GWD-R-P
NML-WG
nml-wg@ogf.org

Jeroen van der Ham, UvA (editor)
Freek Dijkstra, SARA
Roman Łapacz, PSNC
Jason Zurawski, Internet2

September 2012

Network Markup Language Base Schema version 1

Status of This Document

Group Working Draft (GWD), candidate Recommendations Proposed (R-P).

Copyright Notice

Abstract

This document describes a normative schema which allows the description of a computer network topology.

Contents

1 Introduction

This document describes the base schema of the Network Markup Language (NML). Section 2.1 defines the NML classes and their attributes and parameters. Section 2.2 describes the relations defined between NML classes.

A NML network description can be expressed in XML, RDF/XML and Turtle syntax. Section 4 describes the RNC and XSD schema for the XML syntax. Section 5 describes the OWL 2 schema for the XML/RDF and Turtle syntaxes.

These basic classes defined in this document may be extended, or sub-classed, to represent technology specific classes.

Section 6 provides example use cases. This section is informative. Only sections 2 thru 5 are normative and considered part of the recommendation.

1.1 Scope

The Network Markup Language is designed to create a functional description of multi-layer networks (including virtualised networks) and multi-domain networks (including aggregated or abstracted networks). It can not only describe a statics network topology, but also its capabilities and its configuration.

NML is aimed at logical connection-oriented network topologies. It can also be used to describe physical networks or packet-oriented networks, although the current base schema current version does not contain classes or properties to explicitly deal with signal degradation, or complex routing tables.

NML only attempts to describe the data plane of a computer network, not the control plane. It does contain extension mechanism to easily tie it with network provisioning standards and with network monitoring standards.

Finally, you will not find a definition for the terms *Network* or *capacity* in this document. This has been a conscious choice. The term *Network* has become so widely used for so many diverse meanings that it is impossible to create a definition that everyone can agree on, while still expressing something useful. See *Topology* for the concept of a network domain and a *Link* with multiple sources and sinks for the concept of a local area network. The term *capacity* is used by different technologies in such a different way (e.g. including or excluding the packet overhead) that it is better to let technology-specific extensions make an explicit definition.

1.2 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].

This schema defines classes, attributes, relations, parameters and logic. Objects are instances of classes, and the type of an object is a class.

Names of classes are capitalised and written in italics (e.g. the *Node* class). Names of relations are written in camel case and in italics (e.g. the *hasNode* relations).

2 NML Base Schema

The NML Base schema describes an information model for computer networks. This schema is kept intentionally general, with provisions to extend the schema to describe layer-specific information.

The schema consists of classes, attributes, relations, parameters and logic. Classes describe types of objects and are described in section 2.1. Relations describe the relations between classes and are described in section 2.2. Attributes describe properties of classes. Logic describes how some relations may be derived from other relations. Parameters, like attributes, are properties of classes, but may (subtly) change the logic. Logic is described in section 2.3. Attributes and parameters are described with their class description.

All classes, relations, attributes and parameters defined in this document have an identifier within the namespace `http://schemas.ogf.org/nml/2012/10/base#`.

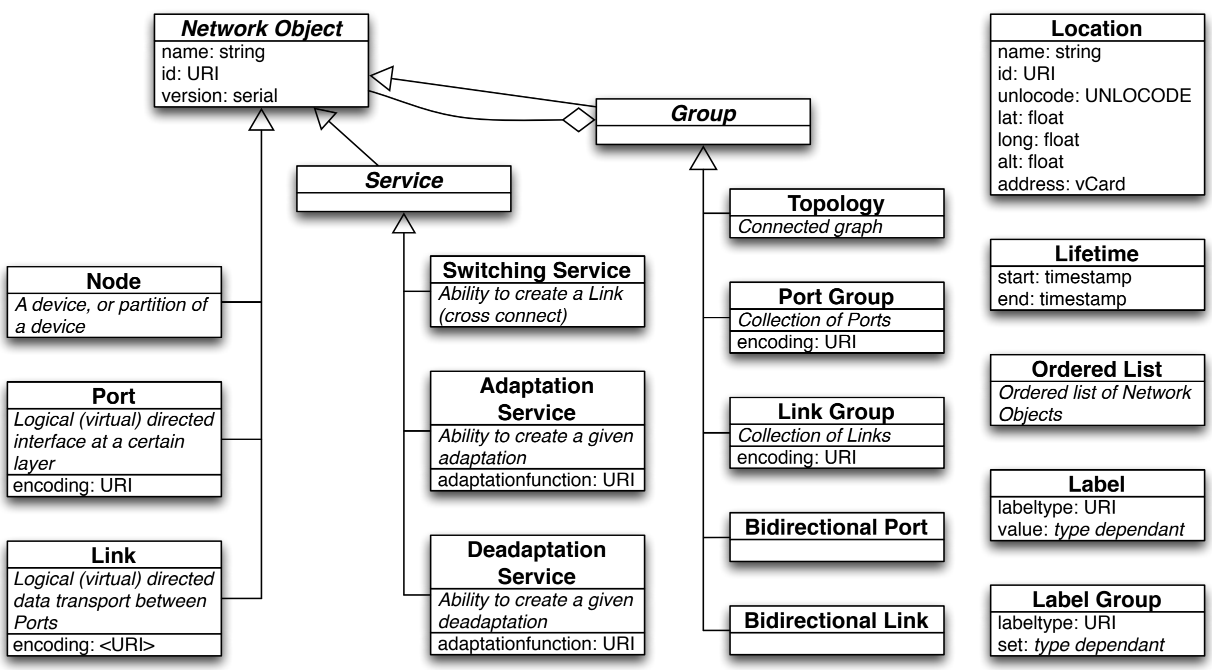

Figure 1: A UML class diagram of the classes in the NML schema and their hierarchy

2.1 Classes

Figure 1 shows an overview of all the classes in the NML schema in a UML class diagram. In the sections below we discuss each of the elements of the schema.

2.1.1 Network Object

The basic abstract class of the schema is the *Network Object*. Most classes inherit from it.

Network Object is an abstract class. It MUST NOT be instantiated directly.

A *Network Object* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *isAlias* to one or more *Network Object*s
- *locatedAt* to one *Location*

A *Network Object* may have the following attributes:

- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string
- *version* to assign a time stamp

A *Network Object* may have the following parameter:

- *isReference* A value of `false` means that the description of the *Network Object* is authoritative and complete. A value of `true` means that the *Network Object* is defined elsewhere and is only augmented in the current topology description. The default value is `false`.

The meaning of the *isAlias* relation is only defined for specific cases (between objects of the same class), and MUST NOT be used between other objects.

The meaning of the *version* attribute is only defined for specific cases (in objects of the Topology class), and SHOULD NOT be used in other objects. Clients that receive a *version* attribute for a non-*Topology* object MAY ignore that attribute.

An *id* is a persistent, globally unique object identifier for the *Network Object*. Section 3 describes these identifiers in detail.

name is a human readable string. A name may be written in any language, but it is RECOMMENDED that names are chosen so that all users can easily distinguish between different names. Two objects MAY have the same name. It is RECOMMENDED to use short, descriptive names. A name MUST NOT be used for anything other than display purposes. Normal Unicode recommendations apply: A name MUST NOT contain control or formatting codepoint (anything in the Other categories), and it is RECOMMENDED to only use codepoints from the Basic Multilingual Plane (BMP).

version is a time stamp in ISO 8601 format. The time stamp can be used to publish updates of a *Topology*. If a client receives multiple *Topology* descriptions, each with a different version time stamp, the version with the latest time stamp in the past or present MUST be considered the valid description. *Topology* descriptions with a time stamp in the future MAY be discarded or cached until the denoted time. See also the *Lifetime* object to describe historic or future network changes.

The base *Network Object* is subclassed into the top-level topology components, that are sufficient to cover the description of networks. The classes in this schema that directly inherit from *Network Object* are:

- Node
- Port
- Link
- Service
- Group

These classes are described in more detail below.

2.1.2 Node

A *Node* is generally a device connected to, or part of, the network. A Node does not necessarily correspond to a physical machine. It MAY be a virtual device or a group of devices.

Node inherits from *Network Object*.

A *Node* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasInboundPort* to one or more *Port*s or *PortGroup*s
- *hasOutboundPort* to one or more *Port*s or *PortGroup*s
- *hasService* to one or more *Service*s of type *Switch*
- *implementedBy* to one or more *Node*s

- *isAlias* to one or more *Node*s
- *locatedAt* to one *Location*

A *Node* may have the following attributes:

- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string

A *Node* may have the following parameter:

- *isReference* to describe if the *Node* is defined elsewhere.

2.1.3 Port

A *Port* defines connectivity from a *Network Object* to the rest of the network. A *Port* object is unidirectional. A *Port* does not necessarily correspond to a physical interface. It represents a logical transport entity at a fixed place in the network.

Port inherits from *Network Object*.

A *Port* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasLabel* to one *Label*
- *hasService* to one or more *Service*s of type *Adaptation* or type *Deadaptation*
- *isAlias* to one or more *Port*s
- *isSink* to one or more *Link*s
- *isSource* to one or more *Link*s

A *Port* may have the following attributes:

- *encoding* to assign a data encoding identifier
- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string

A *Port* may have the following parameter:

- *isReference* to describe if the *Port* is defined elsewhere.

The *encoding* attribute defines the format of the data stream through the Port. The identifier

2.1.4 Link

A *Link* object describes a unidirectional data transport from each of its sources to all of its sinks.

A source of a Link is a Network Object that has a *isSource* relation to the Link. A sink of a Link is a Network Object that has a *isSink* relation to the Link.

Link inherits from *Network Object*.

A *Link* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasLabel* to one *Label*
- *isAlias* to one or more *Link*s
- *isSerialCompoundLink* to one ordered *List* of *Link*s

A *Link* may have the following attributes:

- *encoding* to assign a data encoding identifier
- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string

A *Link* may have the following parameters:

- *isReference* to describe if the *Link* is defined elsewhere.
- *noReturnTraffic*. A value of `true` changes the definition of *Link* to: data transport from each sources to all sinks, except that there is no data transport from a source to a sink if the source and sink are grouped together in a *BidirectionalPort* group. The default value of *noReturnTraffic* is `false`.

2.1.5 Service

Service describes a capability of the network. That is, it describes how a configuration can be changed dynamically.

Service is an abstract class. It MUST NOT be instantiated directly.

Service inherits from *Network Object*. A *Service* may have the same relations, attributes and parameters as a *Network Object*.

This schema defines three different services, the *SwitchingService* the *AdaptationService* and the *DeadaptationService*. These are described in more detail below.

2.1.6 Switching Service

A *SwitchingService* describes the ability to create cross connects from its inbound *Port*s to its outbound *Port*s.

SwitchingService inherits from *Service*.

A *SwitchingService* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasInboundPort* to one or more *Port*s or *PortGroup*s
- *hasOutboundPort* to one or more *Port*s or *PortGroup*s
- *isAlias* to one or more *Switching Service*s
- *providesLink* to one or more *Link*s or *LinkGroup*s.

A *SwitchingService* may have the following attributes:

- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string

A *SwitchingService* may have the following parameters:

- *isReference* to describe if the *SwitchingService* is defined elsewhere.
- *labelSwapping*. A value of `false` adds a restriction to the *SwitchingService*: it is only able to create cross connects from an inbound *Port* to an outbound *Port* if the *Label* of the connected *Port*s has the same value. The default value is `false`.

The *providesLink* relation points to *Link*s which describe the currently configured cross connects in a *SwitchingService*.

2.1.7 Adaptation Service

An *AdaptationService* describes the capability that data from one or more *Port*s can be embedded in the data encoding of one other *Port*. This is commonly referred to as the embedding of client layer (higher network layer) ports in a server layer (lower network layer) port. The *AdaptationService* describes a multiplexing adaptation function, meaning that different channels (the client layer ports) can be embedded in a single data stream (the server layer port).

Like *Port* and *Link*, *AdaptationService* describes a unidirectional transport function. For the inverse transport function, see *DeadaptationService*.

An AdaptationServices describe

AdaptationService inherits from *Service*.

An *AdaptationService* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *isAlias* to one or more *AdaptationService*s
- *providesPort* to one or more *Port*s or *PortGroup*s

An *AdaptationService* may have the following attributes:

- *adaptationfunction* to assign an adaptation technology identifier
- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string

A *AdaptationService* may have the following parameter:

- *isReference* to describe if the *AdaptationService* is defined elsewhere.

2.1.8 De-adaptation Service

A *DeadaptationService* describes the capability that data of one or more ports can be extracted from the data encoding of one other port. This is commonly referred to as the extraction of client layer (higher network layer) ports from the server layer (lower network layer) port. The *DeadaptationService* describes a multiplexing adaptation function, meaning that different channels (the client layer ports) can be extracted from a single data stream (the server layer port).

Like *Port* and *Link*, *AdaptationService* describes a unidirectional transport function. For the inverse transport function, see *AdaptationService*[1].

DeadaptationService inherits from *Service*.

A *DeadaptationService* may have the following relations:

- *existsDuring* to one or more *Lifetime*s

[1]Whilst the *DeadaptationService* is an inverse of te *AdaptationService*, it should not be confused with an inverse multiplexing adaptation function. An inverse multiplexing adaptation function embeds a single data stream in multiple underlying data streams. To describes such a network, the experimental *parallelCompound* relation can be used, which is described in a seperate document [Dijkstra13].

- *isAlias* to one or more *DeadaptationService*s
- *providesPort* to one or more *Port*s or *PortGroup*s

A *DeadaptationService* may have the following attributes:

- *adaptationfunction* to assign a adaptation technology identifier
- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string

A *DeadaptationService* may have the following parameter:

- *isReference* to describe if the *DeadaptationService* is defined elsewhere.

2.1.9 Group

A *Group* describes a collections of network objects. Any object can be part of a group, including another *Group*.

Group is an abstract class. It MUST NOT be instantiated directly.

Group inherits from *Network Object*. A *Group* may have the same relations, attributes and parameters as a *Network Object*.

This schema defines five different *Group*s:

- Topology
- Port Group
- Link Group
- Bidirectional Port
- Bidirectional Link

These classes are described in more detail below.

2.1.10 Topology

A *Topology*[2] is a set of connected *Network Object*s. *connected* means that there is, or it is possible to create, a data transport between any two Network Objects in the same Topology, provided that there are no policy, availability or technical restrictions.

[2] At first this was called a Network, then Graph Network. The term Topology was suggested to avoid the confusion surrounding the term Network.

A *Topology* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasNode* to one or more *Node*s
- *hasInboundPort* to one or more *Port*s or *PortGroup*s
- *hasOutboundPort* to one or more *Port*s or *PortGroup*s
- *hasService* to one or more *Service* of type *Switch*
- *hasTopology* to one or more *Topology*s
- *isAlias* to one or more *Topology*s
- *locatedAt* to one *Location*

A *Topology* may have the following attributes:

- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string
- *version* to assign a serial number

A *Topology* may have the following parameter:

- *isReference* to describe if the *Topology* is defined elsewhere.

The *version* attribute is described at the *Network Object*.

2.1.11 Port Group

A *PortGroup* is an unordered set of *Port*s.

A *PortGroup* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasLabelGroup* to one *LabelGroup*
- *hasPort* to one or more *Port*s or *PortGroups*
- *isAlias* to one or more *PortGroup*s
- *isSink* to one or more *LinkGroup*s
- *isSource* to one or more *LinkGroup*s

A *PortGroup* may have the following attributes:

- *encoding* to assign a data encoding identifier
- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string

A *PortGroup* may have the following parameter:

- *isReference* to describe if the *PortGroup* is defined elsewhere.

2.1.12 Link Group

A *LinkGroup* is an unordered set of *Link*s.

A *LinkGroup* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasLabelGroup* to one *LabelGroup*
- *hasLink* to one or more *Link*s or *LinkGroup*s
- *isAlias* to one or more *LinkGroup*s
- *isSerialCompoundLink* to one ordered *List* of *LinkGroup*s

A *LinkGroup* may have the following attributes:

- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string

A *LinkGroup* may have the following parameter:

- *isReference* to describe if the *LinkGroup* is defined elsewhere.

2.1.13 Bidirectional Port

A *BidirectionalPort* is a group of two (unidirectional) *Port*s or *PortGroup*s together forming a bidirectional representation of a physical or virtual port.

A *BidirectionalPort* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasPort* to exactly two *Port*s or two *PortGroup*s

A *BidirectionalPort* may have the following attributes:

- *id* to assign a persistent globally unique URI

- *name* to assign a human readable string

A *BidirectionalPort* may have the following parameter:

- *isReference* to describe if the *BidirectionalPort* is defined elsewhere.

2.1.14 Bidirectional Link

A *BidirectionalLink* is a group of two (unidirectional) *Link*s or *LinkGroup*s together forming a bidirectional link.

A *BidirectionalLink* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasLink* to exactly two *Link*s or two *LinkGroup*s

A *BidirectionalLink* may have the following attributes:

- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string

A *BidirectionalLink* may have the following parameter:

- *isReference* to describe if the *BidirectionalLink* is defined elsewhere.

2.1.15 Location

A *Location* is a reference to a geographical location or area.

A *Location* may have the following attributes:

- *id* to assign a persistent globally unique URI
- *name* to assign a human readable string
- *long* is the longitude in WGS84 coordinate system (in decimal degrees)
- *lat* is the lattitude in WGS84 coordinate system (in decimal degrees)
- *alt* is the altitude in WGS84 coordinate system (in decimal meters)
- *unlocode* is the UN/LOCODE location identifier
- *address* is a vCard address

A *Location* may have the following parameter:

- *isReference* to describe if the *Location* is defined elsewhere.

2.1.16 Lifetime

A *Lifetime* is an interval between which the object is said to be active.

A *Lifetime* may have the following attributes:

- *start* is the start time and date in ISO datetime notation
- *end* is the end time and date in ISO datetime notation

2.1.17 Ordered List

An *OrderedList* is an ordered list of *Network Object*s.

The representation of an *OrderedList* depends on the syntax.

2.1.18 Label

A *Label* is the technology-specific value to distinguish a single data stream embedded in a larger data stream. The *value* can either be a resource label, or a pair of source and destination labels.

A *Label* may have the following attributes:

- *type* to refer to a technology-specific labelset
- *value* is one specific value taken from the labelset

Technology extensions of NML may define additional attributes.

2.1.19 Label Group

A *LabelGroup* is an unordered set of *Label*s.

A *LabelGroup* may have the following attributes:

- *type* to refer to a technology-specific labelset
- *values* is a set of specific values taken from the labelset

Technology extensions of NML may define additional attributes.

2.2 Relations

*Relation*s describe how different*Network Object*s can be combined to form a network topology description. The relations have been described above, but for ease of reference we also give a full list and definition here (in alphabetical order). In principle a *Relation* can go from any object to any other object. The list below includes definitions for a subset of the possible relations. If a particular *Relation* between two *Network Objects* is not listed below, it is undefined.

existsDuring relates a *LifeTime* object to a *Network Object*

hasInboundPort defines the relation between a *Node*, a *SwitchingService* or a *Topology* and their respective *Port*s or *PortGroup*s

hasLabelGroup assigns one *LabelGroup* to a *PortGroup*

hasLabel assigns one *Label* to a *Port*

hasLink is used for:

- *Bidirectional Link* to relate exactly two *Link*s or two *LinkGroup*s
- *LinkGroup* to one or more *Link*s or *LinkGroup*s to define membership of that group

hasNode relates a *Network Object* to a *Node*, meaning that a *Node* is part of a *Topology*

hasOutboundPort relates either a *Node*, *SwitchingService* or a *Topology* to one or more *Port*s or *PortGroup*s as an outbound port

hasPort is used for:

- *BidirectionalPort* to relate exactly two *Port*s or two *PortGroup*s
- *PortGroup* to one or more *Port*s or *PortGroups*

hasService relates a *Network Object* to a *Service*. This schema only defines the meaning of:

- *Port* to *AdaptationService*, relating one server-layer *Port* to an adaptation function
- *Port* to *DeadaptationService*, relating one server-layer *Port* to a deadaptation function
- *Node* or *Topology* to *SwitchingService*, describing a switching capability of that *Node* or *Topology*.

hasSink relates a *Link* to one *Port* to define the outgoing traffic port

hasSource relates a *Link* to one *Port* to define its incoming traffic port

hasTopology defines a relation between one *Topology* to one or more *Topology*s for aggregation purposes

implementedBy relates a *Node* to one or more *Node*s to describe virtualization

isAlias is a relation from a *Network Object* to a *Network Object* to describe that one can be used as the alias of another.

isSerialCompoundLink is used to define that a *Link* or*LinkGroup* represents an ordered *List* of *Link*s or*LinkGroup*s. This must include cross-connects.

locatedAt relates a *Network Object* to one *Location*

providesLink is used to relate a *SwitchingService* to one or more *Link*s or *LinkGroup*s to define that these have been created by that *SwitchingService*

providesPort is used to relate an *AdaptationService* or *DeadaptationService* to one or more *Port*s or *PortGroup*s to define that these have been created by that *AdaptationService* or *DeadaptationService*

The *hasTopology*, *hasNode*, *implementedBy* *hasPort*, *hasLabel*, *hasLabelGroup*, and *hasLink* are defined as implicit relations.

2.3 Logic

3 Identifiers

3.1 Object Identifiers

The namespace for the class objects defined in this document is `http://schemas.ogf.org/nml/base/2013/`
TODO: change to correct year and month of the schema.

All objects and attributes defined in this document reside in this namespace. For example, the link object is identified by `http://schemas.ogf.org/nml/2013/10/base/link`

3.2 Instance Identifiers

Section 2.1.1 requires that instances of Network Objects MUST have an *id* attribute, which MUST be a unique URI.

Implementations that receive a network topology description MUST be prepared to accept any valid URI as an identifier.

Implementations that publish a network topology description instance identifiers MAY adhere to the syntax of Global Network Identifiers as defined in [URN-OGF-NETWORK], which ensures global uniqueness and that easy recognition of Network Object instances.

Two different Network Objects instance MUST have two different identifiers.

Once an identifier is assigned to a resource, it MUST NOT be re-assigned to another resource.

A URI MAY be interpreted as an International Resource Identifier (IRI) for display purposes, but URIs from external source domains MUST NOT be IRI-normalised before transmitting to others.

3.2.1 Lexical Equivalence

Two identifier are lexical equivalent if they are binary equivalent after case folding.

No interpretation of percent-encoding or PUNYCODE decoding should take place.

For the purpose of equivalence comparison, any possible fragment part or query part of the URI is considered part of the URI.

For example the following identifiers are equivalent:

```
1 - urn:ogf:network:example.net:2012:local_string_1234
2 - URN:OGF:network:EXAMPLE.NET:2012:Local_String_1234
```

while the following identifiers are not equivalent (in this case, the percentage encoding even make URI #3 an invalid Global Network Identifier.):

```
1 - urn:ogf:network:example.net:2012:local_string_1234
3 - urn:ogf:network:example.net:2012:local%5Fstring%5F1234
```

3.2.2 Further Restrictions

An assigning organisation MUST NOT assign Network Object Identifier longer than 255 characters in length.

Parsers MUST be prepared to accept identifiers of up to 255 characters in length.

A Parser SHOULD verify if an identifier adheres to the general URI syntax rules, as specified in RFC 3986 [RFC 3986].

Parsers SHOULD reject identifiers which do not adhere to the specified rules. A parser encountering an invalid identifier SHOULD reply with an error code that includes the malformed identifier, but MAY accept the rest of the message, after purging all references to the Network Object with the malformed identifier.

3.2.3 Interpreting Identifiers

A Network Object identifier MUST be treated as a opaque string, only used to uniquely identify a Network Object. The local-part of a Global Network Identifier MAY have certain meaning to it's assigning organisation, but MUST NOT be interpreted by any other organisation.

3.2.4 Network Object Attribute Change

A Network Object may change during its lifetime. If these changes are so drastic that the assigning organisation considers it a completely new Network Object, the assigning organisation should be assigned a new identifier. In this case, other organisations MUST treat this object as completly new Network Resource.

If the assigning organisation considers the changes are small, it MUST retain the same identifier for the Network Object, and use some mechanism to signal it's peers of the changes in the attributes of the Network Object.

4 XML Schema

5 OWL Schema

6 Examples

The following snippets represent NML structures in the XML format.

- *Topology*

```
<nml:Topology xmlns:nml="http://schemas.ogf.org/nml/2012/10/nml"
              id="urn:ogf:network:example.net:2012:org"
              version="20120814">

  <!-- ... -->

</nml:Topology>
```

- *Node*

```
<nml:Node id="urn:ogf:network:example.net:2012:nodeA">
  <nml:name>Node_A</nml:name>
  <nml:Location idRef="urn:ogf:network:example.net:2012:redcity"/>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasOutboundPort">
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_X:out"/>
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_Y:out"/>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasInboundPort">
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_X:in"/>
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_Y:in"/>
  </nml:Relation>
</nml:Node>
```

- *Ports*

  - *UnidirectionalPort*

```
<nml:Port id="urn:ogf:network:example.net:2012:port_X:out">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
</nml:Port>
```

  - *BidirectionalPort*

```
<nml:BidirectionalPort id="urn:ogf:network:example.net:2012:port_X">
  <nml:name>X</nml:name>
  <nml:Port idRef="urn:ogf:network:example.net:2012:port_X:out"/>
  <nml:Port idRef="urn:ogf:network:example.net:2012:port_X:in"/>
</nml:BidirectionalPort>
```

  - *PortGroup*

```
<nml:PortGroup id="urn:ogf:network:example.net:2012:portgroup_X:out">
  <nml:LabelGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1780-1783
  </nml:LabelGroup>
</nml:PortGroup>
```

- *Link*

– *UnidirectionalLink*

```
<nml:Link id="urn:ogf:network:example.net:2012:linkA:XY"/>

<nml:Port id="urn:ogf:network:example.net:2012:port_X:out">
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSource">
    <nml:Link idRef="␣urn:ogf:network:example.net:2012:linkA:XY"/>
  </nml:Relation>
</nml:Port>

<nml:Port id="urn:ogf:network:example.net:2012:port_Y:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSink">
    <nml:Link idRef="␣urn:ogf:network:example.net:2012:linkA:XY"/>
  </nml:Relation>
</nml:Port>
```

– *UnidirectionalLink that is composed of more than one sub-link*

```
<nml:Relation type="http://schemas.ogf.org/nml/2012/10/isSerialCompoundLink">
  <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:XY">
    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/next">
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:YZ"/>
    </nml:Relation>
  </nml:Link>
  <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:YZ">
    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/next">
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:ZW"/>
    </nml:Relation>
  </nml:lLink>
  <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:ZW"/>
</nml:Relation
```

– *BidirectionalLink*

```
<nml:BidirectionalLink id="urn:ogf:network:example.net:2012:link_XWX">
  <nml:name>Link between ports X and W</nml:name>
  <nml:Link idRef="urn:ogf:network:example.net:2012:link_XW"/>
  <nml:Link idRef="urn:ogf:network:example.net:2012:link_WX"/>
</nml:BidirectionalLink>
```

– *LinkGroup*

```
<nml:LinkGroup id="urn:ogf:network:example.net:2012:domainy_domainx">
  <nml:LabelGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1780−1783
  </nml:LabelGroup>
</nml:LinkGroup>
```

- *Labels*

– *Label*

```
<nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
```

– *LabelGroup*

```
<nml:LabelGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
  1780−1783
</nml:LabelGroup>
```

- *Location*

```
<nml:Location id="urn:ogf:network:example.net:2012:redcity">
  <nml:name>Red City</nml:name>
  <nml:latitude>30.600</nml:latitude>
  <nml:longitude>12.640</nml:longitude>
</nml:Location>
```

- *Services*
  - *SwitchingService*

```
<nml:Node id="urn:ogf:network:example.net:2012:nodeA">
  <nml:name>Node_A</nml:name>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasInboundPort">
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_X:in" />
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_Y:in" />
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasOutboundPort">
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_X:out"/>
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_Y:out"/>
  </nml:Relation>
  <nml:SwitchingService idRef="urn:ogf:network:example.net:2012:nodeA:switchingService"/>
</nml:Node>

<nml:SwitchingService id="urn:ogf:network:example.net:2012:nodeA:switchingService">
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasInboundPort">
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_X:in" />
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_Y:in" />
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasOutboundPort">
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_X:out"/>
    <nml:Port idRef="urn:ogf:network:example.net:2012:nodeA:port_Y:out"/>
  </nml:Relation>
</nml:SwitchingService>
```

  - *AdaptationService*

```
<nml:Port id="urn:ogf:network:example.net:2012:port_X:in">
  <nml:AdaptationService
       idRef="urn:ogf:network:example.net:2012:port_X:in:adaptationService"/>
</nml:Port>

<nml:AdaptationService
     id="urn:ogf:network:example.net:2012:port_X:in:adaptationService">
  <nml:Port idRef="urn:ogf:network:example.net:2012:port_X.1501:in"/>
</nml:AdaptationService>

<nml:Port id="urn:ogf:network:example.net:2012:port_X.1501:in">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
</nml:Port>
```

  - *DeadaptationService*

```
<nml:Port id="urn:ogf:network:example.net:2012:port_X.1501:in">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
  <nml:DeadaptationService
       idRef="urn:ogf:network:example.net:2012:port_X.1501:in:deadaptationService" />
</nml:Port>

<nml:DeadaptationService
     id="urn:ogf:network:example.net:2012:port_X.1501:in:deadaptationService">
  <nml:Port idRef="urn:ogf:network:example.net:2012:port_X:in" />
</nml:DeadaptationService >
```

7 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.

8 Glossary

9 Contributors

Jeroen J. van der Ham (Editor)
Faculty of Science, Informatics Institute, University of Amsterdam
Science Park 904, 1098 XH Amsterdam
The Netherlands
Email: vdham@uva.nl

Freek Dijkstra
SARA
Science Park 140, 1098 XG Amsterdam
The Netherlands
Email: freek.dijkstra@sara.nl

Roman Łapacz
PSNC
ul. Noskowskiego 12/14, 61-704 Poznań
Poland
Email: romradz@man.poznan.pl

Jason Zurawski
Internet2
1150 18th Street, NW
Suite 1020
Washington, DC 20036
USA
Email: zurawski@internet2.edu

10 Acknowledgments

The authors like to thank the NML working group members for their patience.

11 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.

12 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.

13 Full Copyright Notice

Appendix A NML example - first use case

```xml
<?xml version="1.0" encoding="utf-8" ?>


<!-- -->
<!-- port X port Y port Z port W -->
<!-- O-------------------O------------------O-------------------O -->
<!-- link A link B link C -->
<!-- -->
<!-- (---------------------------------------------------------) -->
<!-- link XWX -->
<!-- -->


<nml:Topology xmlns:nml="http://schemas.ogf.org/nml/2012/10/nml"
              id="urn:ogf:network:gn3.net:2012:org"
              version="201207019">


  <nml:name>OGF Test Topology</nml:name>


  <!-- -------------- Links -------------- -->

  <nml:Link id="urn:ogf:network:example.net:2012:link_XW">
    <nml:Relation type="http://schemas.ogf.org/nml/2013/10/isSerialCompoundLink">
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:XY">
        <nml:Relation type="http://schemas.ogf.org/nml/2012/10/next">
          <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:YZ"/>
        </nml:Relation>
      </nml:Link>
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:YZ">
        <nml:Relation type="http://schemas.ogf.org/nml/2012/10/next">
          <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:ZW"/>
        </nml:Relation>
      </nml:lLink>
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:ZW"/>
        </nml:Relation>
  </nml:Link>


  <nml:Link id="urn:ogf:network:example.net:2012:link_WX">
    <nml:Relation type="http://schemas.ogf.org/nml/2013/10/isSerialCompoundLink">
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:WZ">
        <nml:Relation type="http://schemas.ogf.org/nml/2012/10/next">
          <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:ZY"/>
        </nml:Relation>
      </nml:Link>
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:ZY">
        <nml:Relation type="http://schemas.ogf.org/nml/2012/10/next">
          <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:YX"/>
        </nml:Relation>
      </nml:Link>
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:YX"/>
        </nml:Relation>
  </nml:Link>


  <nml:BidirectionalLink id="urn:ogf:network:example.net:2012:link_XWX">
    <nml:name>Bidirectional link between ports X and W</nml:name>
    <nml:Link idRef="urn:ogf:network:example.net:2012:link_XW"/>
    <nml:Link idRef="urn:ogf:network:example.net:2012:link_WX"/>
  </nml:BidirectionalLink>


  <nml:Link id="urn:ogf:network:example.net:2012:linkA:XY"/>
  <nml:Link id="urn:ogf:network:example.net:2012:linkA:YX"/>

  <nml:BidirectionalLink id="urn:ogf:network:example.net:2012:linkA">
    <nml:name>A</nml:name>
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:XY"/>
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:YX"/>
  </nml:BidirectionalLink>


  <nml:Link id="urn:ogf:network:example.net:2012:linkB:YZ"/>
  <nml:Link id="urn:ogf:network:example.net:2012:linkB:ZY">
```

```
<nml:BidirectionalLink id="urn:ogf:network:example.net:2012:linkB">
  <nml:name>B</nml:name>
  <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:YZ"/>
  <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:ZY"/>
</nml:BidirectionalLink>

<nml:Link id="urn:ogf:network:example.net:2012:linkC:ZW"/>
<nml:Link id="urn:ogf:network:example.net:2012:linkC:WZ"/>

<nml:BidirectionalLink id="urn:ogf:network:example.net:2012:linkC">
  <nml:name>C</nml:name>
  <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:ZW"/>
  <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:WZ"/>
</nml:BidirectionalLink>

<!-- --------------- Ports --------------- -->

<nml:Port id="urn:ogf:network:example.net:2012:port-X:out">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSource">
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:XY"/>
  </nml:Relation>
</Port>

<nml:Port id="urn:ogf:network:example.net:2012:port-X:in">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSink">
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:YX"/>
  </nml:Relation>
</Port>

<nml:BidirectionalPort id="urn:ogf:network:example.net:2012:port-X">
  <nml:name>X</nml:name>
  <nml:Port idRef="urn:ogf:network:example.net:2012:port-X:out"/>
  <nml:Port idRef="urn:ogf:network:example.net:2012:port-X:in"/>
</nml:BidirectionalPort>

<nml:Port id="urn:ogf:network:example.net:2012:port-Y:out">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSource">
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:YX"/>
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:YZ"/>
  </nml:Relation>
</Port>

<nml:Port id="urn:ogf:network:example.net:2012:port-Y:in">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSink">
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:XY"/>
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:ZY"/>
  </nml:Relation>
</Port>

<nml:BidirectionalPort id="urn:ogf:network:example.net:2012:port-Y">
  <nml:name>Y</nml:name>
  <nml:Port idRef="urn:ogf:network:example.net:2012:port-Y:out"/>
  <nml:Port idRef="urn:ogf:network:example.net:2012:port-Y:in"/>
</nml:BidirectionalPort>

<nml:Port id="urn:ogf:network:example.net:2012:port-Z:out">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSource">
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:ZY"/>
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:ZW"/>
  </nml:Relation>
</Port>

<nml:Port id="urn:ogf:network:example.net:2012:port-Z:in">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSink">
```

```
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkB:YZ"/>
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:WZ"/>
    </nml:Relation>
  </Port>


  <nml:BidirectionalPort id="urn:ogf:network:example.net:2012:port-Z">
    <nml:name>Z</nml:name>
    <nml:Port idRef="urn:ogf:network:example.net:2012:port-Z:out"/>
    <nml:Port idRef="urn:ogf:network:example.net:2012:port-Z:in"/>
  </nml:BidirectionalPort>


  <nml:Port id="urn:ogf:network:example.net:2012:port-W:out">
    <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSource">
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:WZ"/>
    </nml:Relation>
  </Port>


  <nml:Port id="urn:ogf:network:example.net:2012:port-W:in">
    <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">1501</nml:Label>
    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/isSink">
      <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:ZW"/>
    </nml:Relation>
  </Port>


  <nml:BidirectionalPort id="urn:ogf:network:example.net:2012:port-W">
    <nml:name>W</nml:name>
    <nml:Port idRef="urn:ogf:network:example.net:2012:port-W:out"/>
    <nml:Port idRef="urn:ogf:network:example.net:2012:port-W:in"/>
  </nml:BidirectionalPort>


</nml:Topology>
```

Appendix B NML example - second use case

```
<?xml version="1.0" encoding="utf-8" ?>

<!-- Domain X Domain Y -->
<!-- --------------------- ------------------------------------------------------------- -->
<!-- | | | | -->
<!-- | Node A | | Node B Node C | -->
<!-- | Port ge-0/2/9 | | Port ge-1/0/9 Port ge-1/0/8 Port ge-5/2/7 | -->
<!-- | vlan: | | vlan: vlan: vlan: | -->
<!-- | 1501,1780-1783 | | 1501,1780-1783 1501 1501 | -->
<!-- | O---------------------------------------O------------------------------------O | -->
<!-- | | | | -->
<!-- | | | | -->
<!-- --------------------- ------------------------------------------------------------- -->


<nml:Topology xmlns:nml="http://schemas.ogf.org/nml/2012/10/nml"
             id="urn:ogf:network:gn3.net:2012:org"
             version="20120709">

  <nml:name>OGF Test Topology #1</nml:name>

  <!-- -------------------------------- Domain X -------------------------------- -->

  <nml:Topology id="urn:ogf:network:domainx.net:2012:org">


    <nml:name>Domain X</nml:name>


    <nml:Node id="urn:ogf:network:domainx.net:2012:nodeA">
      <nml:name>Node−A</nml:name>

      <nml:Location idRef="urn:ogf:network:domainx.net:2011:redcity"/>

      <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasOutboundPort">
        <nml:PortGroup idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9-out"/>
        <nml:Port idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9.1501-out"/>
      </nml:Relation>
      <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasInboundPort">
        <nml:PortGroup idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9-in"/>
        <nml:Port idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9.1501-in"/>
      </nml:Relation>

    </nml:Node>


    <nml:PortGroup id="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9-out">
      <nml:LabelGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
        1780−1783
      </nml:LabelGroup>
    </nml:PortGroup>


    <nml:PortGroup id="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9-in">
      <nml:LabelGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
        1780−1783
      </nml:LabelGroup>
    </nml:PortGroup>

    <nml:Port id="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9.1501-out">
      <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
        1501
      </nml:Label>
    </nml:Port>


    <nml:Port id="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9.1501-in">
      <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
        1501
      </nml:Label>
    </nml:Port>


    <nml:BidirectionalPort id="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9">
      <nml:name>ge−0/2/9</nml:name>
      <nml:PortGroup idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9-out"/>
```

```xml
    <nml:PortGroup idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9-in"/>
  </nml:BidirectionalPort>


  <nml:BidirectionalPort id="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9.1501">
    <nml:name>ge-0/2/9 vlan 1501</nml:name>
    <nml:PortGroup idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9.1501-out"/>
    <nml:PortGroup idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9.1501-in"/>
  </nml:BidirectionalPort>


  <nml:LinkGroup id="urn:ogf:network:domainx.net:2012:domainx-domainy">
    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasSource">
      <nml:PortGroup idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9-out"/>
      <nml:Port idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9.1501-out"/>
    </nml:Relation>
    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasSink">
      <nml:PortGroup idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9-in"/>
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:A:port_ge-1.0.9.1501-in"/>
    </nml:Relation>
  </nml:LinkGroup>


  <nml:LinkGroup id="urn:ogf:network:domainx.net:2012:domainy-domainx">
    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasSource">
      <nml:PortGroup idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9-out"/>
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501-out"/>
    </nml:Relation>
    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasSink">
      <nml:PortGroup idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9-in"/>
      <nml:Port idRef="urn:ogf:network:domainx.net:2012:A:port_ge-0.2.9.1501-in"/>
    </nml:Relation>
  </nml:LinkGroup>


  <nml:BidirectionalLink id="urn:ogf:network:domainx.net:2012:domainx-domainy-domainx">
    <nml:name>Link between domain x and domain y</nml:name>
    <nml:LinkGroup idRef="urn:ogf:network:domainx.net:2012:domainx-domainy"/>
    <nml:LinkGroup idRef="urn:ogf:network:domainx.net:2012:domainy-domainx"/>
  </nml:BidirectionalLink>


  <nml:Location id="urn:ogf:network:domainx.net:2011:redcity">
    <nml:name>Red City</nml:name>
    <nml:latitude>15.600</nml:latitude>
    <nml:longitude>32.640</nml:longitude>
  </nml:Location>


</nml:Topology>


<!-- -------------------------------- Domain Y -------------------------------- -->

<nml:Topology id="urn:ogf:network:domainy.net:2012:org">


  <nml:name>Domain Y</nml:name>


  <nml:Node id="urn:ogf:network:domainy.net:2012:nodeB">
    <nml:name>Node-B</nml:name>

    <nml:Location idRef="urn:ogf:network:domainy.net:2011:whitecity"/>

    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasOutboundPort">
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-out"/>
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501-out"/>
      <nml:PortGroup idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9-out"/>
    </nml:Relation>
    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasInboundPort">
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-in"/>
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501-in"/>
      <nml:PortGroup idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9-in"/>
    </nml:Relation>

    <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasService">
      <nml:SwitchingService idRef="urn:ogf:network:domainy.net:2012:B:switchingService_vlan1501"/>
    </nml:Relation>

  </nml:Node>
```

```
<nml:Node id="urn:ogf:network:domainy.net:2012:nodeC">
  <nml:name>Node-C</nml:name>
  <nml:Location idRef="urn:ogf:network:domainy.net:2011:whitecity"/>

  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasOutboundPort">
    <nml:Port idRef="urn:ogf:network:domainy.net:2012:C:port_ge-5.2.7.1501-out"/>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasInboundPort">
    <nml:Port idRef="urn:ogf:network:domainy.net:2012:C:port_ge-5.2.7.1501-in"/>
  </nml:Relation>

</nml:Node>


<nml:SwitchingService id="urn:ogf:network:domainy.net:2012:B:switchingService_vlan1501">
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasInboundPort">
    <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-in" />
    <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501-in" />
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasOutboundPort">
    <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-out" />
    <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501-out" />
  </nml:Relation>
</nml:SwitchingService>


<nml:PortGroup id="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9-out">
  <nml:LabelGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1780-1783
  </nml:LabelGroup>
</nml:PortGroup>


<nml:PortGroup id="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9-in">
  <nml:LabelGroup encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1780-1783
  </nml:LabelGroup>
</nml:PortGroup>


<nml:Port id="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501-out">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1501
  </nml:Label>
</nml:Port>


<nml:Port id="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501-in">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1501
  </nml:Label>
</nml:Port>


<nml:Port id="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-out">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1501
  </nml:Label>
</Port>


<nml:Port id="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-in">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1501
  </nml:Label>
</Port>


<nml:Port id="urn:ogf:network:domainy.net:2012:C:port_ge-5.2.7.1501-out">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1501
  </nml:Label>
</Port>


<nml:Port id="urn:ogf:network:domainy.net:2012:C:port_ge-5.2.7.1501-in">
  <nml:Label encoding="http://schemas.ogf.org/nml/2012/10/ethernet/vlan">
    1501
  </nml:Label>
```

```
    </Port>

    <nml:BidirectionalPort id="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9">
      <nml:name>ge−1/0/9</nml:name>
      <nml:PortGroup idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9-out"/>
      <nml:PortGroup idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9-in"/>
    </nml:BidirectionalPort>
    <nml:BidirectionalPort id="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501">
      <nml:name>ge−1/0/9 vlan 1501</nml:name>
      <nml:PortGroup idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501-out"/>
      <nml:PortGroup idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.9.1501-in"/>
    </nml:BidirectionalPort>
    <nml:BidirectionalPort id="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501">
      <nml:name>ge−1/0/8 vlan 1501</nml:name>
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-out"/>
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-in"/>
    </nml:BidirectionalPort>
    <nml:BidirectionalPort id="urn:ogf:network:domainy.net:2012:C:port_ge-5.2.7.1501">
      <nml:name>ge−5/2/7 vlan 1501</nml:name>
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:C:port_ge-5.2.7.1501-out"/>
      <nml:Port idRef="urn:ogf:network:domainy.net:2012:C:port_ge-5.2.7.1501-in"/>
    </nml:BidirectionalPort>

    <nml:BidirectionalLink id="urn:ogf:network:domainy.net:2012:domainx-domainy-domainx">
      <nml:Relation type"http://schemas.ogf.org/nml/2012/10/relation/isAlias">
        <nml:BidirectionalLink idRef="urn:ogf:network:domainx.net:2012:domainx-domainy-domainx"/>
      <nml:Relation>
    </nml:BidirectionalLink>

    <nml:Link id="urn:ogf:network:domainy.net:2012:B-to-C">
      <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasSource">
        <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-out"/>
      </nml:Relation>
      <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasSink">
        <nml:Port idRef="urn:ogf:network:domainy.net:2012:C:port_ge-5.2.7.1501-in"/>
      </nml:Relation>
    </nml:Link>

    <nml:Link id="urn:ogf:network:domainy.net:2012:C-to-B">
      <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasSource">
        <nml:Port idRef="urn:ogf:network:domainy.net:2012:C:port_ge-5.2.7.1501-out"/>
      </nml:Relation>
      <nml:Relation type="http://schemas.ogf.org/nml/2012/10/relation/hasSink">
        <nml:Port idRef="urn:ogf:network:domainy.net:2012:B:port_ge-1.0.8.1501-sink"/>
      </nml:Relation>
    </nml:Link>

    <nml:BidirectionalLink id="urn:ogf:network:domainy.net:2012:B-C-B">
      <nml:name>Link between boxes B and C</nml:name>
      <nml:Link idRef="urn:ogf:network:domainy.net:2012:B-to-C"/>
      <nml:Link idRef="urn:ogf:network:domainy.net:2012:C-to-B"/>
    </nml:Link>

    <nml:Location id="urn:ogf:network:domainy.net:2011:whitecity">
      <nml:name>White City</nml:name>
      <nml:latitude>30.600</nml:latitude>
      <nml:longitude>12.640</nml:longitude>
    </nml:Location>

  </nml:Topology>

</nml:Topology>
```

References

Normative References

[URN-OGF-NETWORK] Freek Dijkstra, and Jeroen van der Ham. A URN Namespace for Network Resources. GWD-I *draft-gwdi-urn-ogf-network* (Work in Progress), September 2012. URL `https://forge.ogf.org/sf/go/doc16260`.

[RFC 2119] Scott Bradner. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119 (Best Current Practice), March 1997. URL `http://tools.ietf.org/html/rfc2119`.

[RFC 3986] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform Resource Identifier (URI): Generic Syntax RFC 3986 (Standards Track), January 2005. URL `http://tools.ietf.org/html/rfc3986`.

Informative References

[Dijkstra13] Freek Dijkstra, et al. Experimental Features for NML 1. Work in Progress.