# Service Termination Points (STPs) in NSI

**Scope**

This document summarises the usage of STPs in NSI. Only the usage of STPs in the Connection Service is considered here, usage of STPs in the Topology Service and in the Monitoring Service is out of scope. The details of how STPs map to NML objects is described in the NSI Topology Service.

## Agreed decisions relating to STPs

**NSI framework**

* The following set of services are supported:
	+ Topology Service (TS)
	+ Connection Service (CS)
	+ Monitoring Service (MS)
	+ Discovery Service (DS)
* The TS, CS and MS each have their own state-machine/s each of which is associated with an NSA. This is shown diagrammatically here:

NSI Framework

Conn.

Service

Topology

Service

Discovery

Service

Monitoring

Service

NSA

NSA

NRM

TSDB

**Definitions, usage and description of STPs**

* The NSI Topology will be referenced in several places:
	+ STP concepts are to be introduced in the NSI Framework
	+ Use of STP for CS described are to be described in the CS protocol document
	+ The Topology Service will describe how STPs map to NML and will define how to exchange topology

This document considers only the usage of STPS in the Connection Service

**Networks**

* A Network is a group of STPs that belong to a common topology description.
* A 1:1 relation exists between an NSA instance and an associated Network.
* It is legitimate for a Network to advertise a set of STPs some of which come from underlying providers. Eg Nordunet advertises an STP as part of Nordunet network that has been originally defined as a SUnet STP and retains a SUnet in its URN.

**STPs**

* STPs (Service Termination Points) are identifiers and are globally unique and persistent
* The concepts around STPs and Networks remain as currently described in the NS Framework.
* STPs point to an NML port described using the NML port syntax
* STPs are used in Connection requests to identify the endpoints of a Connections, and optionally as intermediate points on the path of the Connection to be used as path finding constraints.

**SDPs**

* An SDP (Service Demarcation Point) is defined as a grouping of two peering STPs on the edge of different Networks.

**Connection Requests in NSI v1.1**

In v1.1 a connection request includes a <path> object:

<path>

<directionality>

<originSTP>

<destSTP>

<stpList>

</path>

Where <stpList> is optional ordered list of type <STP>. Connections are either bidirectional or unidirectional.

The NSI v1.1 syntax for an STPId is: urn:ogf:network:stp:<networkId>:<localId>

Issues with this approach:

1. Not compliant with Freek’s new GFD URN naming.
2. Handling of unidirectional STPs is clumsy for proposes of expansion to multipoint service requests.
3. The routing of a Connection could be ambiguous as end STPs do not have an inherent direction. (ingress/egress interpretation problem)
4. The Network associated with an STP has to be found by parsing the URN.
5. Where an STP is re-advertised by a federating NSA, a new STP has to be created with the NetworkId of the Federating Network. This could be overcome by assigning a separate <Network> object.
6. Vlans cannot be identified in an STP (requires redirection to NML port).

**Connection Requests in NSI v2.0**

In v2.0 a connection request includes a <path> object:

<path>

<originSTP> of type <STP>

<destSTP> of type <STP>

<EROlist> of type ordered list of <STP>

</path>

Where <EROlist> is an optional ordered list of routing constraints. Do we need to identify if an ERO is strict or loose?

**STP syntax**

* <STP> is constituted of <network>, <source> , <sink>
* A bidirectional STP is a grouping of <network> and 2 unidirectional parts: <source> and <sink>.
* A unidirectional STP is a grouping of <network> and one unidirectional parts: <source> or <sink>.

Bidirectional STP:

<STP>

 <network>

 <source>

 <sink>

</STP>

Unidirectional STP:

<STP>

 <network>

 <source> or <sink>

</STP>

**Network**

* <network> is a globally unique identifier that identifies the Network. Rather than forcing parsing of an STP to determine the Network, a separate Network object is defined to allow an intermediate NSA to forward the message to the target Network without needing to know about the STPs within that domain.
* The syntax of <network> is urn:ogf:network:<DNSname>:<date>:org, Where urn:ogf:network:<DNSname>:<date> conform to GFD.191:
	+ DNSname is registered domain name
	+ Date is year in case domain name is reused

**Source/Sink**

* <source>,<sink> are identifiers for unidirectional components of the STP
* In the case of a candidate STP the <source>, <sink> parts are optional. (see below for discussion on candidate STPs)
* The syntax of <source>,<sink> is type port

**Port**

A port is a group that contains <portGroup> and <vlan>, were <portGroup> is a unique identifier in the form of a URN in conformance with GFD.191. <vlan> is a list of optional technology specific attributes.

<port>

 <portGroup>

 <vlan> (or other technology specific attributes)

<port>

* <portGroup> syntax is: urn:ogf:network:<DNSname>:<date>:<opaque>, Where urn:ogf:network:<DNSname>:<date> conforms to GFD.191 and <opaque> is a string conforming to URN characters and must be unique to that Network.
* <vlan> is an option technology attribute, the only currently defined technology attribute is vlan which has a value 1 through 4094. In future other technology specific attributes may be added.

**Candidate/Instance**

* Connection *requests* are made up of ‘**candidate STPs’** where candidate STP support a list or range of vlans. <vlan>118-259</vlan> or <vlan>118, 342,4,259</vlan>
* Candidate STP could also include only the <network> part of STP, i.e. the <source>,<sink> parts are optional.
* A connection *confirmation* will return an ‘**STP instance’** with a single confirmed vlan instance for each port, for example <vlan>118</vlan>

## Appendix 1: Example STP usage

**Example v1.1 NSI Connection request**

 <path>

 <directionality>Bidirectional</directionality>

 <sourceSTP>

 <stpId>urn:ogf:network:stp:czechlight.ets:ps-80</stpId>

 </sourceSTP>

 <destSTP>

 <stpId>urn:ogf:network:stp:czechlight.ets:ams-80</stpId>

 </destSTP>

 </path>

**Example v2.0 NSI Connection request**

 *<connectionrequest>*

 *....*

<path>

 *<originSTP>*

*<STP>*

 *<network>urn:ogf:network:nordu.net:2012:org</network>*

 *<source>*

*<port>*

 *<port\_group*

*idRef="urn:ogf:network:sne.science.uva.nl:2012:lighthouse-egress" />*

*<vlan>1791</vlan>*

*</port>*

 *</source>*

 *<sink>*

*<port>*

 *<port\_group*

*idRef="urn:ogf:network:sne.science.uva.nl:2012:lighthouse-egress" />*

*<vlan>1791</vlan>*

*</port>*

 *</sink>*

*</STP>*

 *</originSTP>*

 *<destSTP>*

*<STP>*

 *<network>urn:ogf:network:nordu.net:2012:org</network>*

 *<source>*

*<port>*

 *<port\_group*

*idRef="urn:ogf:network:sne.science.uva.nl:2012:lighthouse-egress" />*

*</port>*

 *</source>*

 *<sink>*

*<port>*

 *<port\_group*

*idRef="urn:ogf:network:sne.science.uva.nl:2012:lighthouse-egress" />*

*</port>*

 *</sink>*

*</STP>*

 *</destSTP>*

 *</destSTP>*

<EROlist>

*<STP>*

 *<network>urn:ogf:network:nordu.net:2012:org</network>*

 *<source>*

*<port>*

 *<port\_group*

*idRef="urn:ogf:network:sne.science.uva.nl:2012:lighthouse-egress" />*

*<vlan>1791</vlan>*

*</port>*

 *</source>*

 *<sink>*

*<port>*

 *<port\_group*

*idRef="urn:ogf:network:sne.science.uva.nl:2012:lighthouse-egress" />*

*<vlan>1791</vlan>*

*</port>*

 *</sink>*

*</STP>*

*<STP>*

 *<network>urn:ogf:network:nordu.net:2012:org</network>*

 *<source>*

*<port>*

 *<port\_group*

*idRef="urn:ogf:network:sne.science.uva.nl:2012:lighthouse-egress" />*

*</port>*

 *</source>*

 *<sink>*

*<port>*

 *<port\_group*

*idRef="urn:ogf:network:sne.science.uva.nl:2012:lighthouse-egress" />*

*</port>*

 *</sink>*

*</STP>*

</EROlist>