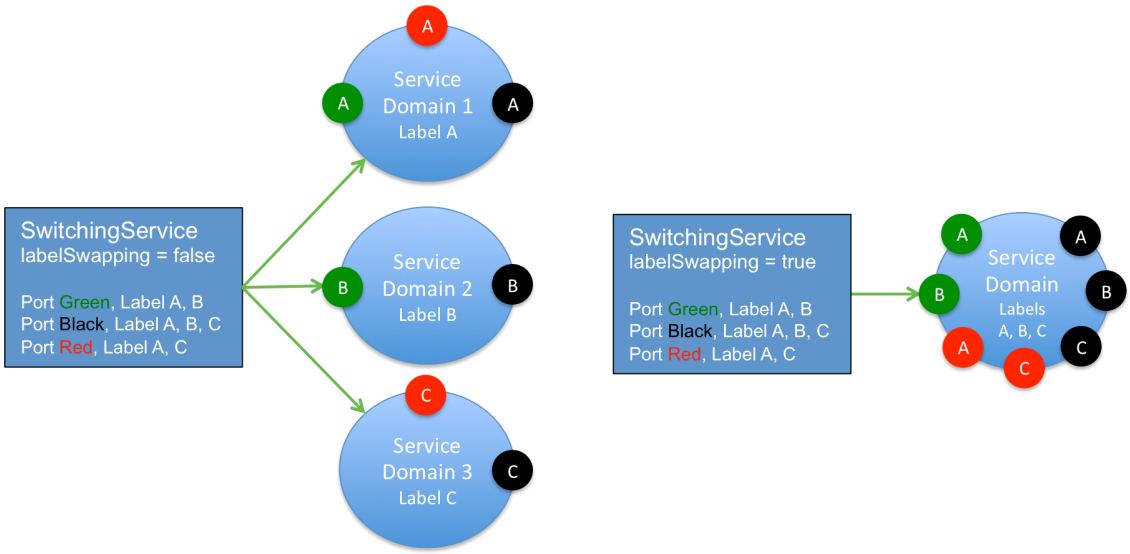


Figure 8 – SwitchingService to Service Domain mapping.

The following is a simple instance representation of the *nml:SwitchingService* element:

```
<nml:SwitchingService encoding="" id="" labelSwapping="" labelType="" version=""
      xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#">
  <nml:name>{0,1}</nml:name>
  <nml:Lifetime>{0,1}</nml:Lifetime>
  <nml:Location id="">{0,1}</nml:Location>
  <nml:Relation type="">{0,unbounded}</nml:Relation>
  <sd:serviceDefinition id=""/>
</nml:SwitchingService>
```

NSI Topology uses the following attributes/elements from the *nml:SwitchingService* element:

encoding

(optional) The data encoding identifier of the Service Domain if one is associated with the service. In NSI we use this attribute to identify similarly typed STP for matching for wildcard population of *nml:Port/nml:PortGroup* elements within a default *nml:SwitchingService*. Encoding is not required if all *nml:SwitchingService* elements are populated with their corresponding *Port/PortGroup* elements. For Ethernet framed services the following encoding URL will be used:

```
encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
```

id
: Assigned a globally unique *URN* identifier for the *Service Domain*.

labelSwapping
: A value of **false** adds a restriction to the nml:SwitchingService in that it is only able to create cross connects from an inbound Port to an outbound Port if the Label of the connected Ports have the same value. If the *labelSwapping* attribute is set to **true** then any inbound Port within the *nml:SwitchingService* can be connected to any other outbound Port independent of Label value. The default value is **false**.

  If *labelSwapping* is **true** then a single Service Domain is generated with all member Ports mapped to member STP; however, if **false** then a Service Domain is generated per Port Label value with each STP sharing the same Port Label value as a member of the same Service Domain.

labelType
- Assigned the type of Port Label associated with member ports that has values switched by the *nml:SwitchingService*. Will also be used to match on wildcard population of a default *nml:SwitchingService*. A *nml:Port* or *nml:PortGroup* may have at most one *labelType* attribute. The *labelType* attribute will use the following URL if supporting IEEE 802.1Q Ethernet:

```
labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan"
```

version
: (optional) Use is not currently defined in NSI topology.

nml:name
: (optional) A user visible string for the Service Domain.

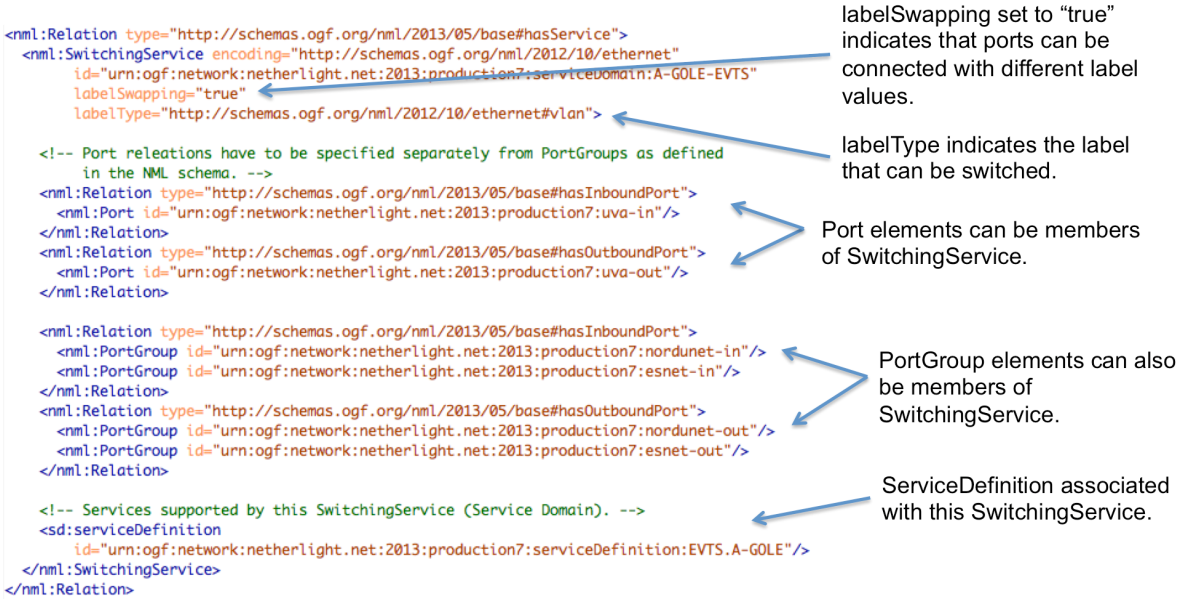
nml:Lifetime
: (optional) Use is not currently defined in NSI topology.

nml:Location
: (optional) May be used to model a geographical location for the Service Domain.

nml:Relation
: (optional) A set of one or more *hasInboundPort* and *hasOutboundPort* relations to *nml:Port* or *nml:PortGroup* that are members of the nml:SwitchingService. If there are no nml:Relations defined as part of the nml:SwitchingService then wildcard matching rules will be applied to populate the service with member *nml:Port* and/or *nml:PortGroup.*

sd:serviceDefinition
: At a minimum the reference identifier to the Service Definition associated with this Service Domain stored in either this NSI Topology document, or externally in the NSI-DDS. It is possible to store the entire Service Definition in this element if required by the deployment.

The following example shows a *nml:SwitchingService* element with the *labelSwapping* attribute set to **true**, resulting in a single Service Domain containing all the STP generated from the member *nml:Port* and *nml:PortGroup* elements. The generated Service Domain would support the "urn:ogf:network:netherlight.net:2013:production7:servicedefinition:EVTS.A-GOLE" Service Definition, which when defined in the next section supports the "`http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE`" Service Type.




```
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasService">
  <nml:SwitchingService encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
        id="urn:ogf:network:netherlight.net:2013:production7:serviceDomain:A-GOLE-EVTS"
        labelSwapping="true"
        labelType="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">

    <!-- Port releations have to be specified separately from PortGroups as defined
         in the NML schema. -->
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
      <nml:Port id="urn:ogf:network:netherlight.net:2013:production7:uva-in"/>
    </nml:Relation>
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
      <nml:Port id="urn:ogf:network:netherlight.net:2013:production7:uva-out"/>
    </nml:Relation>

    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-in"/>
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:esnet-in"/>
    </nml:Relation>
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-out"/>
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:esnet-out"/>
    </nml:Relation>

    <!-- Services supported by this SwitchingService (Service Domain). -->
    <sd:serviceDefinition
        id="urn:ogf:network:netherlight.net:2013:production7:serviceDefinition:EVTS.A-GOLE"/>
  </nml:SwitchingService>
</nml:Relation>
```

In the following example we have defined two port pairs with overlapping VLAN Label ranges “1779-1799” and “2-4095” but our *nml:SwitchingService* has a *labelSwapping* attribute set to **false**. This means ports with different Labels cannot be interconnected, and therefore, forces generation of a Service Domain corresponding to each VLAN label value defined in the *nml:PortGroup* set. In this case, we would get 4093 Service Domains defined with “urn:ogf:network:netherlight.net:2013:production7:nordunet” ports a member of all the Service Domains, but “urn:ogf:network:netherlight.net:2013:production7:uva” a member of only the Service Domains relating to VLAN labels “1779-1799”.

```
<!-- BidirectionalPort definitions for our two example ports. -->
<nml:BidirectionalPort id="urn:ogf:network:netherlight.net:2013:production7:uva">
  <nml:name>Asd001A_8700_07 4/1 UvA (SNE)</nml:name>
  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:uva-out"/>
  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:uva-in"/>
</nml:BidirectionalPort>

<nml:BidirectionalPort id="urn:ogf:network:netherlight.net:2013:production7:nordunet">
  <nml:name>Asd001A_8700_07 7/2 NORDUnet (nl-sar2-nordunet xe-0/0/3)</nml:name>
  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-out"/>
  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-in"/>
</nml:BidirectionalPort>

<!-- Basic service definition element for the EVTS.A-GOLE service. -->
<sd:serviceDefinition
      id="urn:ogf:network:netherlight.net:2013:production7:serviceDefinition:EVTS.A-GOLE">
  <name>GLIF Automated GOLE Ethernet VLAN Transfer Service</name>
  <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
</sd:serviceDefinition>

<!-- Unidirectional inbound port definitions. -->
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:uva-in">
    <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
      1779-1799
```

```
    </nml:LabelGroup>
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
      <nml:PortGroup id="urn:ogf:network:uvalight.net:2013:topology:netherlight-out"/>
    </nml:Relation>
  </nml:PortGroup>

  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-in">
    <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
      2-4095
    </nml:LabelGroup>
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
      <nml:PortGroup id="urn:ogf:network:nordu.net:2013:topology:netherlight-out"/>
    </nml:Relation>
  </nml:PortGroup>
</nml:Relation>

<!-- Unidirectional outbound port definitions. -->
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:uva-out">
    <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
      1779-1799
    </nml:LabelGroup>
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
      <nml:PortGroup id="urn:ogf:network:uvalight.net:2013:topology:netherlight-in"/>
    </nml:Relation>
  </nml:PortGroup>

  <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-out">
    <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
      2-4095
    </nml:LabelGroup>
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
      <nml:PortGroup id="urn:ogf:network:nordu.net:2013:topology:netherlight-in"/>
    </nml:Relation>
  </nml:PortGroup>
</nml:Relation>

<!-- SwitchingService definition with labelSwapping == false. -->
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasService">
  <nml:SwitchingService encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
        id="urn:ogf:network:netherlight.net:2013:production7:servicedomain:A-GOLE-EVTS"
        labelSwapping="false" labelType="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:uva-in"/>
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-in"/>
    </nml:Relation>
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:uva-out"/>
      <nml:PortGroup id="urn:ogf:network:netherlight.net:2013:production7:nordunet-out"/>
    </nml:Relation>
    <sd:serviceDefinition
        id="urn:ogf:network:netherlight.net:2013:production7:serviceDefinition:EVTS.A-GOLE"/>
  </nml:SwitchingService>
</nml:Relation>
```

6.9.1 Populating a default SwitchingService

To help with dynamic NSI Topology creation a few simple rules were added to help facilitate the generation of correct NML elements when they were either intentionally omitted or not fully populated (wildcard rules). The first example of these rules are around automatic creation of

default *nml:SwitchingService* element when one has not been defined in an NML Topology document.

If no *nml:SwitchingService* is specified within the *nml:Topology* element then this implies:

- A single *nml:SwitchingService* element is generated for each supported *labeltype/encoding* pair defined in the *nml:Port* and *nml:PortGroup* elements within the topology.
- Each *nml:SwitchingService* element has labelType set to the labeltype and encoding set to the encoding of the *nml:Port* and *nml:PortGroup* elements from which it was generated.
- Each *nml:SwitchingService* element has *labelSwapping* attribute set to false.
- Each *nml:SwitchingService* element contains all unidirectional ports of that *labeltype* and *encoding* from which it was generated defined as members within *hasInboundPort* and *hasOutboundPort* relations.
- Each *nml:SwitchingService* element contains all defined *ServiceDefinitions* supporting that *labelType* and *encoding*.

Ports defined with no labels are matched on *encoding* type only and placed in a *nml:SwitchingService* defined with no *labelType* or *labelSwapping* attributes.

Ports defined with no *encoding* are matched on *labelType* only (if available) and placed in a *nml:SwitchingService* defined with no *encoding* attribute, however, *labelType* and *labelSwapping* attributes can be present if used as a matching criteria.

6.9.2 Specifying a wildcard SwitchingService

When a specific default behavior is required, a *nml:SwitchingService* can be specified in the *nml:Topology* element with wildcard behaviors. In this case, all ports in the NML Topology document matching the wildcard specification of a *nml:SwitchingService* will be included included in that *nml:SwitchingService*. When a wildcard *nml:SwitchingService* is defined the default *nml:SwitchingService* behavior is no longer used.

A wildcard *nml:SwitchingService* is specified within the *nml:Topology* element similar to a normal *nml:SwitchingService* specification except no *nml:Relation* elements are included (*hasInboundPort* or *hasOutboundPort*). The lack of *nml:Relation* elements implies the *nml:SwitchingService* includes any *nml:Port or nml:PortGroups* that match the specified *labelType* and *encoding* of that *nml:SwitchingService*.

For example, the following wildcard *nml:SwitchingService* is defined that includes all ports using a "vlan" *labletype* and an "ethernet" *encoding*. This *nml:SwitchingService* is defined with *labelSwapping* set to **true** and with the "EVTS.A-GOLE" *ServiceDefinition*:

```
<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasService">
  <nml:SwitchingService encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
        id="urn:ogf:network:netherlight.net:2013:production7:serviceDomain:A-GOLE-EVTS"
        labelSwapping="true"
        labelType="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
    <sd:serviceDefinition
          id="urn:ogf:network:netherlight.net:2013:production7:serviceDefinition:EVTS.A-GOLE"/>
  </nml:SwitchingService>
</nml:Relation>
```

If there are ports defined within the *nml:Topology* element that do not match a defined *nml:SwitchingService* then these ports are not connectable.

6.10 Service Definition

A “Service Definition” describes, in very formal and declarative terms, the service being offered by a service provider to the service consumers. NSI [NSI-SD] conceived of service definitions to provide a formal mechanism to declare and bound the range of service characteristics associated with a Connection Service. This allows providers to guaranty the service capabilities they express in a Service Definition, and allows users to discern exactly what service capabilities they can request – with the expectation that these capabilities will be guaranteed. Within this document an XML schema was defined modeling the elements of a Service Definition, and the Service Type mechanism within the NSI CS protocol was proposed to decouple service parameterization from the core CS protocol itself. The contents of this document were later summarized and included in [GDF-212], however, the detailed XML schema documentation and example Service Definition templates can now only be found in the NSI schema repository [NSI-XSD].

The following is a simple instance representation of the *sd:serviceDefinition* element:

```
<sd:serviceDefinition id="" xmlns:tns="http://schemas.ogf.org/nsi/2013/12/services/definition">
  <name>{0,1}</name>
  <comment>{0,1}</comment>
  <serviceType>{0,1}</serviceType>
  <adaptation name="" symmetrical="" type="">{0,1}</adaptation>
  <schema name="" namespace="" required="" type="">{0,unbounded}</schema>
  <attribute name="" namespace="" type="" units="">{0,unbounded}</attribute>
  <error id="" text="">{0,unbounded}</error>
</sd:serviceDefinition>
```

Much of the *sd:serviceDefinition* document is for use by the NSI CS protocol engine and not used by NSI Topology. This will be explained below through example.

The following attributes/elements are defined as part of the *sd:serviceDefinition* document:

id
: A Service Definition must have a unique identifier within the network sourcing the definition.

sd:comment
: A comment describes the basic service offered in this Service Definition. Any differences from the global Service Definition identified by *sd:serviceType* should be identified here.

sd:serviceType
: This is the standard service type as defined in the associated global Service Definition. A standardize globally unique namespace identifier is allocated for each template.

sd:adaptation
: If this element is present then the Service Definition is providing service adaptation functionality. A service adaptation allows services from two incompatible Service Domains to be interconnected through a special adaptation function. This element will identify the type of adaptation, and the Service Types compatible with the adaptation.

sd:schema
: This is the schema elements specified in a reservation. There can be multiple schema entries here for services if they requiring multiple schema in a reserve request.

sd:parameter
: Parameter definitions for the service and their values. These reflect the XML schema definitions and any local range restrictions.

sd:attribute
: Attribute definitions for the service and their values. Attributes are aspects of the service that are not specified in the XML schema for the service.

sd:error
: Service specific errors defined for this service.

At a minimum the NSI Topology must include a *sd:serviceDefinition* element within the referencing *nml:SwitchingService* element that can be used to lookup the appropriate Service Definition document externally (such as from the NSI-DDS). This Service Definition document contains the *sd:serviceType* element holding the globally unique namespace identifier for the service. The *sd:serviceType* value is what is compared for equality between networks to determine if a service is compatible.

In the Automated GOLE deployment instead of requiring an external lookup of the Service Definition document, the NSI Topology document for a network contains a simple *sd:serviceDefinition* element defining the supported *sd:serviceType* offered by the Automated GOLE network. This unique identifier of this included *sd:serviceDefinition* is then referenced by *nml:SwitchingService* elements from within the topology document. The full Service Definition document for the Automated GOLE EVTS service is stored offline for implementation reference.

The following example shows a simplified *sd:serviceDefinition* element referenced from within a wildcard *nml:SwitchingService* element:

```
<sd:serviceDefinition
        id="urn:ogf:network:netherlight.net:2013:production7:serviceDefinition:EVTS.A-GOLE">
  <name>GLIF Automated GOLE Ethernet VLAN Transfer Service</name>
  <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
</sd:serviceDefinition>

<nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasService">
  <nml:SwitchingService encoding="http://schemas.ogf.org/nml/2012/10/ethernet"
        id="urn:ogf:network:netherlight.net:2013:production7:serviceDomain:A-GOLE-EVTS"
        labelSwapping="true"
        labelType="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
    <sd:serviceDefinition
          id="urn:ogf:network:netherlight.net:2013:production7:serviceDefinition:EVTS.A-GOLE"/>
  </nml:SwitchingService>
</nml:Relation>
```

Which mechanism is used to define and communicate the Service Definition document is a implementation and deployment specific decision.

[Should I go into more details here on the individual elements of the Service Definition?]

6.11 Adaptation (Discussion on the topic)

Within NML the function of adaptation describes the ability to embed data from one or more Ports into the data encoding of one other Port. In ITU terms this is embedding client layer ports into a

server layer port. The *nml:AdaptationService* element describes a unidirectional multiplexing adaptation function, while the *nml:DeadaptationService* describes the inverse unidirectional de-multiplexing function. This can be easily described using IEEE 802.1AD Ethernet as an example.

IEEE 802.1AD allows multiple IEEE 802.1Q VLAN tagged flows to be multiplexed (encapsulated) into a single IEEE 802.1AD flow by inserting a Service Provider tag (stag) into the Ethernet header, allowing the contents of the newly tagged frame to be switched through the network with no need to view the multiplexed VLAN identifier until it is unwrapped at the destination. Figure 9 below shows how two unidirectional ports would be modeled using a basic 802.1Q service within NSI Topology.

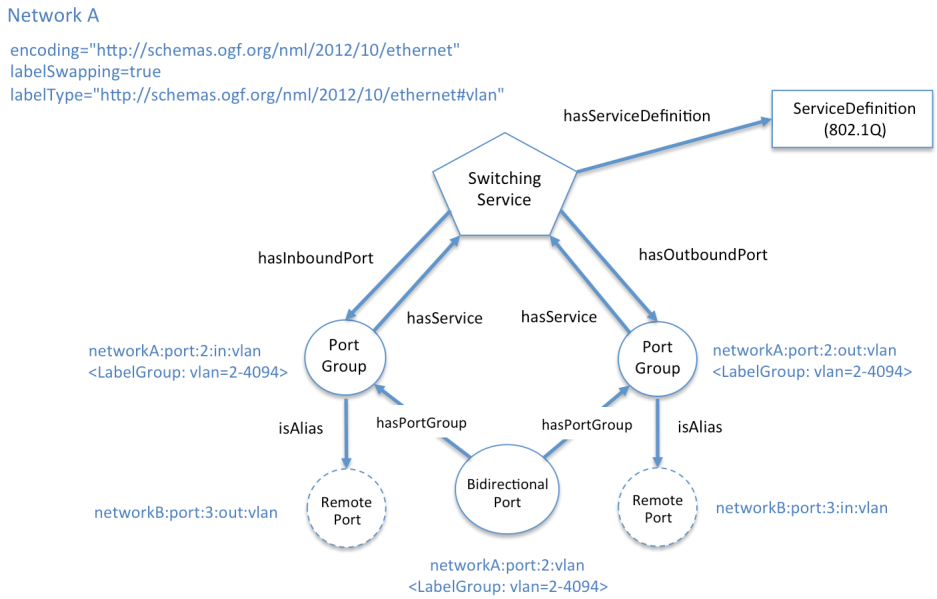


Figure 9 – Example NML for 802.1Q service.

Using standard STP naming scheme we can encode the VLAN identifier in URN query parameters to define the following STP identifiers for this NSI Service Domain:

Inbound:	`urn:ogf:network:example.com:networkA:port:2:in:vlan?vlan=2-4094`
Outbound:	`urn:ogf:network:example.com:networkA:port:2:out:vlan?vlan=2-4094`
Bidirectional:	`urn:ogf:network:example.com:networkA:port:2:vlan?vlan=2-4094`

In Figure 10 below an 802.1AD service is modeled similar to the modeling of the 802.1Q service in Figure 9, but instead of switching on the VLAN label, this service switches on stag label. Using the standard STP naming scheme the following STP identifiers would be creadted for this NSI Service Domain:

Inbound:	`urn:ogf:network:example.com:networkA:port:2:in:stag?stag=2-4094`
Outbound:	`urn:ogf:network:example.com:networkA:port:2:out:stag?stag=2-4094`
Bidirectional:	`urn:ogf:network:example.com:networkA:port:2:stag?stag=2-4094`

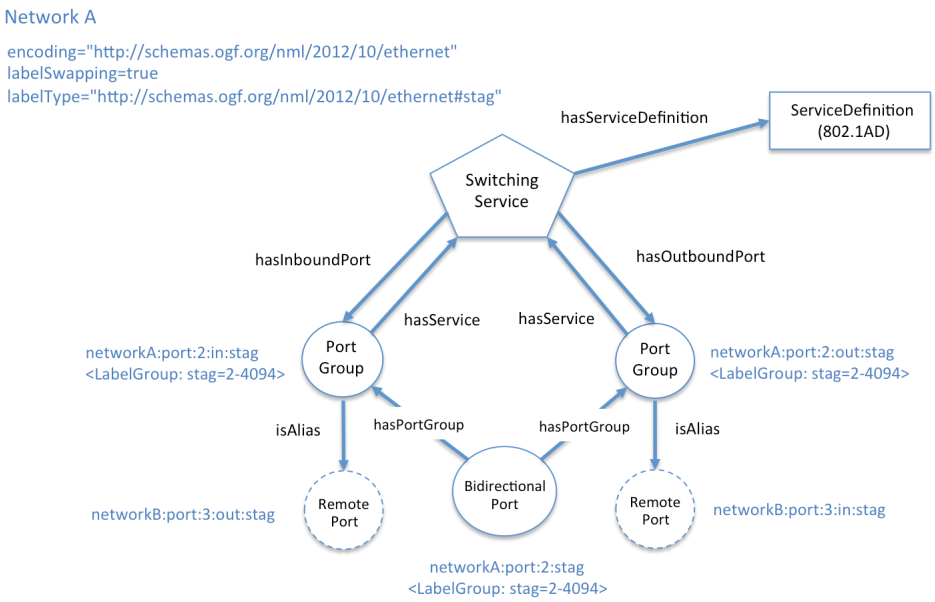


Figure 10 - Example NML for 802.1AD service.

The 802.1Q and 802.1AD examples can be combined to include port adaptation from 802.1Q into an 802.1AD port encoding by inserting an *nml:AdaptationService* on the outbound *nml:PortGroup* element using a *canProvidePort* relation, and an *nml:DeadaptationService* element on the inbound *nml:PortGroup* element also using the *canProvidePort* relation. The *canProvidePort* relation describes the potential, or ability of the adaptation to configure this port into existence. The 802.1AD adaptation is identified by setting the *adaptationFunction* attribute on each service to "`http://schemas.ogf.org/nml/2012/10/ethernet#802.1ad`".

The underlying 802.1AD *nml:PortGroup* elements are linked to their associated adaptations though the *hasService* relation. The 802.1AD ports now contain the *isAlias* relationships to the remote connected port to model an SDP. This modified example is shown below in Figure 11.

Fundamentally the service being offered by the 802.1AD NSI Service Domain does not change, as STP naming and switching capabilities remain the same. The only difference is the addition of encapsulation capabilities that had not been defined in the previous version.

The 802.1Q switched VLAN service however does change, requiring the ability to handle reservation requests with underlying adaptation into the 802.1AD ports. Implementing an 802.1Q service request that utilizes this adaptation could be implemented one of two ways:

1. The Service Definition is modified by the adaptation now requiring two pairs of STP in a reservation using a port in the adaptation. For example, if the port modeled above in Figure 11 was used in a bidirectional point to point service request a requester would specify the source (or destination) STP pair:

```
sourceSTP = {
        urn:ogf:network:example.com:networkA:port:2:vlan?vlan=1790,
        urn:ogf:network:example.com:networkA:port:2:stag?stag=1200
}
```

2. The Service Definition is modified by adaptation to allow for two query parameters when using a port in the adaptation.

```
sourceSTP = {
        urn:ogf:network:example.com:networkA:port:2:vlan?stag=1200&vlan=1790
}
```

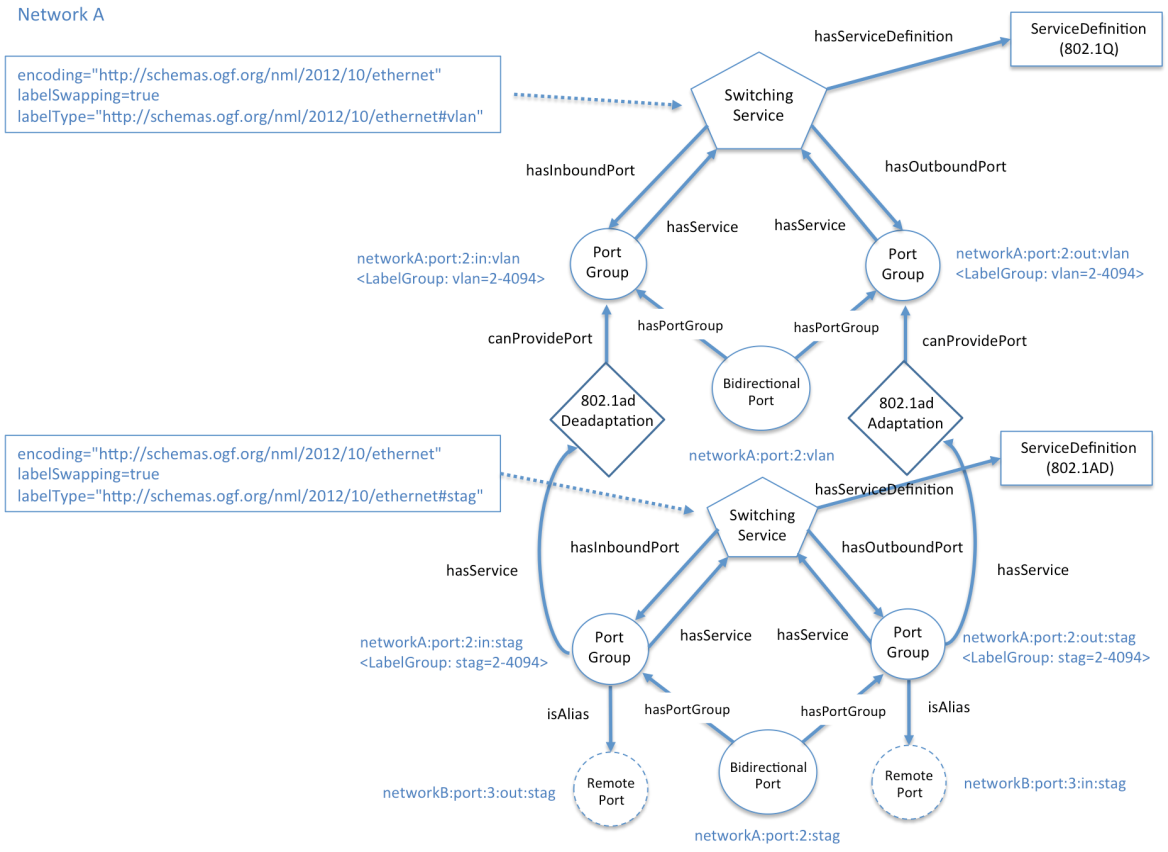
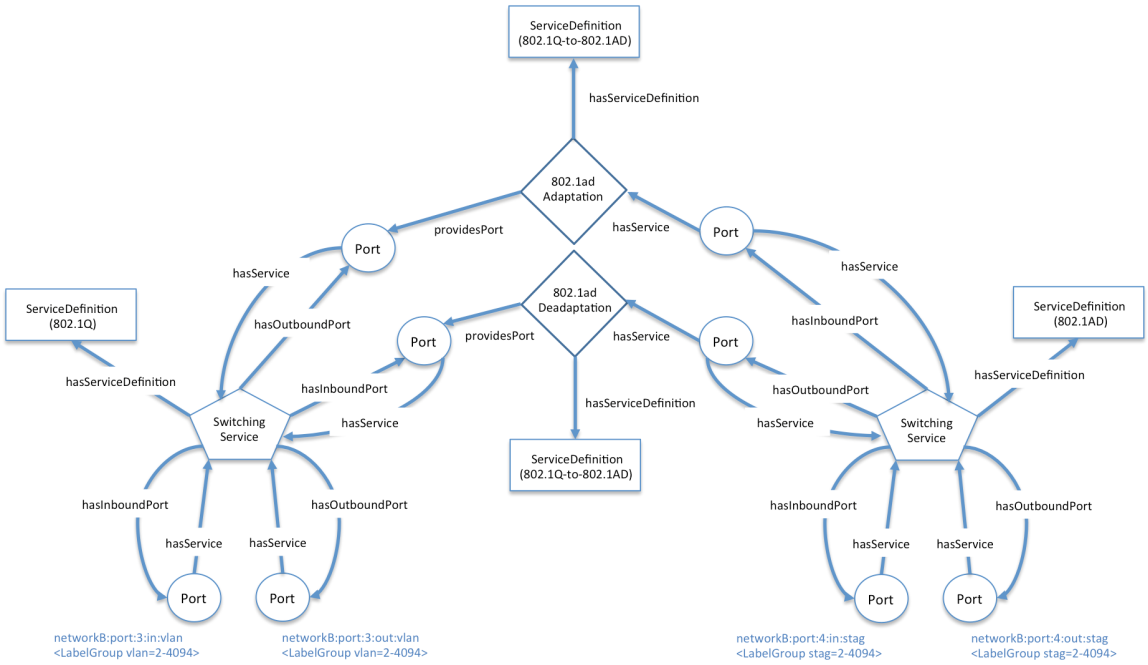


Figure 11 - Example NML for 802.1Q to 802.1AD adaptation.

It should be noted that this adaptation did not adapt between the *nml:SwitchingService* elements that represent the NSI Service Domains, but instead adapted on the *nml:Port/nml:PortGroup* elements. This port adaptation model does more closely follow how the service encapsulation would be done within the hardware itself. It would allow pathfinders to shortcut a path through the 802.1AD *nml:SwitchingService* if VLAN switching was not required, avoid de-encapsulation and re-encapsulation through the 802.1Q *nml:SwitchingService.*



6.12 Security Considerations

There are important security concerns associated with the generation and distribution of network topology information. For example, ISPs frequently consider network topologies to be proprietary. We do not address these concerns in this document, but implementers are encouraged to consider the security implications of generating and distributing network topology information. Implementers should be aware that the NML descriptions do not provide any guarantee regarding their integrity nor their authenticity. The NML documents also cannot provide this for the identifiers contained in the documents. Implementers should use external means of verifying the authenticity of identifiers contained in the documents.

7 Glossary

Adaptation	By definition, Service Domains of different Service Types cannot be directly connected due to the differing Service Definitions, however, an Adaptation can be defined that permits interconnection of STP from two different Service Domains using the concepts of encapsulation and adaptation. An Adaptation defines the (de)encapsulation or (de)adaptation of one service type into another service type if the Network is capable of offering the service.
Aggregator NSA (AG)	The Aggregator NSA is a Provider Agent that acts as both a requester and provider NSA. It can service requests from

	other NSA, perform path finding, and distribute segment requests to child NSA for processing.
Connection Service (CS)	The NSI Connection Service is a service that allows an RA to request and manage a Connection from a PA. See [OGF NSI-CS].
Explicit Routing Object (ERO)	An Explicit Routing Object (ERO) is a parameter in a Connection request. It is an ordered list of STP constraints to be used by the inter-Network pathfinder.
Intra-Network Topology	Refers to the topology of resources within a network, and the services offered by that network.
Inter-Network Topology	Refers to the topology of interconnected Networks and the common services offered across these interconnected Networks. Inter-Network Topology is concerned with describing the way in which Networks are statically interconnected by treating each Network as an aggregated set of Network capabilities and Edge Points.
Network	A group of network resources managed by a single network provider and a single NSA. A network exposes a set of defined service types representing the services offered to a user by the network.
Network Service Agent (NSA)	The Network Service Agent is a concrete piece of software that sends and receives NSI Messages. The NSA includes a set of capabilities that allow Network Services to be delivered.
Network Service Interface (NSI)	The NSI is the interface between RAs and PAs. The NSI defines a set of interactions or transactions between these NSAs to realize a Network Service.
Requester/Provider Agent (RA/PA)	An NSA acts in one of two possible roles relative to a particular instance of an NSI. When an NSA requests a service, it is called a Requester Agent (RA). When an NSA realizes a service, it is called a Provider Agent (PA). A particular NSA may act in different roles at different interfaces.
Service	A service is a “connection” between two points in a Network with certain predefined and dynamically specified characteristics that will deliver a payload from Network ingress to Network egress unmodified.
Service Definition	A document that describes the predefined characteristics and requestable elements associated with a service being offered by a Network.
Service Demarcation Point (SDP)	SDP are formed when a pair STPs of matching capabilities are considered adjacent (and connectable) between two Service Domains.
Service Domain (SD)	A group of STP within a Network described by a single Service Type and that can be fully interconnected without restriction.

Service Termination Point (STP)	An STP names a topological location that is the ingress/egress point of a Network and is defined by a single Service Type. An STP can be fully specified representing a single termination point, or under specified representing a set or bundle of STP.
Service Type	A predefined type of service offered by a network and specified by a Service Definition.
Service Region	The set of interconnected Service Domains of the same Service Type. (i.e. Inter-Network Topology for that service type).
XML Schema Definition (XSD)	XSD is a schema language for XML. See [W3C XSD]
eXtensible Markup Language (XML)	XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

8 Contributors

Jeroen J. van der Ham, UvA
Email: vdham@uva.nl

John H. MacAuley, ESnet
Email: macauley@es.net

9 Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights, which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

10 Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

11 Full Copyright Notice

12 References

[GFD.212] Roberts, G., OGF GFD-I.212, “Network Service Interface Connection Service, v2.0”, May 13, 2014.

[GFD.213] Roberts, G., OGF GFD-I.213, “Network Services Framework v2.0”, October 20, 2014.

[GFD.217] MacAuley, J., OGF GFD-I.217, “NSI Signaling and Path Finding”, April 24, 2014.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

[RFC2141bis] Sait-Andre, P., "Uniform Resource Name (URN) Syntax", draft-ietf-urnbis-rfc2141bis-urn-07, January 2014 (Expired).

[RFC5890] Klensin, j., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC6453] Dijkstra, F., "A URN Namespace for the Open Grid Forum (OGF)", RFC 6453, December 2011.

[GFD.202] van der Ham, J., GDF.202, "A URN Namespace for Network Resources", April 2013.

[GFD.220] MacAuley, J., GFD-R-P.220, “Network Service Agent Description”, July 1 2016.

[NSI-DDS] MacAuley, J., GWD-R, “Network Service Interface Document Distribution Service”, December 16, 2016.

[ISO8601] Technical Committee ISO/TC 154, ISO 8601:2004, “Data elements and interchange formats -- Information interchange -- Representation of dates and times”, 2004.

[NSI-SD] MacAuley, J., GWD-Draft, “NSI-CS Service Decoupling”, July 30 2013.

[NSI-XSD] MacAuley, J., “Open Grid Forum - Network Service Interface - Protocol Schema“, https://github.com/OpenGridForum/ogf-nsi-project.