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 set of normative schemas which allow the description of computer network topologies.

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.

An NML network description can be expressed in XML[XML], and RDF/XML[RDF-XML] syntax. Section A describes the XSD schema for the XML syntax. Section B describes the OWL 2 schema for the RDF/XML syntax.

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

Section 4 provides example use cases. This section is informative. Only sections 2, 3, and appendices A and B 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 and multi-domain networks. An example of a multi-layered network can be a virtualised network, but also using different technologies. The multi-domain network descriptions can include aggregated or abstracted network topologies. NML can not only describe a static 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 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* relation).

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, and parameters. 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. 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#`.

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.

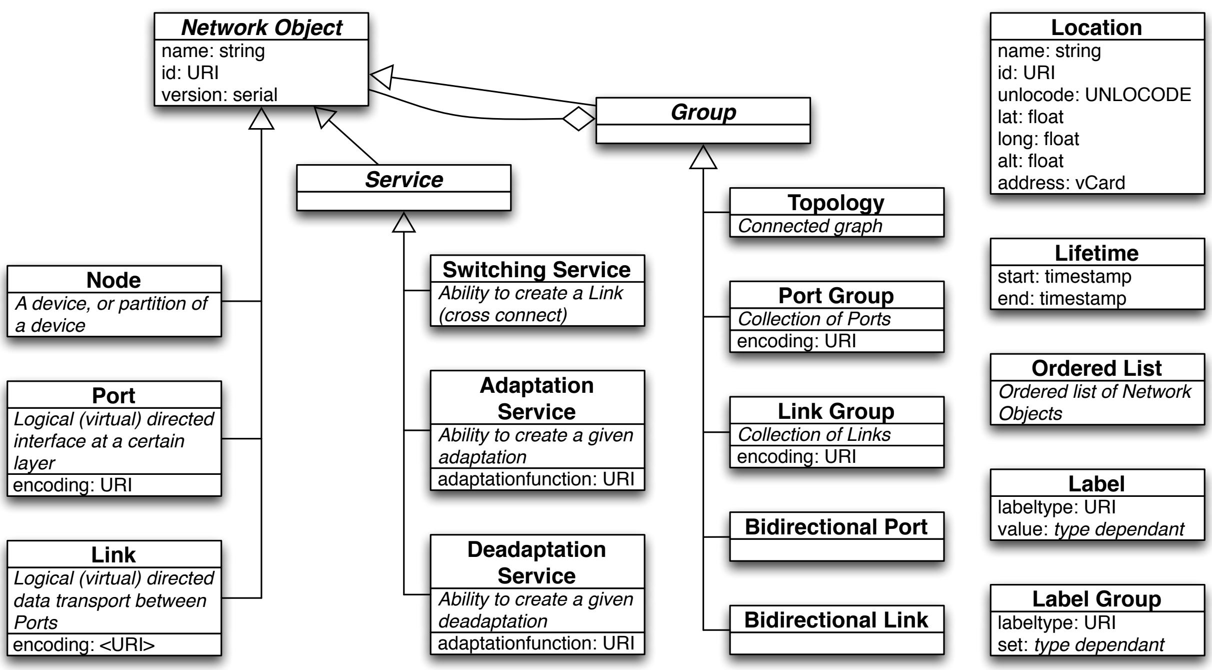

Figure 1: A UML class diagram of the classes in the NML schema and their hierarchy

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

The meaning of the *isAlias* relation is only defined for specific cases (between objects of the same concrete 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 SHOULD ignore that attribute.

An *id* is a persistent, globally unique object identifier for the *Network Object*. The *id* SHOULD be used to refer to this 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 formatted as ISO 8601 calendar date, and MUST be a basic (compact) representation with UTC timezone (`YYYYMMDDThhmmssZ`) [ISO 8601]. 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 (e.g. when used in aggregation).

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

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

The *encoding* attribute defines the format of the data streaming through the Port. The identifier for the encoding MUST be a URI. Encoding URIs SHOULD be specified in a Grid Forum Documents (GFD).

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.

A *Link* object can have a *isSerialCompoundLink* relation to a *List* of *Links*. This describes that the *Link* represents a path through the network implemented by that (ordered) *List* of *Links*.

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 parameter:

- *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 the behavior 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 new *Link*s from any of its inbound *Port*s to any of 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 parameter:

- *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). For example multiplexing several VLANs over a single trunk port.

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

AdaptationService inherits from *Service*.

An *AdaptationService* may have the following relations:

- *canProvidePort* to one or more *Port*s or *PortGroup*s
- *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

DeadaptationService is an inverse of *AdaptationService*. This 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 an experimental relation, described in a separate document [Dijkstra13].

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 demultiplexing adaptation function, meaning that different channels (the client layer ports) can be extracted from a single data stream (the server layer port). For example demultiplexing several VLANs from a single trunk port.

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

DeadaptationService inherits from *Service*.

A *DeadaptationService* may have the following relations:

- *canProvidePort* to one or more *Port*s or *PortGroup*s
- *existsDuring* to one or more *Lifetime*s
- *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

2.1.9 Group

A *Group* describes a collections of 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*[1] 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.

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

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

- *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

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

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

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

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

2.1.15 Location

A *Location* is a reference to a geographical location or area. A *Location* object can be related to other *Network Object*s to describe that these are located there. This can be relevant for network measurements, visualisations, et cetera.

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) [WGS84]
- *lat* is the latitude 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 [UNLOCODE]
- *address* is a vCard ADR (address) property. The exact syntax of the address property is not specified, to allow other (e.g. XML or RDF) representations of the string-based format specified in [RFC 6350].

2.1.16 Lifetime

A *Lifetime* is an interval between which the object is said to be active. This can be used to track changes in a network, reflect dynamic operations, to help debug problems, et cetera.

A *Lifetime* 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 (`YYYYMMDDThhmmssZ`) [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 (`YYYYMMDDThhmmssZ`)

Objects with multiple lifetimes mean that the lifetime of the object is the union of all lifetimes (as opposed to a intersection).

If a Network Object has no associated Lifetime objects, or the start or end attribute of a Lifetime object is missing, the default lifetime may be assumed to start on or before the time specified in the version attribute of the most specific Topology object that contains this Network Object, and the end on or later than the version attribute of the next published Topology object.

If a Network Object has no associated Lifetime objects, and the Topology object does not have a version attribute, than the lifetime of the Network Object is undefined.

2.1.17 Ordered List

An *OrderedList* is an ordered list of *Network Object*s. These are used for the *isSerialCompoundLink* relation to an ordered list of *Link*s to describe a path through the network.

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. Examples of this are a VLAN number, wavelength, et cetera.

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 relate to each other, typically to form a network topology description. The relations have been listed above, and are defined (in alphabetical order). In principle a *Relation* can go from any object to any other object.

The list below make a distinction between *allowed* and *defined* relations. An *allowed* relation means it is valid NML. A *defined* relation means that it has a specific meaning, as described here.

A relation which is *not allowed* SHOULD be rejected by a client. A relation which is *allowed*, but (yet) *undefined* SHOULD be silently ignored by a client. This distinction allows future extension of NML, while retaining limited backward compatibility.

The *existsDuring*, *hasLabel*, *hasLabelGroup*, *hasLink*, *hasNode*, *hasPort*, *hasService*, *hasTopology*, *locatedAt*, *providesLink*, and *providesPort* are defined as *implicit* relations. All other relations are *explicit*. The distinction between implicit and explicit relations may be used by some syntaxes to allow a more compact network description.

2.2.1 canProvidePort

canProvidePort is used to relate an *AdaptationService* or *DeadaptationService* to one or more *Port*s or *PortGroup*s to define that these can be created by that *AdaptationService* or *DeadaptationService*.

Allowed relations are:

- *Service* * — canProvidePort ▸ — * *Port*
- *Service* * — canProvidePort ▸ — * *PortGroup*

Defined relations are:

- *AdaptationService* * — canProvidePort ▸ — * *Port*
- *AdaptationService* * — canProvidePort ▸ — * *PortGroup*
- *DeadaptationService* * — canProvidePort ▸ — * *Port*
- *DeadaptationService* * — canProvidePort ▸ — * *PortGroup*

2.2.2 existsDuring

existsDuring relates one *Network Object* object to zero or more *LifeTime* objects. This defines the existence of the object at a certain time.

Network Object 1 — existsDuring ▸ — * *LifeTime*

Objects with multiple lifetimes mean that the lifetime of the object is the union of all lifetimes (as opposed to a intersection).

If a Network Object has no associated Lifetime objects, or the start or end attribute of a Lifetime object is missing, the default lifetime may be assumed to start on or before the time specified in the version attribute of the most specific Topology object that contains this Network Object, and the end on or later than the version attribute of the next published Topology object.

If a Network Object has no associated Lifetime objects, and the Topology object does not have a version attribute, than the lifetime of the Network Object is undefined.

2.2.3 hasInboundPort

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

Allowed relations are:

- *Network Object* * —hasInboundPort ▸— * *Port*
- *Network Object* * —hasInboundPort ▸— * *PortGroup*

Defined relations are:

- *Node* * —hasInboundPort ▸— * *Port*
- *Node* * —hasInboundPort ▸— * *PortGroup*
- *SwitchingService* * —hasInboundPort ▸— * *Port*
- *SwitchingService* * —hasInboundPort ▸— * *PortGroup*
- *Topology* * —hasInboundPort ▸— * *Port*
- *Topology* * —hasInboundPort ▸— * *PortGroup*

This defines that the related *Network Object* has an inbound *Port* or *PortGroup* object. The direction of the *Port* object is relative to the *Network Object* the*Port* is attached to, so in this case the traffic flows towards that *Network Object* (similarly for the *PortGroup*).

2.2.4 hasLabel

hasLabel assigns one *Label* to a *Port* or *Link*

Allowed relations are:

- *Port* 1 —hasLabel ▸— * *Label*
- *Link* 1 —hasLabel ▸— * *Label*

The *Label* assigned to this *Port* is the technology label that the traffic through this *Port* has been assigned. This *Label* is used to distinguish the *Port* in a *PortGroup*, or to define the traffic flow through a *SwitchingMatrix*.

The meaning of*hasLabel* is only defined for a cardinality of 0 or 1.

2.2.5 hasLabelGroup

hasLabelGroup assigns one *LabelGroup* to a *PortGroup* or *LinkGroup*

Allowed relations are:

- *PortGroup* 1 — hasLabelGroup ▸ — * *LabelGroup*
- *LinkGroup* 1 — hasLabelGroup ▸ — * *LabelGroup*

The *LabelGroup* assigned to this *PortGroup* defines the *Label*s associated with the *Port*s member of that group. The size of the *LabelGroup* MUST be equal to the size of the *PortGroup*.

The meaning of*hasLabelGroup* is only defined for a cardinality of 0 or 1.

2.2.6 hasLink

hasLink is used for:

- *BidirectionalLink* 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

Allowed relations are:

- *Group* * — hasLink ▸ — * *Link*
- *Group* * — hasLink ▸ — * *LinkGroup*

Defined relations are:

- *LinkGroup* * — hasLink ▸ — * *Link*
- *LinkGroup* * — hasLink ▸ — * *LinkGroup*
- *BidirectionalLink* * — hasLink ▸ — 2 *Link*
- *BidirectionalLink* * — hasLink ▸ — 2 *LinkGroup*

The *hasLink* relationships for a *BidirectionalLink* point to the two unidirectional *Link*s that together form a bidirectional connection between its respective associated *Node*s.

The *hasLink* relationships for a *LinkGroup* define the membership of the *Link*s in that *LinkGroup*.

2.2.7 hasNode

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

Allowed relations are:

- $\boxed{\textit{Network Object}}\ {}_{*}\xrightarrow{\text{hasNode}}{}_{*}\ \boxed{\textit{Node}}$

Defined relations are:

- $\boxed{\textit{Topology}}\ {}_{*}\xrightarrow{\text{hasNode}}{}_{*}\ \boxed{\textit{Node}}$

2.2.8 hasOutboundPort

hasOutboundPort relates either a *Node*, *SwitchingService* or a *Topology* to one or more *Port*s or *PortGroup*s.

Allowed relations are:

- $\boxed{\textit{Network Object}}\ {}_{*}\xrightarrow{\text{hasOutboundPort}}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{Network Object}}\ {}_{*}\xrightarrow{\text{hasOutboundPort}}{}_{*}\ \boxed{\textit{PortGroup}}$

Defined relations are:

- $\boxed{\textit{Node}}\ {}_{*}\xrightarrow{\text{hasOutboundPort}}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{Node}}\ {}_{*}\xrightarrow{\text{hasOutboundPort}}{}_{*}\ \boxed{\textit{PortGroup}}$
- $\boxed{\textit{SwitchingService}}\ {}_{*}\xrightarrow{\text{hasOutboundPort}}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{SwitchingService}}\ {}_{*}\xrightarrow{\text{hasOutboundPort}}{}_{*}\ \boxed{\textit{PortGroup}}$
- $\boxed{\textit{Topology}}\ {}_{*}\xrightarrow{\text{hasOutboundPort}}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{Topology}}\ {}_{*}\xrightarrow{\text{hasOutboundPort}}{}_{*}\ \boxed{\textit{PortGroup}}$

This defines that the related *Network Object* has an outbound *Port* or *PortGroup* object. The direction of the *Port* object is relative to the *Network Object* the*Port* is attached to, so in this case the traffic flows away from that *Network Object* (similarly for the *PortGroup*).

2.2.9 hasPort

hasPort is used for:

- *BidirectionalPort* to relate exactly two *Port*s or two *PortGroup*s

- *PortGroup* to one or more *Port*s or *PortGroup*s

Allowed relations are:

- $\boxed{\textit{Group}}\ {}_{*}\underline{\text{ hasPort } \blacktriangleright}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{Group}}\ {}_{*}\underline{\text{ hasPort } \blacktriangleright}{}_{*}\ \boxed{\textit{PortGroup}}$

Defined relations are:

- $\boxed{\textit{PortGroup}}\ {}_{*}\underline{\text{ hasPort } \blacktriangleright}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{PortGroup}}\ {}_{*}\underline{\text{ hasPort } \blacktriangleright}{}_{*}\ \boxed{\textit{PortGroup}}$
- $\boxed{\textit{BidirectionalPort}}\ {}_{*}\underline{\text{ hasPort } \blacktriangleright}{}_{2}\ \boxed{\textit{Port}}$
- $\boxed{\textit{BidirectionalPort}}\ {}_{*}\underline{\text{ hasPort } \blacktriangleright}{}_{2}\ \boxed{\textit{PortGroup}}$

The *hasPort* relationships for a *BidirectionalPort* point to the two unidirectional *Port*s that together form a bidirectional port for the associated *Node*.

The *hasPort* relationships for a *PortGroup* define the membership of the *Port*s in that *PortGroup*.

2.2.10 hasService

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*.

Allowed relations are:

- $\boxed{\textit{Network Object}}\ {}_{*}\underline{\text{ hasService } \blacktriangleright}{}_{*}\ \boxed{\textit{Service}}$

Defined relations are:

- $\boxed{\textit{Port}}\ {}_{1}\underline{\text{ hasService } \blacktriangleright}{}_{*}\ \boxed{\textit{AdaptationService}}$
- $\boxed{\textit{Port}}\ {}_{1}\underline{\text{ hasService } \blacktriangleright}{}_{*}\ \boxed{\textit{DeadaptationService}}$
- $\boxed{\textit{Node}}\ {}_{*}\underline{\text{ hasService } \blacktriangleright}{}_{*}\ \boxed{\textit{SwitchingService}}$

- *Topology* * — hasService ▸ — * *SwitchingService*

2.2.11 hasTopology

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

Allowed relations are:

- *Network Object* * — hasTopology ▸ — * *Topology*

Defined relations are:

- *Topology* * — hasTopology ▸ — * *Topology*

2.2.12 implementedBy

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

Allowed relations are:

- *Network Object* * — implementedBy ▸ — * *Network Object*

Defined relations are:

- *Node* * — implementedBy ▸ — * *Node*

2.2.13 isAlias

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

Allowed relations are:

- *Network Object* * — isAlias ▸ — * *Network Object*

The relation is only defined if the type of both objects is the same (e.g. a Node can be related to another Node, but if it is related to a Topology using the **isAlias** relation, that relation is *undefined*.)

2.2.14 isSerialCompoundLink

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.

Allowed relations are:

- *Link* 1 —isSerialCompoundLink ▸— * [*1.* *Link*, *2.* *Link*, …, *n.* *Link*]

The following relation is allowed, but undefined:

- *LinkGroup* * —isSerialCompoundLink ▸— * [*1.* *LinkGroup*, *2.* *LinkGroup*, …, *n.* *LinkGroup*]

2.2.15 isSink

isSink relates a *Port* to one *Link* to define the outgoing traffic port, and similarly for *PortGroup* and *LinkGroup*.

Allowed relations are:

- *Network Object* * —isSink ▸— * *Link*
- *Network Object* * —isSink ▸— * *LinkGroup*

Defined relations are:

- *Port* * —isSink ▸— * *Link*
- *PortGroup* * —isSink ▸— * *LinkGroup*

isSink between a *PortGroups* and a *LinkGroup* is *defined* only if the *PortGroup* and *LinkGroup* in question have the exact same *LabelGroup*.

2.2.16 isSource

isSource relates a *Port* to one *Link* to define its incoming traffic port, and similarly for *PortGroup* and *LinkGroup*.

Allowed relations are:

- *Network Object* * —isSource ▸— * *Link*
- *Network Object* * —isSource ▸— * *LinkGroup*

Defined relations are:

- $\boxed{\mathit{Port}}\ {}_{*}\ \overline{\text{isSource} \blacktriangleright}\ {}_{*}\ \boxed{\mathit{Link}}$
- $\boxed{\mathit{PortGroup}}\ {}_{*}\ \overline{\text{isSource} \blacktriangleright}\ {}_{*}\ \boxed{\mathit{LinkGroup}}$

isSource between a *PortGroups* and a *LinkGroup* is *defined* only if the *PortGroup* and *LinkGroup* in question have the exact same *LabelGroup*.

2.2.17 locatedAt

locatedAt relates a *Network Object* to one *Location* to describe that a *Network Object* is located at that *Location*.

- $\boxed{\mathit{Network\ Object}}\ {}_{*}\ \overline{\text{locatedAt} \blacktriangleright}\ {}_{*}\ \boxed{\mathit{Location}}$

2.2.18 providesLink

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*.

Allowed relations are:

- $\boxed{\mathit{Service}}\ {}_{*}\ \overline{\text{providesLink} \blacktriangleright}\ {}_{*}\ \boxed{\mathit{Link}}$
- $\boxed{\mathit{Service}}\ {}_{*}\ \overline{\text{providesLink} \blacktriangleright}\ {}_{*}\ \boxed{\mathit{LinkGroup}}$

Defined relations are:

- $\boxed{\mathit{SwitchingService}}\ {}_{1}\ \overline{\text{providesLink} \blacktriangleright}\ {}_{*}\ \boxed{\mathit{Link}}$
- $\boxed{\mathit{SwitchingService}}\ {}_{1}\ \overline{\text{providesLink} \blacktriangleright}\ {}_{*}\ \boxed{\mathit{LinkGroup}}$

2.2.19 providesPort

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*.

Allowed relations are:

- $\boxed{\mathit{Service}}\ {}_{*}\ \overline{\text{providesPort} \blacktriangleright}\ {}_{*}\ \boxed{\mathit{Port}}$
- $\boxed{\mathit{Service}}\ {}_{*}\ \overline{\text{providesPort} \blacktriangleright}\ {}_{*}\ \boxed{\mathit{PortGroup}}$

Defined relations are:

- $\boxed{\mathit{AdaptationService}}\ {}_{1}\!\xrightarrow{\text{providesPort} \blacktriangleright}\!{}_{*}\ \boxed{\mathit{Port}}$
- $\boxed{\mathit{AdaptationService}}\ {}_{1}\!\xrightarrow{\text{providesPort} \blacktriangleright}\!{}_{*}\ \boxed{\mathit{PortGroup}}$
- $\boxed{\mathit{DeadaptationService}}\ {}_{1}\!\xrightarrow{\text{providesPort} \blacktriangleright}\!{}_{*}\ \boxed{\mathit{Port}}$
- $\boxed{\mathit{DeadaptationService}}\ {}_{1}\!\xrightarrow{\text{providesPort} \blacktriangleright}\!{}_{*}\ \boxed{\mathit{PortGroup}}$

2.3 Syntax

The Network Markup Language has two different normative syntaxes. The syntaxes are in regular XML defined using an XML Schema, and another in OWL RDF/XML syntax, defined in an OWL schema. The OWL syntax is aimed at Semantic Web-oriented applications, the XML syntax is suitable for any application. These syntaxes are defined in Appendices A and B respectively.

3 Identifiers

3.1 Object Identifiers

The namespace for the class objects defined in this document is `http://schemas.ogf.org/nml/base/2012/10#`.

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/2012/10/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.

It is possible to describe additional information on an instance, in this case an *idRef* attribute can be used.

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[RFC 3492] 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 Examples

4.1 Examples in XML

The following snippets represent NML structures in the XML format.

- *Topology* (section 2.1.10)

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

  <!-- ... -->

</nml:Topology>
```

- *Node* (section 2.1.2)

```
<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/base#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/base#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* (section 2.1.3)

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

  - *BidirectionalPort* (section 2.1.13)

```
<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* (section 2.1.11)

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

- *Links*
  - *UnidirectionalLink* (section 2.1.4)

```xml
<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/base#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/base#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*

```xml
<nml:Link id="urn:ogf:network:example.net:2012:link_XW">
  <nml:Relation type="http://schemas.ogf.org/nml/2012/10/base#isSerialCompoundLink">
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkA:XY">
      <nml:Relation type="http://schemas.ogf.org/nml/2012/10/base#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/base#next">
        <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:ZW"/>
      </nml:Relation>
    </nml:Link>
    <nml:Link idRef="urn:ogf:network:example.net:2012:linkC:ZW"/>
  </nml:Relation>
</nml:Link>
```

  - *BidirectionalLink* (section 2.1.14)

```xml
<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* (section 2.1.12)

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

- *Labels*
  - *Label* (section 2.1.18)

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

- *LabelGroup* (section 2.1.19)

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

- *Location* (section 2.1.15)

```
<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* (section 2.1.6)

```
<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/base#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/base#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/base#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/base#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* (section 2.1.7)

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

  - *DeadaptationService* (section 2.1.8)

```
<nml:Port id="urn:ogf:network:example.net:2012:port_X.1501:in">
  <nml:Label labeltype="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 >
```

4.2 Examples in OWL

The following snippets represent NML structures in the OWL format.

- *Topology* (section 2.1.10)

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:nml="http://schemas.ogf.org/nml/2012/10/base#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:nmleth="http://schemas.ogf.org/nml/2012/10/ethernet#"
>
<nml:Topology rdf:about="urn:ogf:network:example.net:2012:org">
    <nml:version>20120814</nml:version>

  <!-- ... -->

</nml:Topology>
```

- *Node* (section 2.1.2)

```
<nml:Node rdf:about="urn:ogf:network:example.net:2012:nodeA">
  <nml:name>Node_A</nml:name>
  <nml:locatedAt rdf:resource="urn:ogf:network:example.net:2012:redcity"/>
  <nml:hasOutboundPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_X:out"/>
  <nml:hasOutboundPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_Y:out"/>
  <nml:hasInboundPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_X:in"/>
  <nml:hasInboundPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_Y:in"/>
</nml:Node>
```

- *Ports*
  - *UnidirectionalPort* (section 2.1.3)

```
<nml:Port rdf:about="urn:ogf:network:example.net:2012:port_X:out">
    <nmleth:vlan>1501</nmleth:vlan>
</nml:Port>
```

  - *BidirectionalPort* (section 2.1.13)

```
<nml:BidirectionalPort rdf:about="urn:ogf:network:example.net:2012:port_X">
  <nml:name>X</nml:name>
  <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:port_X:out"/>
  <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:port_X:in"/>
```

```
</nml:BidirectionalPort>
```

- *PortGroup* (section 2.1.11)

```
<nml:PortGroup rdf:about="urn:ogf:network:example.net:2012:portgroup_X:out">
    <nml:hasLabel>
      <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
        1780−1783
      </nml:LabelGroup>
    </nml:hasLabel>
</nml:PortGroup>
```

- *Links*
  - *UnidirectionalLink* (section 2.1.4)

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

<nml:Port rdf:about="urn:ogf:network:example.net:2012:port_X:out">
  <nml:isSource rdf:resource="urn:ogf:network:example.net:2012:linkA:XY"/>
</nml:Port>

<nml:Port id="urn:ogf:network:example.net:2012:port_Y:in">
  <nml:isSink rdf:resource="urn:ogf:network:example.net:2012:linkA:XY"/>
</nml:Port>
```

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

```
<nml:Link rdf:about="urn:ogf:network:example.net:2012:link_XW">
  <nml:isSerialCompoundLink>
    <nml:ListItem rdf:resource="urn:ogf:network:example.net:2012:link_XW_1">
      <nml:item rdf:resource="urn:ogf:network:example.net:2012:linkA:XY"/>
      <nml:next rdf:resource="urn:ogf:network:example.net:2012:link_XW_2"/>
    </nml:ListItem>
  </nml:isSerialCompoundLink>
</nml:Link>

<nml:ListItem rdf:resource="urn:ogf:network:example.net:2012:link_XW_2">
  <nml:item rdf:resource="urn:ogf:network:example.net:2012:linkB:YZ"/>
  <nml:next rdf:resource="urn:ogf:network:example.net:2012:link_XW_3"/>
</nml:ListItem>

<nml:ListItem rdf:resource="urn:ogf:network:example.net:2012:link_XW_3">
  <nml:item rdf:resource="urn:ogf:network:example.net:2012:linkC:ZW"/>
</nml:ListItem>
```

  - *BidirectionalLink* (section 2.1.14)

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

  - *LinkGroup* (section 2.1.12)

```
<nml:LinkGroup rdf:about="urn:ogf:network:example.net:2012:domainy_domainx">
```

```
  <nmleth:vlan>1780−1783</nmleth:vlan>
</nml:LinkGroup>
```

- *Labels*
  - *Label* (section 2.1.18)

```
<rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
  <owl:subPropertyOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#hasLabel" />
</rdf:Description>
</rdf:Description><nmleth:vlan>1501</nmleth:vlan>
```

  - *LabelGroup* (section 2.1.19)

```
<rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">
  <owl:subPropertyOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#hasLabel" />
</rdf:Description>
<nmleth:vlan>1780−1783</nmleth:vlan>
```

- *Location* (section 2.1.15)

```
<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* (section 2.1.6)

```
<nml:Node rdf:about="urn:ogf:network:example.net:2012:nodeA">
  <nml:name>Node_A</nml:name>
  <nml:hasInboundPort>
    <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_X:in" />
    <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_Y:in" />
  </nml:hasInboundPort>
  <nml:hasOutboundPort>
    <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_X:out"/>
    <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_Y:out"/>
  </nml:hasOutboundPort>
  <nml:hasService rdf:about="urn:ogf:network:example.net:2012:nodeA:switchingService"/>
</nml:Node>

<nml:SwitchingService rdf:about="urn:ogf:network:example.net:2012:nodeA:switchingService">
  <nml:hasInboundPort>
    <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_X:in" />
    <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_Y:in" />
  </nml:hasInboundPort>
  <nml:hasOutboundPort>
    <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_X:out"/>
    <nml:hasPort rdf:resource="urn:ogf:network:example.net:2012:nodeA:port_Y:out"/>
  </nml:hasOutboundPort>
</nml:SwitchingService>
```

  - *AdaptationService* (section 2.1.7)

```
<nml:Port rdf:about="urn:ogf:network:example.net:2012:port_X:in">
  <nml:hasService rdf:resource="urn:ogf:network:example.net:2012:port_X:in:adaptationService">
</nml:Port>

<nml:AdaptationService
     rdf:about="urn:ogf:network:example.net:2012:port_X:in:adaptationService">
  <nml:providesPort rdf:resource="urn:ogf:network:example.net:2012:port_X.1501:in"/>
</nml:AdaptationService>

<nml:Port rdf:about="urn:ogf:network:example.net:2012:port_X.1501:in">
  <nmleth:vlan>1501</nmleth:vlan>
</nml:Port>
```

– *DeadaptationService* (section 2.1.8)

```
<nml:Port rdf:about="urn:ogf:network:example.net:2012:port_X.1501:in">
  <nmleth:vlan>1501</nmleth:vlan>
  <nml:hasService>
    <nml:DeadaptationService
      rdf:resource="urn:ogf:network:example.net:2012:port_X.1501:in:deadaptationService">
      <nml:providesPort rdf:about="urn:ogf:network:example.net:2012:port_X:in"/>
    </nml:DeadaptationService>
  </nml:hasService>
</nml:Port>
```

5 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 the integrity nor the authenticity. The NML documents also can not provide this for the identifiers contained in the documents. Implementers should use external means of verifying the authenticity of identifiers contained in the documents.

6 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 900
Washington, DC 20036
USA
Email: zurawski@internet2.edu

7 Acknowledgments

The authors like to thank the NML working group members for their patience. The NML group has operated in the web of infrastructure groups and is thankful for all the input from the NM, NMC and NSI working-groups.

8 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.

9 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.

10 Full Copyright Notice

Appendix A XML Schema

This section describes the normative schema of XML documents using the XML Schema language.

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


<!--

 File: nmlbase.xsd - Main XSD schema definition
 Version: $Id$
 Purpose: This is the main XSD schema file, it defines the
             general topology elements of NML.

-->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://schemas.ogf.org/nml/2012/10/base#"
           xmlns:nml="http://schemas.ogf.org/nml/2012/10/base#"
           elementFormDefault="qualified">



  <xs:complexType name="NetworkObject">
    <xs:sequence>
      <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="relation" type="nml:RelationType" minOccurs="0" />
      <xs:element name="Location" type="nml:LocationType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="parameter" type="nml:ParameterType" minOccurs="0"/>
      <xs:choice>
        <xs:element name="lifetime" type="nml:LifeTimeType" minOccurs="0" maxOccurs="1"/>
        <xs:element name="existDuring" type="nml:ExistDuringType" minOccurs="0" maxOccurs="1"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:anyURI" use="optional"/>
    <xs:attribute name="idRef" type="xs:anyURI" use="optional"/> <!-- referencing and inheritance -->
    <xs:attribute name="version" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>


  <xs:complexType name="RelationType">
    <xs:choice>
      <xs:element ref="nml:Node"/>
      <xs:element ref="nml:Port" minOccurs="1"/>
      <xs:element ref="nml:PortGroup"/>
      <xs:element ref="nml:Link"/>
      <xs:element ref="nml:LinkGroup"/>
    </xs:choice>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="http://schemas.ogf.org/nml/2012/10/base#implementedBy"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2012/10/base#hasInboundPort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2012/10/base#hasOutboundPort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2012/10/base#isSource"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2012/10/base#isSink"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2012/10/base#isSerialCompoundLink"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2012/10/base#isAlias"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2012/10/base#next"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>


  <xs:complexType name="LocationType">
    <xs:all>
      <xs:element name="long" type="xs:float" minOccurs="0" maxOccurs="1"/>
      <xs:element name="lat" type="xs:float" minOccurs="0" maxOccurs="1"/>
      <xs:element name="alt" type="xs:float" minOccurs="0" maxOccurs="1"/>
      <xs:element name="unlocode" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <!-- address: rfc6351 xCard: vCard XML Representation -->
      <xs:element name="address" minOccurs="0" maxOccurs="1">
        <xs:complexType>
```

```
        <xs:sequence>
          <xs:any namespace="##other" processContents="lax" minOccurs="1" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>

<xs:complexType name="ParameterType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>


<xs:complexType name="LifeTimeType">
  <xs:sequence>
    <xs:element name="start" type="xs:dateTime"/>
     <xs:choice>
       <xs:element name="end" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
       <xs:element name="duration" type="xs:duration" minOccurs="0" maxOccurs="1"/>
     </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:anyURI" use="optional"/>
</xs:complexType>

<xs:complexType name="ExistDuringType">
  <xs:attribute name="idRef" type="xs:anyURI" use="required"/>
</xs:complexType>

<xs:group name="Group">
  <xs:choice>
    <xs:element ref="nml:Topology"/>
    <xs:element ref="nml:LinkGroup"/>
    <xs:element ref="nml:PortGroup"/>
    <xs:element ref="nml:BidirectionalLink"/>
    <xs:element ref="nml:BidirectionalPort"/>
  </xs:choice>
</xs:group>

<!-- Topology -->

<xs:group name="BaseTopologyContent">
  <xs:sequence>
    <xs:element ref="nml:Link" minOccurs="0"/>
    <xs:element ref="nml:Port" minOccurs="0"/>
    <xs:element ref="nml:Node" minOccurs="0"/>
    <xs:group ref="nml:Service" minOccurs="0"/>
    <xs:group ref="nml:Group" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" />
  </xs:sequence>
</xs:group>

<xs:complexType name="TopologyType">
  <xs:complexContent>
    <xs:extension base="nml:NetworkObject">
      <xs:group ref="nml:BaseTopologyContent"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="Topology" type="nml:TopologyType"/>

<!-- Link -->

<xs:group name="BaseLinkContent">
  <xs:sequence>
    <xs:element ref="nml:Label" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" />
```

```
    </xs:sequence>
  </xs:group>

  <xs:complexType name="LinkType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:group ref="nml:BaseLinkContent"/>
        <xs:attribute name="encoding" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="Link" type="nml:LinkType"/>

  <!-- Port -->

  <xs:group name="BasePortContent">
    <xs:sequence>
      <xs:element ref="nml:Label" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" />
    </xs:sequence>
  </xs:group>

  <xs:complexType name="PortType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:group ref="nml:BasePortContent"/>
        <xs:attribute name="encoding" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="Port" type="nml:PortType"/>

  <!-- Node -->

  <xs:complexType name="NodeType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:any namespace="##other" processContents="lax" minOccurs="0" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="Node" type="nml:NodeType"/>

  <!-- Service -->

  <xs:group name="Service">
    <xs:choice>
      <xs:element ref="nml:SwitchingService"/>
      <xs:element ref="nml:AdaptationService"/>
      <xs:element ref="nml:DeadaptationService"/>
    </xs:choice>
  </xs:group>

  <xs:complexType name="SwitchingServiceType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:element ref="nml:Link" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="SwitchingService" type="nml:SwitchingServiceType"/>
```

```
<xs:complexType name="AdaptationServiceType">
  <xs:complexContent>
    <xs:extension base="nml:NetworkObject">
      <xs:sequence>
        <xs:element ref="nml:Port" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="adaptationFunction" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="AdaptationService" type="nml:AdaptationServiceType"/>

<xs:complexType name="DeadaptationServiceType">
  <xs:complexContent>
    <xs:extension base="nml:NetworkObject">
      <xs:sequence>
        <xs:element ref="nml:Port" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="deadaptationFunction" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="DeadaptationService" type="nml:DeadaptationServiceType"/>

<!-- Label -->

<xs:complexType name="LabelType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="labeltype" type="xs:anyURI" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="Label" type="nml:LabelType"/>

<!-- LinkGroup -->

<xs:group name="BaseLinkGroup">
  <xs:sequence>
    <xs:element ref="nml:LabelGroup" minOccurs="0"/>
    <xs:element ref="nml:Link" minOccurs="0"/>
    <xs:element ref="nml:LinkGroup" minOccurs="0"/>
  </xs:sequence>
</xs:group>

<xs:complexType name="LinkGroupType">
  <xs:complexContent>
    <xs:extension base="nml:NetworkObject">
      <xs:group ref="nml:BaseLinkGroup"/>
      <xs:attribute name="encoding" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="LinkGroup" type="nml:LinkGroupType"/>

<!-- PortGroup -->

<xs:group name="BasePortGroup">
  <xs:sequence>
    <xs:element ref="nml:LabelGroup" minOccurs="0"/>
    <xs:element ref="nml:Port" minOccurs="0"/>
    <xs:element ref="nml:PortGroup" minOccurs="0"/>
  </xs:sequence>
```

```
</xs:group>

<xs:complexType name="PortGroupType">
  <xs:complexContent>
    <xs:extension base="nml:NetworkObject">
      <xs:group ref="nml:BasePortGroup"/>
      <xs:attribute name="encoding" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="PortGroup" type="nml:PortGroupType"/>


<!-- BidirectionalLink -->

<xs:group name="BaseBidirectionalLink">
  <xs:choice>
    <xs:sequence>
      <xs:element ref="nml:Link"/>
      <xs:element ref="nml:Link"/>
    </xs:sequence>
    <xs:sequence>
      <xs:element ref="nml:LinkGroup"/>
      <xs:element ref="nml:LinkGroup"/>
    </xs:sequence>
  </xs:choice>
</xs:group>

<xs:complexType name="BidirectionalLinkType">
  <xs:complexContent>
    <xs:extension base="nml:NetworkObject">
      <xs:group ref="nml:BaseBidirectionalLink"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="BidirectionalLink" type="nml:BidirectionalLinkType"/>

<!-- BidirectionalPort -->

<xs:group name="BaseBidirectionalPort">
  <xs:choice>
    <xs:sequence>
      <xs:element ref="nml:Port"/>
      <xs:element ref="nml:Port"/>
    </xs:sequence>
    <xs:sequence>
      <xs:element ref="nml:PortGroup"/>
      <xs:element ref="nml:PortGroup"/>
    </xs:sequence>
  </xs:choice>
</xs:group>

<xs:complexType name="BidirectionalPortType">
  <xs:complexContent>
    <xs:extension base="nml:NetworkObject">
      <xs:group ref="nml:BaseBidirectionalPort"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="BidirectionalPort" type="nml:BidirectionalPortType"/>

<!-- LabelGroup -->

<xs:complexType name="LabelGroupType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
```

```
        <xs:attribute name="labeltype" type="xs:anyURI" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:element name="LabelGroup" type="nml:LabelGroupType"/>

</xs:schema>
```

Appendix B OWL Schema

This section describes the normative schema of the OWL syntax using the OWL ontology definition below.

The XML and OWL schemas have a different approach to defining the NML *List*. In XML a *List* can be constructed using an additional *next* attribute on an object. The OWL syntax does not use attributes, and using a*next* object property on an object would cause confusion if an item is in multiple lists. Therefore we have introduced a *ListItem* object, which holds the value through the *item* relationship, and also has an optional *next* item to relate to the next item of the list. See also the SerialCompoundLink examples in the example section.

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://schemas.ogf.org/nml/2012/10/base#"
     xml:base="http://schemas.ogf.org/nml/2012/10/base#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:nml="http://schemas.ogf.org/nml/2012/10/base#">
    <owl:Ontology rdf:about="http://schemas.ogf.org/nml/2012/10/base#">
        <rdfs:label>NML Schema</rdfs:label>
    </owl:Ontology>
    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#existsDuring">
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Lifetime"/>
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasInboundPort">
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Port"/>
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Node"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#SwitchingService"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Topology"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasLabel">
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Label"/>
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Link"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Port"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasLabelGroup">
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#LabelGroup"/>
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#PortGroup"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasLink">
        <rdfs:range>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Link"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#LinkGroup"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:range>
        <rdfs:domain>
            <owl:Class>
```

```
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#BidirectionalLink"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#LinkGroup"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasNode">
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Node"/>
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Topology"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasOutboundPort">
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Port"/>
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Node"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#SwitchingService"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Topology"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasPort">
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Port"/>
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#BidirectionalPort"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#PortGroup"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasService">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Service"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasSink">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Link"/>
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Port"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasSource">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Link"/>
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Port"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#hasTopology">
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Topology"/>
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Topology"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#implementedBy">
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Node"/>
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Node"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#isAlias">
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#isSerialCompoundLink">
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#List"/>
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Link"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#LinkGroup"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#isSink">
```

```
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Link"/>
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#LinkGroup"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Port"/>
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#PortGroup"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#isSource">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Port"/>
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#PortGroup"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Link"/>
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#LinkGroup"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#item">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#ListItem"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#locatedAt">
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Location"/>
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Node"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#next">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#ListItem"/>
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2012/10/base#ListItem"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#providesLink">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#SwitchingService"/>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Link"/>
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#LinkGroup"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#providesPort">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#AdaptationService"/>
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#DeadaptationService"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Port"/>
          <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#PortGroup"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>
```

```
<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#adaptationfunction">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#AdaptationService"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#DeadaptationService"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#address">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Location"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#alt">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Location"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#encoding">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Port"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#PortGroup"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#endtime">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Lifetime"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#labeltype">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Label"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#LabelGroup"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#labelvalue">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Label"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#labelvalues">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#LabelGroup"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#lat">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Location"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#long">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Location"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#name">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#Location"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
            </owl:unionOf>
        </owl:Class>
```

```
        </rdfs:domain>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#starttime">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Lifetime"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#time">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Lifetime"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#unlocode">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Location"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2012/10/base#version">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Topology"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#AdaptationService">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Service"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#BidirectionalLink">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Group"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#BidirectionalPort">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Group"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#DeadaptationService">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Service"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#Group">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#Label"/>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#LabelGroup">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#Lifetime"/>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#Link">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#LinkGroup">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Group"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#List"/>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#ListItem">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#Location">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#Node">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#Port">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#PortGroup">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Group"/>
```

```
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#Service">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#NetworkObject"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#SwitchingService">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Service"/>
    </owl:Class>

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2012/10/base#Topology">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2012/10/base#Group"/>
    </owl:Class>
</rdf:RDF>
```

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.

[ISO 8601] Data elements and interchange formats – Information interchange – Representation of dates and times. ISO 8601:2004 (Third edition), December 2004. Section 4.3.2 (a), Complete representations of a date and time. Calendar date in basic format. URL http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=40874.

[RDF-XML] Dave Beckett (editor) RDF/XML Syntax Specification (Revised) W3C Recommendation 10 February 2004. URL http://www.w3.org/TR/rdf-syntax-grammar/.

[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 3492] A. Costello Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA) RFC 3492 (Standards Track), March 2003 URL http://tools.ietf.org/html/rfc3492.

[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.

[UNLOCODE] United Nations Code for Trade and Transport Locations UN/LOCODE, revision 2012-01, September 2012. URL http://www.unece.org/cefact/locode/welcome.html.

[WGS84] Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems NIMA Technical Report TR8350.2, Third Edition, June 2004 URL http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html.

[XML] Henry S. Thompson, David Beech, Murray Maloney and Noah Mendelsohn XML Schema Part 1: Structures Second Edition W3C Recommendation 28 October 2004. URL http://www.w3.org/TR/xmlschema-1/.

Informative References

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

[RFC 6350] Simon Perreault. vCard Format Specification RFC 6350 (Standards Track), August 2011. URL http://tools.ietf.org/html/rfc6350.

[RFC 6351] S. Perreault. xCard: vCard XML Representation RFC 6351 (Standards Track), August 2011. URL http://tools.ietf.org/html/rfc6351.

[RDFVCARD] Harry Halpin, Renato Iannella, Brian Suda, Norman Walsh Representing vCard Objects in RDF W3C Member Submission 20 January 2010. URL http://www.w3.org/TR/vcard-rdf/.