GFD-R-P.206
NML-WG
nml-wg@ogf.org

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

May 2013

Network Markup Language Base Schema version 1

Status of This Document

Grid Final Draft (GFD), Proposed Recommendation (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 5 provides example use cases. This section is informative. Only sections 2, 3, 4, and appendices A and B are normative and considered part of the recommendation.

Appendix C is informative and explains the relation between terms defined in this document and those defined in the ITU-T G.800 recommendation [G.800].

1.1 Context

The Network Markup Language (NML) has been defined in the context of research and education networks to describe so-called hybrid network topologies. The NML is defined as an abstract and generic model, so it can be applied for other network topologies as well. See [GFD.165] for an detailed overview including prior work.

1.2 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 primarily static network topology, but also its potential capabilities (services) and its configuration.

NML is aimed at logical connection-oriented network topologies, more precisely topologies where switching is performed on a label associated with a flow, such as a VLAN, wavelength or time slot. NML 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, this document omits a definition for the terms *Network* or *capacity*. 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 header and footer overhead) that it is better to let technology-specific extensions make an explicit definition.

1.3 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). Names of identifiers and string literals are written in monspaces font (e.g. `Port_X:in`).

Diagrams in this document follow the diagrammatic conventions of UML class diagrams.

- A subclass-superclass relationship is represented by a line with hollow triangle shape pointing to the superclass.
- A whole-part relationship is represented by a line with a hollow diamond shape pointing to the whole (group).
- A entity-relationship is represented by a line, optionally with numbers at each end indicating the cardinality of the relation. A named entity-relationship has a verb next to the line, and a filled triangle pointing to the object of the verb. (e.g. the entitity-relationship $\boxed{\mathit{BidirectionalPort}} \overset{\mathit{hasPort} \blacktriangleright}{\underset{* \qquad\qquad 2}{\rule[0.5ex]{6em}{0.4pt}}} \boxed{\mathit{Port}}$ is named *hasPort*, and each *BidirectionalPort* is related to exactly 2 *Port*s, and each *Port* may be associated with zero, one or more *BidirectionalPort*s.)

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. Parameters, like attributes, are properties of classes, but may (subtly) change the logic. 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/2013/05/base#`.

2.1 Classes

Figure 1 shows an overview of all the classes in the NML schema in a UML class diagram. Each box defines the name of a class, a short description, and possible attributes with their

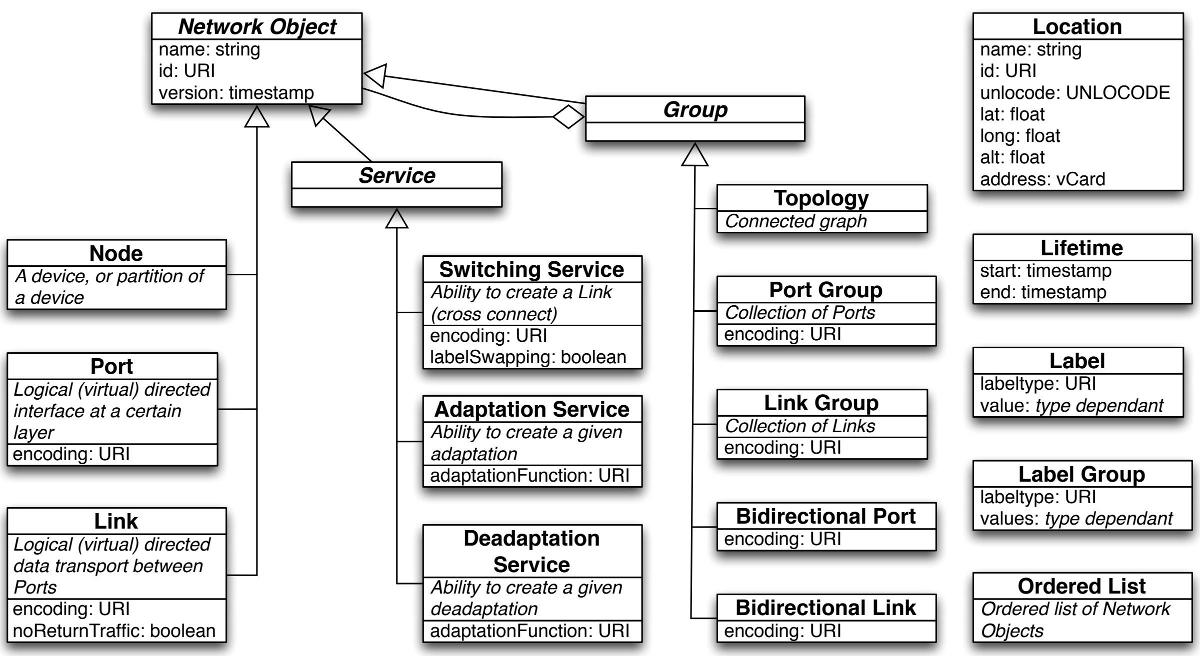


Figure 1: A UML class diagram of the classes in the NML schema and their hierarchy

value type. 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

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

The meaning of the *version* attribute is only defined for specific cases (for 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. Names are not globally unique, and two objects can 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, 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.

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, e.g. a *Port*, that has a *isSource* relation to the *Link*. A sink of a Link is a Network Object, e.g. a *Port*, that has a *isSink* relation to the *Link*.

A *Link* object can refer to any link connection. A link segment and an end-to-end path are both described by a *Link* object. The composition of links into a path, and decomposition into link segments is described by the *isSerialCompoundLink* relation.

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`.

  An example of where this is used is in an Ethernet broadcast domain, where broadcast traffic is sent to all sinks, except the sink *Port*s associated with the sending source *Port*.

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

2.1.5 Service

Service describes an ability 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:

- *encoding* to assign a data encoding identifier
- *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 have 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*.

A *Port* object can have a *hasService* relation, however the *SwitchingService* defines a more specific relation *hasInboundPort* / *hasOutboundPort* relation to a *Port* object. The latter relation is preferred over the *hasService* relation of the Port to the SwitchingService.

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

2.1.7 Adaptation Service

An *AdaptationService* describes the ability 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 (this describes a ability)
- *existsDuring* to one or more *Lifetime*s
- *isAlias* to one or more *AdaptationService*s
- *providesPort* to one or more *Port*s or *PortGroup*s (this describes a configuration)

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 *parallelCompound* relation can be used, which is a future extension relation, described in a separate document [Dijkstra13].

2.1.8 De-adaptation Service

A *DeadaptationService* describes the ability 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*. An object can also be part of multiple *Group*s.

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,

[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.

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
- *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 *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. See Figure 2 for an example of a *BidirectionalPort* and its associated *Port*s.

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:

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

There is explicitly no direct relation between a *BidirectionalPort* and a *BidirectionalLink*, since NML is a unidirectional model.

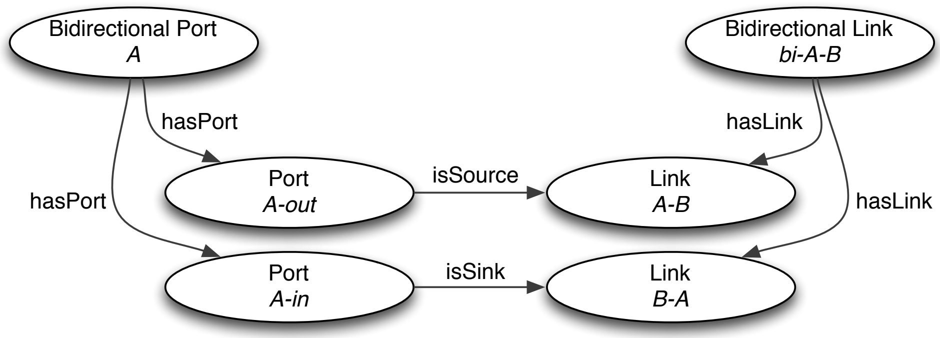


Figure 2: An abstract example of *BidirectionalPort* and *BidirectionalLink*

2.1.14 Bidirectional Link

A *BidirectionalLink* is a group of two (unidirectional) *Link*s or *LinkGroup*s together forming a bidirectional link. See Figure 2 for an example of a *BidirectionalLink* and its associated *Link*s.

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:

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

There is explicitly no direct relation between a *BidirectionalPort* and a *BidirectionalLink*, since NML is a unidirectional model.

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. The end of that assumed lifetime is indefinite, until a Topology object with a higher version number is published. This new description can define a new Lifetime for the object, or the Topology. If the new description does not contain the Network Object, the end time is assumed to have passed.

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 Label

A *Label* is the technology-specific value that distinguishes a single data stream (a channel) embedded in a larger data stream. The *Label* can be a resource label (with one value). In a future extension it may be a pair of source and destination labels (with two values) [G.800]. Examples of resource labels are a VLAN number, wavelength, et cetera.

A *Label* may have the following attributes:

- *labeltype* to refer to a technology-specific labelset, e.g. a URI for VLANs
- *value* is one specific value taken from the labelset, e.g. a VLAN number

Technology extensions of NML may define additional attributes. Label type URIs SHOULD be specified in a Grid Forum Documents (GFD), which SHOULD also define possible values.

This version of NML only deals with resource labels. The use of source and destination labels is a future extension [Dijkstra13].

2.1.18 Label Group

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

A *LabelGroup* may have the following attributes:

- *labeltype* 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.

2.1.19 Ordered List

An *Ordered List* 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 *Ordered List* depends on the syntax, and is defined in section 4.4.

2.1.20 List Item

A *ListItem* is a syntactical construct which may be used by syntaxes to construct a *Ordered List*. The exact usage depends on the syntax.

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 here (in alphabetical order). In principle a *Relation* can go from any object to any other object.

The list below makes 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* MUST be rejected by a client, and the sender SHOULD be notified with an error. A relation which is *allowed*, but (yet) *undefined* SHOULD be ignored by a client (either silently, or with a warning to the sender). 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 a syntax 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:

- $\boxed{\textit{Service}} \; * \xrightarrow{\textit{canProvidePort} \;\blacktriangleright} * \; \boxed{\textit{Port}}$
- $\boxed{\textit{Service}} \; * \xrightarrow{\textit{canProvidePort} \;\blacktriangleright} * \; \boxed{\textit{PortGroup}}$

Defined relations are:

- $\boxed{\textit{AdaptationService}} \; * \xrightarrow{\textit{canProvidePort} \;\blacktriangleright} * \; \boxed{\textit{Port}}$
- $\boxed{\textit{AdaptationService}} \; * \xrightarrow{\textit{canProvidePort} \;\blacktriangleright} * \; \boxed{\textit{PortGroup}}$
- $\boxed{\textit{DeadaptationService}} \; * \xrightarrow{\textit{canProvidePort} \;\blacktriangleright} * \; \boxed{\textit{Port}}$
- $\boxed{\textit{DeadaptationService}} \; * \xrightarrow{\textit{canProvidePort} \;\blacktriangleright} * \; \boxed{\textit{PortGroup}}$

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.

$\boxed{\textit{Network Object}}\ {}_{1}\xrightarrow{\textit{existsDuring}\ \blacktriangleright}{}_{*}\ \boxed{\textit{LifeTime}}$

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, then 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:

- $\boxed{\textit{Network Object}}\ {}_{*}\xrightarrow{\textit{hasInboundPort}\ \blacktriangleright}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{Network Object}}\ {}_{*}\xrightarrow{\textit{hasInboundPort}\ \blacktriangleright}{}_{*}\ \boxed{\textit{PortGroup}}$

Defined relations are:

- $\boxed{\textit{Node}}\ {}_{*}\xrightarrow{\textit{hasInboundPort}\ \blacktriangleright}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{Node}}\ {}_{*}\xrightarrow{\textit{hasInboundPort}\ \blacktriangleright}{}_{*}\ \boxed{\textit{PortGroup}}$
- $\boxed{\textit{SwitchingService}}\ {}_{*}\xrightarrow{\textit{hasInboundPort}\ \blacktriangleright}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{SwitchingService}}\ {}_{*}\xrightarrow{\textit{hasInboundPort}\ \blacktriangleright}{}_{*}\ \boxed{\textit{PortGroup}}$
- $\boxed{\textit{Topology}}\ {}_{*}\xrightarrow{\textit{hasInboundPort}\ \blacktriangleright}{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{Topology}}\ {}_{*}\xrightarrow{\textit{hasInboundPort}\ \blacktriangleright}{}_{*}\ \boxed{\textit{PortGroup}}$

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*). This *Port* would then be related to a *Link* object using the *isSink* relation (or a *PortGroup* and *LinkGroup* respectively).

A *Network Object* with a *hasInboundPort* relation pointing to a *PortGroup* has the same meaning as defining a *hasInboundPort* relation pointing to every *Port* in that *PortGroup* (as defined by a *hasPort* relation between the *PortGroup* and *Port*).

2.2.4 hasLabel

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

Allowed relations are:

- $\boxed{Port}\,{}_{1}\xrightarrow{\textit{hasLabel}\ \blacktriangleright}{}_{*}\,\boxed{Label}$
- $\boxed{Link}\,{}_{1}\xrightarrow{\textit{hasLabel}\ \blacktriangleright}{}_{*}\,\boxed{Label}$

The *Label* assigned to a *Port* or *Link* is the technology label that identifies the traffic through this *Port* or *Link* (including in *Link*s provided by a *SwitchingMatrix*).

A *Label* is used to distinguish a *Port* in a *PortGroup*, or distinguish a *Link* in a *LinkGroup*.

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:

- $\boxed{PortGroup}\,{}_{1}\xrightarrow{\textit{hasLabelGroup}\ \blacktriangleright}{}_{*}\,\boxed{LabelGroup}$
- $\boxed{LinkGroup}\,{}_{1}\xrightarrow{\textit{hasLabelGroup}\ \blacktriangleright}{}_{*}\,\boxed{LabelGroup}$

The *LabelGroup* assigned to this *PortGroup* or *LinkGroup* defines the *Label*s associated with the *Port*s member of that group. There MUST be a one-to-one correspondence between the *LabelGroup* and 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:

- $\boxed{\textit{Group}}\ {*}\xrightarrow{\textit{hasLink}}{*}\ \boxed{\textit{Link}}$
- $\boxed{\textit{Group}}\ {*}\xrightarrow{\textit{hasLink}}{*}\ \boxed{\textit{LinkGroup}}$

Defined relations are:

- $\boxed{\textit{LinkGroup}}\ {*}\xrightarrow{\textit{hasLink}}{*}\ \boxed{\textit{Link}}$
- $\boxed{\textit{LinkGroup}}\ {*}\xrightarrow{\textit{hasLink}}{*}\ \boxed{\textit{LinkGroup}}$
- $\boxed{\textit{BidirectionalLink}}\ {*}\xrightarrow{\textit{hasLink}}{2}\ \boxed{\textit{Link}}$
- $\boxed{\textit{BidirectionalLink}}\ {*}\xrightarrow{\textit{hasLink}}{2}\ \boxed{\textit{LinkGroup}}$

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:

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

Defined relations are:

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

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:

- $\boxed{\textit{Network Object}} \xrightarrow[* \qquad\qquad *]{\textit{hasOutboundPort} \blacktriangleright} \boxed{\textit{Port}}$
- $\boxed{\textit{Network Object}} \xrightarrow[* \qquad\qquad *]{\textit{hasOutboundPort} \blacktriangleright} \boxed{\textit{PortGroup}}$

Defined relations are:

- $\boxed{\textit{Node}} \xrightarrow[* \qquad\qquad *]{\textit{hasOutboundPort} \blacktriangleright} \boxed{\textit{Port}}$
- $\boxed{\textit{Node}} \xrightarrow[* \qquad\qquad *]{\textit{hasOutboundPort} \blacktriangleright} \boxed{\textit{PortGroup}}$
- $\boxed{\textit{SwitchingService}} \xrightarrow[* \qquad\qquad *]{\textit{hasOutboundPort} \blacktriangleright} \boxed{\textit{Port}}$
- $\boxed{\textit{SwitchingService}} \xrightarrow[* \qquad\qquad *]{\textit{hasOutboundPort} \blacktriangleright} \boxed{\textit{PortGroup}}$
- $\boxed{\textit{Topology}} \xrightarrow[* \qquad\qquad *]{\textit{hasOutboundPort} \blacktriangleright} \boxed{\textit{Port}}$
- $\boxed{\textit{Topology}} \xrightarrow[* \qquad\qquad *]{\textit{hasOutboundPort} \blacktriangleright} \boxed{\textit{PortGroup}}$

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*). This *Port* would then be related to a *Link* object using the *isSource* relation (or az *PortGroup* and *LinkGroup* respectively).

A *Network Object* with a *hasOutboundPort* relation pointing to a *PortGroup* has the same meaning as defining a *hasOutboundPort* relation pointing to every *Port* in that *PortGroup* (as defined by a *hasPort* relation between the *PortGroup* and *Port*).

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:

- $\boxed{\textit{Group}} \xrightarrow[* \qquad\quad *]{\textit{hasPort} \blacktriangleright} \boxed{\textit{Port}}$
- $\boxed{\textit{Group}} \xrightarrow[* \qquad\quad *]{\textit{hasPort} \blacktriangleright} \boxed{\textit{PortGroup}}$

Defined relations are:

- $\boxed{\textit{PortGroup}}\ {}_{*}\!\xrightarrow{\textit{hasPort}\ \blacktriangleright}\!{}_{*}\ \boxed{\textit{Port}}$
- $\boxed{\textit{PortGroup}}\ {}_{*}\!\xrightarrow{\textit{hasPort}\ \blacktriangleright}\!{}_{*}\ \boxed{\textit{PortGroup}}$
- $\boxed{\textit{BidirectionalPort}}\ {}_{*}\!\xrightarrow{\textit{hasPort}\ \blacktriangleright}\!{}_{2}\ \boxed{\textit{Port}}$
- $\boxed{\textit{BidirectionalPort}}\ {}_{*}\!\xrightarrow{\textit{hasPort}\ \blacktriangleright}\!{}_{2}\ \boxed{\textit{PortGroup}}$

The *hasPort* relationships for a *BidirectionalPort* point to the two unidirectional *Port*s that together form a bidirectional port for the associated *Node*. These *Port*s would have a *hasInboundPort* and *hasOutboundPort* relation with that *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 ability of that *Node* or *Topology*.

Allowed relations are:

- $\boxed{\textit{Network Object}}\ {}_{*}\!\xrightarrow{\textit{hasService}\ \blacktriangleright}\!{}_{*}\ \boxed{\textit{Service}}$

Defined relations are:

- $\boxed{\textit{Port}}\ {}_{1}\!\xrightarrow{\textit{hasService}\ \blacktriangleright}\!{}_{*}\ \boxed{\textit{AdaptationService}}$
- $\boxed{\textit{Port}}\ {}_{1}\!\xrightarrow{\textit{hasService}\ \blacktriangleright}\!{}_{*}\ \boxed{\textit{DeadaptationService}}$
- $\boxed{\textit{Node}}\ {}_{*}\!\xrightarrow{\textit{hasService}\ \blacktriangleright}\!{}_{*}\ \boxed{\textit{SwitchingService}}$
- $\boxed{\textit{Topology}}\ {}_{*}\!\xrightarrow{\textit{hasService}\ \blacktriangleright}\!{}_{*}\ \boxed{\textit{SwitchingService}}$

A *Port* object can have a *hasService* relation to a *Service*, however the *SwitchingService* defines a more specific relation *hasInboundPort* / *hasOutboundPort* relation to a *Port* object. The latter relation is preferred over the *hasService* relation of the *Port* to the *SwitchingService*.

2.2.11 hasTopology

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

Allowed relations are:

- $\boxed{\textit{Network Object}} \xrightarrow[* \qquad\qquad *]{\textit{hasTopology} \blacktriangleright} \boxed{\textit{Topology}}$

Defined relations are:

- $\boxed{\textit{Topology}} \xrightarrow[* \qquad\qquad *]{\textit{hasTopology} \blacktriangleright} \boxed{\textit{Topology}}$

2.2.12 implementedBy

implementedBy relates a *Node* to one or more *Node*s to describe virtualization or partitioning of a *Node*. The relation MAY be recursive, thus a virtual *Node* MAY be further partitioned.

Allowed relations are:

- $\boxed{\textit{Network Object}} \xrightarrow[* \qquad\qquad *]{\textit{implementedBy} \blacktriangleright} \boxed{\textit{Network Object}}$

Defined relations are:

- $\boxed{\textit{Node}} \xrightarrow[* \qquad\qquad *]{\textit{implementedBy} \blacktriangleright} \boxed{\textit{Node}}$

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:

- $\boxed{\textit{Network Object}} \xrightarrow[* \qquad *]{\textit{isAlias} \blacktriangleright} \boxed{\textit{Network Object}}$

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.

The following relation is allowed and defined:

- *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:

- $\boxed{\textit{Port}}\ {*}\overset{\textit{isSource}\ \blacktriangleright}{\rule{3cm}{0.4pt}}{*}\ \boxed{\textit{Link}}$
- $\boxed{\textit{PortGroup}}\ {*}\overset{\textit{isSource}\ \blacktriangleright}{\rule{3cm}{0.4pt}}{*}\ \boxed{\textit{LinkGroup}}$

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 item

A *item* relation is a syntactical construct which may be used by syntaxes to construct a *Ordered List*. The exact usage depends on the syntax.

2.2.18 locatedAt

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

- $\boxed{\textit{Network Object}}\ {*}\overset{\textit{locatedAt}\ \blacktriangleright}{\rule{3cm}{0.4pt}}{*}\ \boxed{\textit{Location}}$

2.2.19 next

next relation is a syntactical construct which may be used by syntaxes to construct a *Ordered List*. The exact usage depends on the syntax.

2.2.20 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:

- $\boxed{\textit{Service}}\ {*}\overset{\textit{providesLink}\ \blacktriangleright}{\rule{3cm}{0.4pt}}{*}\ \boxed{\textit{Link}}$
- $\boxed{\textit{Service}}\ {*}\overset{\textit{providesLink}\ \blacktriangleright}{\rule{3cm}{0.4pt}}{*}\ \boxed{\textit{LinkGroup}}$

Defined relations are:

- $\boxed{\textit{SwitchingService}}\ {1}\overset{\textit{providesLink}\ \blacktriangleright}{\rule{3cm}{0.4pt}}{*}\ \boxed{\textit{Link}}$
- $\boxed{\textit{SwitchingService}}\ {1}\overset{\textit{providesLink}\ \blacktriangleright}{\rule{3cm}{0.4pt}}{*}\ \boxed{\textit{LinkGroup}}$

2.2.21 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:

- $\boxed{\mathit{Service}}\,{}_{*}\underline{\;\mathit{providesPort}\ \blacktriangleright\;}{}_{*}\,\boxed{\mathit{Port}}$
- $\boxed{\mathit{Service}}\,{}_{*}\underline{\;\mathit{providesPort}\ \blacktriangleright\;}{}_{*}\,\boxed{\mathit{PortGroup}}$

Defined relations are:

- $\boxed{\mathit{AdaptationService}}\,{}_{1}\underline{\;\mathit{providesPort}\ \blacktriangleright\;}{}_{*}\,\boxed{\mathit{Port}}$
- $\boxed{\mathit{AdaptationService}}\,{}_{1}\underline{\;\mathit{providesPort}\ \blacktriangleright\;}{}_{*}\,\boxed{\mathit{PortGroup}}$
- $\boxed{\mathit{DeadaptationService}}\,{}_{1}\underline{\;\mathit{providesPort}\ \blacktriangleright\;}{}_{*}\,\boxed{\mathit{Port}}$
- $\boxed{\mathit{DeadaptationService}}\,{}_{1}\underline{\;\mathit{providesPort}\ \blacktriangleright\;}{}_{*}\,\boxed{\mathit{PortGroup}}$

2.3 Attributes

Attributes are properties of an object. The following attributes have been defined in section 2.1.

Attribute	Class (section)
adaptationFunction	AdaptationService (2.1.7), DeadaptationService (2.1.8)
address	Location (2.1.15)
alt	Location (2.1.15)
encoding	Port (2.1.3), Link (2.1.4), PortGroup (2.1.11), LinkGroup (2.1.12), BidirectionalPort (2.1.14), BidirectionalLink (2.1.14),SwitchingService (2.1.6)
end	LifeTime (2.1.16)
id	NetworkObject (2.1.1), Location (2.1.15)
labeltype	Label (2.1.17), LabelGroup (2.1.18)
lat	Location (2.1.15)
long	Location (2.1.15)
name	NetworkObject, Location (2.1.15)
start	LifeTime (2.1.16)
unlocode	Location (2.1.15)
value	Label (2.1.17)
values	LabelGroup (2.1.18)
version	NetworkObject (2.1.1)

2.4 Parameters

Parameters are properties of an object. Parameters, like attributes, are properties of objects, but may (subtly) change the logic of the object. The following parameters have been defined in section 2.1.

Parameter	Class (section)
labelSwapping	SwitchingService (2.1.6)
noReturnTraffic	Link (2.1.4)

3 Identifiers

3.1 Schema Identifier

The namespace for the schema defined in document is `http://schemas.ogf.org/nml/2013/05/base\#`.

All classes, relations, parameters and attributes defined in this document reside in this namespace. For example, the Link class is identified by `http://schemas.ogf.org/nml/2013/05/base#Link`

3.2 Instance Identifiers

Section 2.1.1 requires that instances of Network Objects SHOULD 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 [GFD.202], which ensures global uniqueness and easy recognition as Network Object instances.

Two different Network Objects instances 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[2] [Unicode].

Other interpretation (such as percent-decoding or Punycode decoding [RFC 3492]) MUST NOT 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:

[2] *Case folding* is primarily used for caseless comparison of text. *Case mapping* is used for display purposes.

```
1 - urn:ogf:network:example.net:2013:local_string_1234
2 - URN:OGF:network:EXAMPLE.NET:2013:Local_String_1234
```

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

```
1 - urn:ogf:network:example.net:2013:local_string_1234
3 - urn:ogf:network:example.net:2013: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 completely 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. An appropriate mechanism is to send a new description of the Topology or the Network Object with an updated *version* attribute.

3.3 Unnamed Objects

Network Objects that do not have a regular URI as id attribute, may have either:

- Have no id attribute. These are so-called *unnamed* network objects.
- Have an id attribute which is a fragment identifier only, thus an URI starting with a crosshatch (#) character. These are so-called *ad-hoc named* network objects.

A *unnamed* network object can not be referenced. A network objects generally SHOULD NOT be unnamed, since there is no possibility for an external party to refer to the object.

A *ad-hoc named* network object can only be referenced from within the same topology description. *ad-hoc id*s must be considered a syntactical construct, not as a persistent identifier. The MUST NOT be referred to from another scope or another topology description. *ad-hoc id*s SHOULD NOT be stored. If a two peers exchange topology messages, it is perfectly valid to change the ad-hoc id in each message (since they are only valid within scope of that message anyway).

A possible reason to use unnamed or ad-hoc named network objects it to make a statement such as *"Port A and Port B are grouped in a BidirectionalPort"* without actually assigning an identifier to this BidirectionalPort.

4 Syntax

The Network Markup Language has two different normative syntaxes. The syntaxes are in regular XML defined using an XML Schema (XSD), 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. These syntaxes follow the model as defined in section 2, should there be any inconsistencies between the syntaxes or the syntaxes and the model, the definitions in section 2 take precedence.

4.1 XML Syntax

An NML object is represented as an XML element. For example:

```
<nml:BidirectionalLink />
```

An NML attribute or parameter is represented as either an XML attribute, XML child element or text value of the XML element. The table list the mapping for the attributes and parameters defined in this document:

XML representation	**Attribute**	**Child element**	**Text of element**
NML attribute or parameter	id	name	value
	adaptationFunction	address	values
	encoding	lat	
	labeltype	long	
	version	alt	
	noReturnTraffic	unlocode	
	labelSwapping	start	
		end	

For example:

```
<nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:in">
  <nml:name>VLAN 1501 at Port X (in)</nml:name>
  <nml:Label labeltype="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">1501</nml:Label>
</nml:Port>
```

Explicit relation are represented as a <nml:Relation> XML element, with the domain as the parent element, and the range as the child element. Implicit relations are not given: the range object is represented as an XML child element of the domain. Below is an example of an explicit relation:

```
<nml:Port id="urn:ogf:network:example.net:2013:port_X:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isSource">
    <nml:Link id="urn:ogf:network:example.net:2013:linkA:XY"/>
  </nml:Relation>
</nml:Port>
```

And this is an example of an implicit relation:

```
<nml:Topology id="urn:ogf:network:example.net:2013:Example_Network">
  <nml:Node id="urn:ogf:network:example.net:2013:example_node"/>
</nml:Port>
```

4.2 OWL RDF/XML Syntax

An NML object is represented as an RDF subject. Exceptions are *Label* and *LabelGroup*.

An NML attribute or parameter is represented as a predicate.

For example:

```
<nml:Location id="urn:ogf:network:example.net:2013:redcity">
  <nml:name>Red City</nml:name>
  <nml:lat>30.600</nml:lat>
  <nml:long>12.640</nml:long>
</nml:Location>
```

Relations are represented as an RDF triplet, with the full URI of the attribute or parameter. For example:

```
<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X:in">
   <nml:isSource rdf:resource="urn:ogf:network:example.net:2013:linkA:XY"/>
</nml:Port>
```

A *Label* is represented as a two triplets: one triplet defining a labeltype as a subproperty of the abstract Label resource, and one relating a Port or Link to a value using this labeltype. For example:

```
<rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">
  <owl:subPropertyOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#hasLabel"/>
</rdf:Description>
<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X.1501:in">
  <nmleth:vlan>1501</nmleth:vlan>
</nml:Port>
```

A *LabelGroup* is represented as three triplets. The URI of the labeltype is not the URI of the predicate, to avoid naming clashes with the definition of the Label. Instead, the predicate of the LabelGroup is related to the predicate of the Label using the nml:labeltype property:

```
<rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/ethernet#vlans">
  <nml:labelType rdf:resource="http://schemas.ogf.org/nml/2013/05/ethernet#vlan"/>
  <owl:subPropertyOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#hasLabelGroup"/>
</rdf:Description>
<nml:PortGroup rdf:about="urn:ogf:network:example.net:2013:portgroup_X:out">
  <nmleth:vlans>1480-1530</nmleth:vlans>
</nml:PortGroup>
```

4.3 Combining Object Descriptions

A given object may have multiple attributes and relations. These attributes and relations may be described in different places in a syntax. It is up to the parser to combine all attributes and relations.

NML currently does not have a mechanism to check if a given description of an object is *complete*. Thus, it does not distinguish between a full description of an object or merely a pointer to an object.

Parsers should be aware that the NML descriptions do not provide any guarantee regarding the integrity nor the authenticity of the description. Parsers are advised to use external mechanism to avoid that an erroneous description of an object in one (possibly malicious) topology description pollutes a correct description of the same object in another topology description.

4.4 Ordered Lists

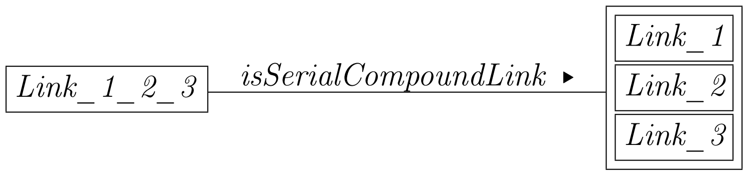
The range of an *isSerialCompoundLink* relation is an *Ordered List*.

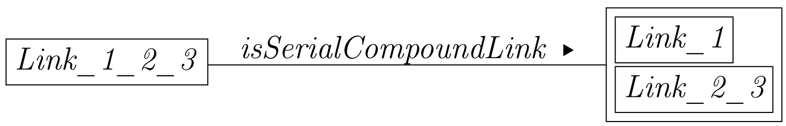
Neither XML nor OWL uses the *Ordered List* directly in the syntax, and have a different way of constructing ordered lists. XML lists values with additional *next* relations, while OWL uses a *ListItem* class, and the *item* and *next* relations.

A *ListItem* behaves as a class, while *item* and *next* behave like relations, with the exception that these classes and relations are local in scope.

This means that these relations are only valid within the scope of a given *Ordered List*, but may not be valid in scope of a different *Ordered List*. It also means that any identifier given to these classes may change when the objects are codified in a syntax.

For example, consider the following two decompositions of *Link* `Link_1_2_3` into shorter *Links*:

- 

- 

In the first *Ordered List*, there is a *next* relation from `Link_1` to `Link_2`, while in the second Ordered List, the *next* relation is from `Link_1` to `Link_2_3`.

In XML an *Ordered List* can be constructed by using all objects in the list as child elements, and using a *next* relation between consecutive objects in the list to denote ordering.

In OWL an *Ordered List* can be constructed by creating as many *ListItem* objects as there are items in the list. Each *ListItem* object is correlated with the actual list item using the *item* relation, while using a *next* relation to point to the following *ListItem*. A predicate points to the first *ListItem* in the *Ordered List* to point to the whole list, which is chained using the *next* relation.

See also the *isSerialCompoundLink* examples in the example section.

5 Examples

A graphical overview showing the combination of all the examples is shown in figure 3.

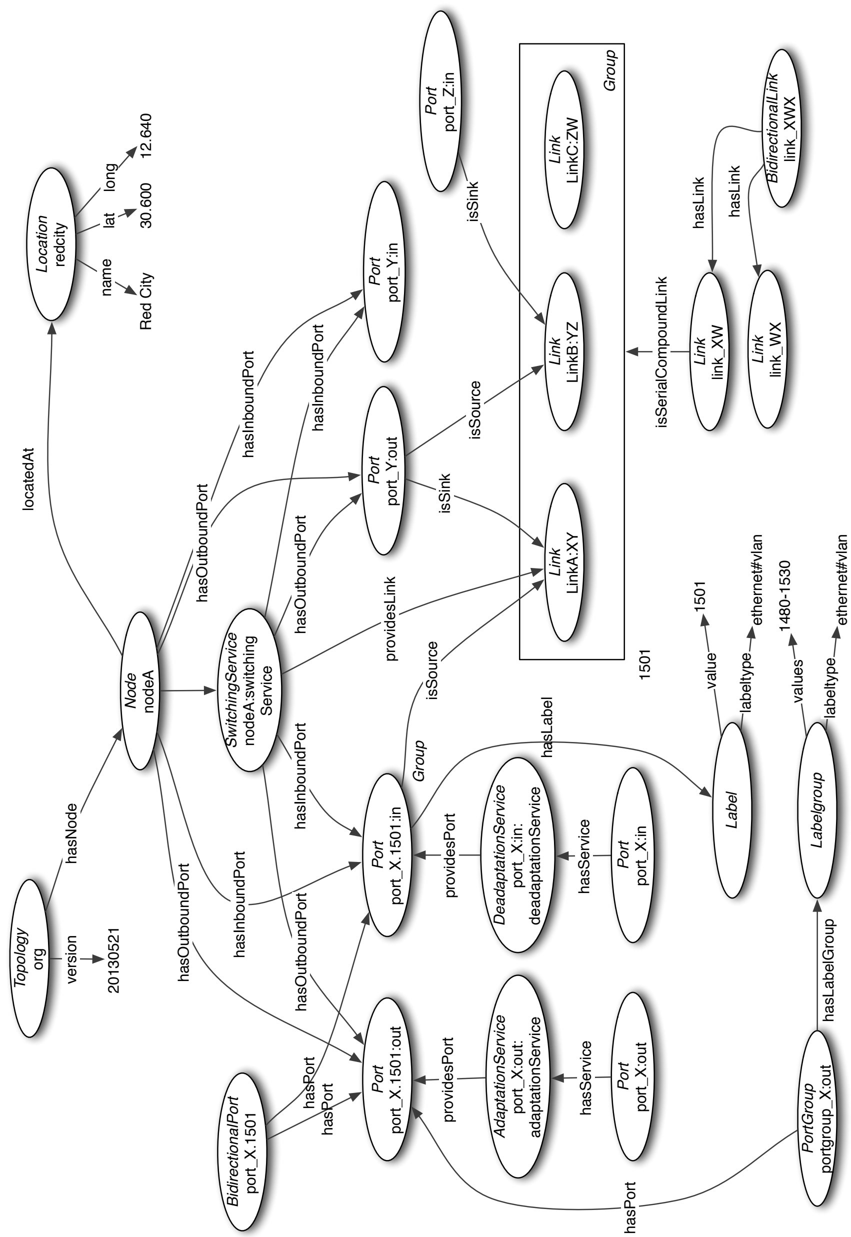


Figure 3: A graphical overview showing the combination of all examples

5.1 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/2013/05/base#"
              id="urn:ogf:network:example.net:2013:org"
              version="20130529T121112Z">
  <nml:Node id="urn:ogf:network:example.net:2013:nodeA">

  <!-- ... -->

  </nml:Node>

</nml:Topology>
```

- *Node* (section 2.1.2)

```
<nml:Node id="urn:ogf:network:example.net:2013:nodeA">
  <nml:name>Node_A</nml:name>
  <nml:Location id="urn:ogf:network:example.net:2013:redcity"/>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
    <nml:Port id="urn:ogf:network:example.net:2013:port_X:out"/>
    <nml:Port id="urn:ogf:network:example.net:2013:port_Y:out"/>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
    <nml:Port id="urn:ogf:network:example.net:2013:port_X:in"/>
    <nml:Port id="urn:ogf:network:example.net:2013:port_Y:in"/>
  </nml:Relation>
</nml:Node>
```

- *Ports*
  - *(Unidirectional) Port* (section 2.1.3)

```
<nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:in">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">1501</nml:Label>
</nml:Port>
```

  - *BidirectionalPort* (section 2.1.13)

```
<nml:BidirectionalPort id="urn:ogf:network:example.net:2013:port_X.1501">
  <nml:name>X.1501</nml:name>
  <nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:out"/>
  <nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:in"/>
</nml:BidirectionalPort>
```

– *PortGroup* (section 2.1.11)

```
<nml:PortGroup id="urn:ogf:network:example.net:2013:portgroup_X:out">
  <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">
    1480-1530
  </nml:LabelGroup>
  <nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:out"/>
</nml:PortGroup>
```

- *Links*

  – *UnidirectionalLink* (external) (section 2.1.4)

```
<nml:Link id="urn:ogf:network:example.net:2013:linkB:YZ"/>

<nml:Port id="urn:ogf:network:example.net:2013:port_Y:out">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isSource">
    <nml:Link id="urn:ogf:network:example.net:2013:linkB:YZ"/>
  </nml:Relation>

  <nml:Port id="urn:ogf:network:example.net:2013:port_Z:in">
    <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isSink">
      <nml:Link id="␣urn:ogf:network:example.net:2013:linkB:YZ"/>
    </nml:Relation>
  </nml:Port>

</nml:Port>
```

  – *UnidirectionalLink* (internal) (section 2.1.4)

```
<nml:Link id="urn:ogf:network:example.net:2013:linkA:XY"/>

<nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isSource">
    <nml:Link id="␣urn:ogf:network:example.net:2013:linkA:XY"/>
  </nml:Relation>
</nml:Port>

<nml:Port id="urn:ogf:network:example.net:2013:port_Y:out">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isSink">
    <nml:Link id="urn:ogf:network:example.net:2013:linkA:XY"/>
  </nml:Relation>
</nml:Port>
```

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

```
<nml:Link id="urn:ogf:network:example.net:2013:link_XW">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#isSerialCompoundLink">
    <nml:Link id="urn:ogf:network:example.net:2013:linkA:XY">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#next">
        <nml:Link id="urn:ogf:network:example.net:2013:linkB:YZ"/>
```

```
      </nml:Relation>
    </nml:Link>
    <nml:Link id="urn:ogf:network:example.net:2013:linkB:YZ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#next">
        <nml:Link id="urn:ogf:network:example.net:2013:linkC:ZW"/>
      </nml:Relation>
    </nml:Link>
    <nml:Link id="urn:ogf:network:example.net:2013:linkC:ZW"/>
  </nml:Relation>
</nml:Link>
```

- *BidirectionalLink* (section 2.1.14)

```
<nml:BidirectionalLink id="urn:ogf:network:example.net:2013:link_XWX">
  <nml:name>Link between ports X and W</nml:name>
  <nml:Link id="urn:ogf:network:example.net:2013:link_XW"/>
  <nml:Link id="urn:ogf:network:example.net:2013:link_WX"/>
</nml:BidirectionalLink>
```

- *LinkGroup* (section 2.1.12)

```
<nml:LinkGroup id="urn:ogf:network:example.net:2013:domainy_domainx">
  <nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">
    1480-1530
  </nml:LabelGroup>
</nml:LinkGroup>
```

- *Labels*
  - *Label* (section 2.1.17)

```
<nml:Label labeltype="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">1501</nml:Label>
```

  - *LabelGroup* (section 2.1.18)

```
<nml:LabelGroup labeltype="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">
  1480-1530
</nml:LabelGroup>
```

- *Location* (section 2.1.15)

```
<nml:Location id="urn:ogf:network:example.net:2013:redcity">
  <nml:name>Red City</nml:name>
  <nml:lat>30.600</nml:lat>
  <nml:long>12.640</nml:long>
</nml:Location>
```

- *Services*
  - *SwitchingService* (section 2.1.6)

```
<nml:Node id="urn:ogf:network:example.net:2013:nodeA">
  <nml:name>Node_A</nml:name>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
    <nml:Port id="urn:ogf:network:example.net:2013:nodeA:port_X:in" />
    <nml:Port id="urn:ogf:network:example.net:2013:nodeA:port_Y:in" />
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
    <nml:Port id="urn:ogf:network:example.net:2013:nodeA:port_X:out"/>
    <nml:Port id="urn:ogf:network:example.net:2013:nodeA:port_Y:out"/>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasService">
    <nml:SwitchingService id="urn:ogf:network:example.net:2013:nodeA:switchingService"/>
  </nml:Relation>
</nml:Node>

<nml:SwitchingService id="urn:ogf:network:example.net:2013:nodeA:switchingService">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
    <nml:Port id="urn:ogf:network:example.net:2013:nodeA:port_X:in" />
    <nml:Port id="urn:ogf:network:example.net:2013:nodeA:port_Y:in" />
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
    <nml:Port id="urn:ogf:network:example.net:2013:nodeA:port_X:out"/>
    <nml:Port id="urn:ogf:network:example.net:2013:nodeA:port_Y:out"/>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#providesLink">
      <nml:Link id="urn:ogf:network:example.net:2013:LinkA:XY"/>
  </nml:Relation>
</nml:SwitchingService>
```

  - *AdaptationService* (section 2.1.7)

```
<nml:Port id="urn:ogf:network:example.net:2013:port_X:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasService">
    <nml:AdaptationService id="urn:ogf:network:example.net:2013:port_X:in:adaptationService"/>
  </nml:Relation>
</nml:Port>

<nml:AdaptationService
     id="urn:ogf:network:example.net:2013:port_X:in:adaptationService"
     adaptationFunction="http://schemas.ogf.org/nml/2013/05/ethernet#802.1q">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#providesPort">
      <nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:in"/>
  </nml:Relation>
</nml:AdaptationService>

<nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:in">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">1501</nml:Label>
</nml:Port>
```

  - *DeadaptationService* (section 2.1.8)

```
<nml:Port id="urn:ogf:network:example.net:2013:port_X:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasService">
    <nml:DeadaptationService id="urn:ogf:network:example.net:2013:port_X.1501:in:deadaptationService"/>
  </nml:Relation>
</nml:Port>

<nml:DeadaptationService
     id="urn:ogf:network:example.net:2013:port_X.1501:in:deadaptationService"
     adaptationFunction="http://schemas.ogf.org/nml/2013/05/ethernet#802.1q">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/05/base#providesPort">
      <nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:in"/>
  </nml:Relation>
</nml:DeadaptationService>

<nml:Port id="urn:ogf:network:example.net:2013:port_X.1501:in">
    <nml:Label labeltype="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">1501</nml:Label>
</nml:Port>
```

5.2 Examples in OWL

The following snippets represent NML structures in the OWL format. The namespaces used in all the examples follow the definitions of the Topology example.

- *Topology* (section 2.1.10)

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:nml="http://schemas.ogf.org/nml/2013/05/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/2013/05/ethernet#"
>
<nml:Topology rdf:about="urn:ogf:network:example.net:2013:org">
    <nml:version>20130529T121112Z</nml:version>
    <nml:hasNode rdf:resource="urn:ogf:network:example.net:2013:nodeA"/>
  <!-- ... -->

</nml:Topology>
```

- *Node* (section 2.1.2)

```
<nml:Node rdf:about="urn:ogf:network:example.net:2013:nodeA">
  <nml:name>Node_A</nml:name>
  <nml:locatedAt rdf:resource="urn:ogf:network:example.net:2013:redcity"/>
  <nml:hasOutboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_X:out"/>
  <nml:hasOutboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_Y:out"/>
  <nml:hasInboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_X:in"/>
  <nml:hasInboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_Y:in"/>
</nml:Node>
```

- *Ports*
  - *(Unidirectional) Port* (section 2.1.3)

```
<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X.1501:in">
    <nmleth:vlan>1501</nmleth:vlan>
</nml:Port>
```

  - *BidirectionalPort* (section 2.1.13)

```
<nml:BidirectionalPort rdf:about="urn:ogf:network:example.net:2013:port_X.1501">
  <nml:name>X.1501</nml:name>
  <nml:hasPort rdf:resource="urn:ogf:network:example.net:2013:port_X.1501:out"/>
  <nml:hasPort rdf:resource="urn:ogf:network:example.net:2013:port_X.1501:in"/>
</nml:BidirectionalPort>
```

  - *PortGroup* (section 2.1.11)

```
<nml:PortGroup rdf:about="urn:ogf:network:example.net:2013:portgroup_X:out">
    <nmleth:vlans>1480-1530</nmleth:vlans>
    <nml:hasPort>
      <nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X.1501:out"/>
    </nml:hasPort>
</nml:PortGroup>
```

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

```
<nml:Link rdf:about="urn:ogf:network:example.net:2013:linkB:YZ"/>

<nml:Port id="urn:ogf:network:example.net:2013:port_Y:out">
  <nml:isSink rdf:resource="urn:ogf:network:example.net:2013:linkB:YZ"/>
</nml:Port>

<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_Z:in">
  <nml:isSource rdf:resource="urn:ogf:network:example.net:2013:linkB:YZ"/>
</nml:Port>
```

  - *UnidirectionalLink* (internal) (section 2.1.4)

```
<nml:Link rdf:about="urn:ogf:network:example.net:2013:linkA:XY"/>

<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X.1501:in">
  <nml:isSource rdf:resource="urn:ogf:network:example.net:2013:linkA:XY"/>
</nml:Port>

<nml:Port id="urn:ogf:network:example.net:2013:port_Y:out">
  <nml:isSink rdf:resource="urn:ogf:network:example.net:2013:linkA:XY"/>
</nml:Port>
```

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

```
<nml:Link rdf:about="urn:ogf:network:example.net:2013:link_XW">
  <nml:isSerialCompoundLink>
    <nml:ListItem rdf:resource="urn:ogf:network:example.net:2013:link_XW_1">
      <nml:item rdf:resource="urn:ogf:network:example.net:2013:linkA:XY"/>
      <nml:next rdf:resource="urn:ogf:network:example.net:2013:link_XW_2"/>
    </nml:ListItem>
  </nml:isSerialCompoundLink>
</nml:Link>

<nml:ListItem rdf:resource="urn:ogf:network:example.net:2013:link_XW_2">
  <nml:item rdf:resource="urn:ogf:network:example.net:2013:linkB:YZ"/>
  <nml:next rdf:resource="urn:ogf:network:example.net:2013:link_XW_3"/>
</nml:ListItem>

```

```
<nml:ListItem rdf:resource="urn:ogf:network:example.net:2013:link_XW_3">
  <nml:item rdf:resource="urn:ogf:network:example.net:2013:linkC:ZW"/>
</nml:ListItem>
```

– *BidirectionalLink* (section 2.1.14)

```
<nml:BidirectionalLink rdf:about="urn:ogf:network:example.net:2013:link_XWX">
  <nml:name>Link between ports X and W</nml:name>
  <nml:hasLink rdf:resource="urn:ogf:network:example.net:2013:link_XW"/>
  <nml:hasLink rdf:resource="urn:ogf:network:example.net:2013:link_WX"/>
</nml:BidirectionalLink>
```

– *LinkGroup* (section 2.1.12)

```
<nml:LinkGroup rdf:about="urn:ogf:network:example.net:2013:domainy_domainx">
  <nmleth:vlans>1480−1530</nmleth:vlans>
</nml:LinkGroup>
```

- *Labels*
  – *Label* (section 2.1.17)

```
<rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/ethernet#vlan">
  <owl:subPropertyOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#hasLabel"/>
</rdf:Description>
<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X.1501:in">
  <nmleth:vlan>1501</nmleth:vlan>
</nml:Port>
```

  – *LabelGroup* (section 2.1.18)

```
<rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/ethernet#vlans">
  <nml:labelType rdf:resource="http://schemas.ogf.org/nml/2013/05/ethernet#vlan"/>
  <owl:subPropertyOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#hasLabelGroup"/>
</rdf:Description>
<nml:PortGroup rdf:about="urn:ogf:network:example.net:2013:portgroup_X:out">
  <nmleth:vlans>1480−1530</nmleth:vlans>
</nml:PortGroup>
```

- *Location* (section 2.1.15)

```
<nml:Location id="urn:ogf:network:example.net:2013: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:2013:nodeA">
  <nml:name>Node_A</nml:name>
  <nml:hasInboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_X:in"/>
  <nml:hasInboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_Y:in"/>
  <nml:hasOutboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_X:out"/>
  <nml:hasOutboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_Y:out"/>
  <nml:hasService rdf:resource="urn:ogf:network:example.net:2013:nodeA:switchingService"/>
</nml:Node>

<nml:SwitchingService rdf:about="urn:ogf:network:example.net:2013:nodeA:switchingService">
  <nml:hasInboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_X:in"/>
  <nml:hasInboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_Y:in"/>
  <nml:hasOutboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_X:out"/>
  <nml:hasOutboundPort rdf:resource="urn:ogf:network:example.net:2013:nodeA:port_Y:out"/>
</nml:SwitchingService>
```

  - *AdaptationService* (section 2.1.7)

```
<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X:out">
  <nml:hasService>
      <nml:AdaptationService
           rdf:about="urn:ogf:network:example.net:2013:port_X:out:adaptationService">
        <nml:adaptationFunction rdf:resource="http://schemas.ogf.org/nml/2013/05/ethernet#802.1q"/>
        <nml:providesPort rdf:resource="urn:ogf:network:example.net:2013:port_X.1501:out"/>
      </nml:AdaptationService>
  </nml:hasService>
</nml:Port>

<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X.1501:out">
  <nmleth:vlan>1501</nmleth:vlan>
</nml:Port>
```

  - *DeadaptationService* (section 2.1.8)

```
<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X:in">
  <nml:hasService>
    <nml:DeadaptationService
      rdf:resource="urn:ogf:network:example.net:2013:port_X:in:deadaptationService">
      <nml:adaptationFunction rdf:resource="http://schemas.ogf.org/nml/2013/05/ethernet#802.1q"/>
      <nml:providesPort rdf:resource="urn:ogf:network:example.net:2013:port_X.1501:in"/>
    </nml:DeadaptationService>
  </nml:hasService>
</nml:Port>

<nml:Port rdf:about="urn:ogf:network:example.net:2013:port_X.1501:in">
    <nmleth:vlan>1501</nmleth:vlan>
</nml:Port>
```

5.3 Conceptual Examples

This section shows a few examples how NML was designed to be used. Like the other examples, this section is informative. It may be possible that there are other ways to use the NML objects and attributes.

5.3.1 Topology and Node

A *Topology* and *Node* behave similar: they both contain inbound ports and outbound ports, and can contain a *SwitchingService* to allow creation of internal links (cross connects) from inbound ports to outbound ports. Especially with the ability to create logical, sliced or virtual devices, the distinction is getting blurred.

The distinction is that a *Node* is located at a single geographic location, while a *Topology* is a set of geographically disperse Network Objects.

5.3.2 Hierarchical Topology

Large networks may want to publish both details of their network topology as a whole, as well as details about regional segments, without publishing details of the actual devices. NML allows the publication of a hierarchical *Topology* tree, where the top-level *Topology* has a *hasTopology* relation with smaller *Topologies*. These smaller Topologies must be fully enclosed – the *hasTopology* relation can not be used to relate partial overlapping Topologies.

For example, a *Topology* `A` may want to publish about two parts of its Topology, `A_West`, and `A_East`. This allows it to publish difference in connectivity and costs between the two parts. It can do so with the following relations:

Topology `A` — *hasTopology* ▸ — *Topology* `A_West`

Topology `A` — *hasTopology* ▸ — *Topology* `A_East`

5.3.3 Links, Segments and Paths

A *Link* object can refer to any link connection. A link segment and an end-to-end path are both described by a *Link* object. This is by design, since it is easy to extend a *Link*, or to describe a partition of a *Link*.

Figure 4 gives an example of three different partitionings of a link between `port_X:in` and `port_W:out`.

Note that in this example Port `port_Y:out` is the source of both `linkB:YZ` and of `linkBC:YW`. If a single topology description would contain the full link and the partitioning, a path finding

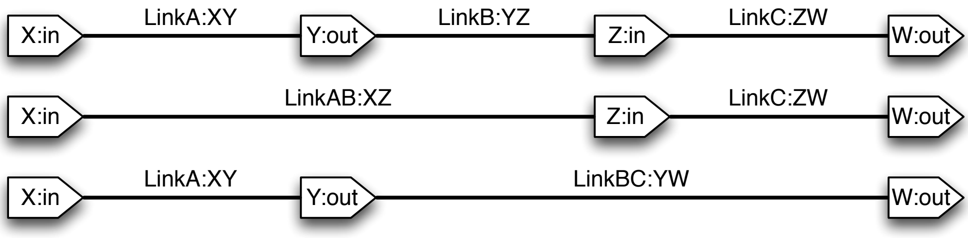


Figure 4: Different partitionings of the same link.

algoritm MUST be aware that the fact that if a Port is the source of two NML Links, this does not mean it multicast to different network links. For this reason, it is RECOMMENDED that applications either add metadata about the type of link, or specify that in certain messages, only one particular type of Link MUST be used.

5.3.4 Patch Panel and Media Convertor

A port on a patch panel or optical distribution frame can simply be described as a NML Port (thus without an associated Node):

Port `odf_X` — *isSink* ▸ — *Link* `A`
Port `odf_X` — *isSource* ▸ — *Link* `B`

A mediaconvertor, e.g. from Ethernet over UTP to Ethernet over fiber, can be described in the same way, provided that the connected Links described the Ethernet connections. If the connected Links describe the underlying UTP and fiber connections, it is necessary to describe the conversion between them:

Port `Port_X_UTP` — *isSink* ▸ — *Link* `UTP_A`
Port `Port_X_fiber` — *isSource* ▸ — *Link* `fiber_B`
Port `Port_X_UTP` — *hasService* ▸ — *DeAdaptataionService* `X_deadaptation`
Port `Port_X_fiber` — *hasService* ▸ — *AdaptationService* `X_adaptation`
DeAdaptataionService `X_deadaptation` — *providesPort* ▸ — *Port* `Port_X_eth`
AdaptationService `X_adaptation` — *providesPort* ▸ — *Port* `Port_X_eth`

5.3.5 VLAN and Broadcast Medium

A VLAN is much like a broadcast medium, which can be described as a multipoint-to-multipoint Link:

Port Port_X:in	isSink ▸	Link VLAN_42
Port Port_X:out	isSink ▸	Link VLAN_42
Port Port_Y:in	isSink ▸	Link VLAN_42
Port Port_Y:out	isSink ▸	Link VLAN_42
Port Port_Z:in	isSink ▸	Link VLAN_42
Port Port_Z:out	isSink ▸	Link VLAN_42

Where X, Y and Z are in fact bidirectional ports:

BidirectionalPort Port_X	hasPort ▸	Port Port_X:in
BidirectionalPort Port_X	hasPort ▸	Port Port_X:out
BidirectionalPort Port_Y	hasPort ▸	Port Port_Y:in
BidirectionalPort Port_Y	hasPort ▸	Port Port_Y:out
BidirectionalPort Port_Z	hasPort ▸	Port Port_Z:in
BidirectionalPort Port_Z	hasPort ▸	Port Port_Z:out

However, this is not entirely correct: in the above description data coming from `Port_X:in` would also be forwarded to `Port_X:out`. However, the Ethernet technology prevents data returning on the same interface.

NML introduced the *noReturnTraffic* parameter to describe this technological restriction: if the *noReturnTraffic* parameter of a Link is true, there is no data transport from a source to a sink if the source and sink are grouped together in a BidirectionalPort group.

Link VLAN_42	noReturnTraffic ▸	"true"

5.3.6 Configuration and Potential Capability

NML is able to both describe the network services (potential capability) as well as the network configuration.

A switching service can be described by a SwitchingService object along with associated inbound ports and outbound ports:

SwitchingService switchmatrix_A	hasInboundPort ▸	PortGroup Port_X:in
SwitchingService switchmatrix_A	hasOutboundPort ▸	PortGroup Port_X:out

SwitchingService `switchmatrix_A` — *hasInboundPort* ▸ — *PortGroup* `Port_Y:in`
SwitchingService `switchmatrix_A` — *hasOutboundPort* ▸ — *PortGroup* `Port_Y:out`
SwitchingService `switchmatrix_A` — *labelSwapping* ▸ — `true`

A cross connect created by this switching service can be specified by a Link object:

SwitchingService `switchmatrix_A` — *providesLink* ▸ — *Link* `crossconnect_A.1501`
PortGroup `Port_X:in` — *hasPort* ▸ — *Port* `Port_X.1501:in`
PortGroup `Port_Y:out` — *hasPort* ▸ — *Port* `Port_Y.1501:out`
Port `Port_X.1501:in` — *isSource* ▸ — *Link* `crossconnect_A.1501`
Port `Port_Y.1501:out` — *isSink* ▸ — *Link* `crossconnect_A.1501`

An encoding and decoding service can be described by a AdapdationService and DeAdaptationService:

Port `Port_X_fiber:in` — *hasService* ▸ — *DeAdaptationService* `port_X:in:deadaptation`

DeAdaptationService `port_X:in:deadaptation` — *canProvidePort* ▸ — *PortGroup* `Port_X:in`

A channel created by this encoding service can be specified by a *providesPort* relation:

DeAdaptationService `port_X:in:deadaptation` — *providesPort* ▸ — *Port* `Port_X.1501:in`

PortGroup `Port_X:in` — *hasPort* ▸ — *Port* `Port_X.1501:in`

5.3.7 Versioning and Lifetime

The version of a *Topology* indicated the serial number. If there are two *Topology* descriptions for the same network, the one with the highest version number is the most recent version. The *LifeTime* object is used to indicate when a certain resource is available.

Imagine that a link will have a scheduled downtime due to maintenance next week between 2 AM and 4 AM. This can be specified with these relations:

Topology `org` — *version* ▸ — `20130521T000000Z`
Link `A` — *existsDuring* ▸ — *LifeTime* `A_lifetime1`
Link `A` — *existsDuring* ▸ — *LifeTime* `A_lifetime2`
LifeTime `A_lifetime1` — *end* ▸ — `20130611T020000Z`
LifeTime `A_lifetime2` — *start* ▸ — `20130611T040000Z`

Imagine that this planned maintainance is rescheduled. That can be specified by creating a new Topology with a new version number, and updated data:

Topology `org` — *version* ▸ — `20130604T000000Z`
Link `A` — *existsDuring* ▸ — *LifeTime* `A_lifetime1`
Link `A` — *existsDuring* ▸ — *LifeTime* `A_lifetime2`
LifeTime `A_lifetime1` — *end* ▸ — `20130618T020000Z`
LifeTime `A_lifetime2` — *start* ▸ — `20130618T040000Z`

6 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 confidential. We do not address these concerns in this document, but implementers are encouraged to consider the security implications of generating and distributing network topology information.

Implementers should be aware that the NML descriptions do not provide any guarantee regarding their integrity nor their authenticity. The NML documents also 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.

7 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
SURFsara
Science Park 140, 1098 XG Amsterdam
The Netherlands
Email: freek.dijkstra@surfsara.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@es.net

8 Acknowledgments

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

Financial support has been provided by several projects and institutions:

This work was partially supported by the European Commission, 7^{th} Framework Programme for Research and Technological Development, Capacities, The GN3 project – Grant No. 238875, GEYSERS – Grant No. 248657 and NOVI – Grant No. 257867. This project was also made possible by the support of SURF, the collaborative organisation for higher education institutes and research institutes aimed at breakthrough innovations in ICT. More information on SURF is available on the website www.surf.nl. Furthermore, this work was supported by the Dutch national program COMMIT.

Jason Zurawski would like to thank Internet2, along with the National Science Foundation (NSF) for support through grants: #0962704, #1019008, and #0721902.

The authors are indebted to the many participants of working group sessions and on the mailing list, including the following contributors: Aaron Brown, Jeff W. Boote, Aurélien Cedeyn, Evangelos Chaniotakis, Chin Guok, Paola Grosso, Richard Hughes-Jones, Houssem Medhioub, Ralph Niederberger, Anand Patil, Victor Reijs, Guy Roberts, Jerry Sobieski, Martin Swany, Fausto Vetter, Pascale Vicat-Blanc, and John Vollbrecht.

9 Intellectual Property Statement

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

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

10 Disclaimer

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

11 Full Copyright Notice

Appendix A XML Schema

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

```xml
<?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/2013/05/base#"
           xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#"
           elementFormDefault="qualified">


  <xs:complexType name="NetworkObject">
    <xs:sequence>
      <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="Lifetime" type="nml:LifeTimeType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="Location" type="nml:LocationType" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:anyURI" use="optional"/>
    <xs:attribute name="version" type="xs:dateTime" use="optional"/>
  </xs:complexType>


  <xs:complexType name="LocationType">
    <xs:all>
      <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <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"/>
      <!-- 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" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:all>
    <xs:attribute name="id" type="xs:anyURI" use="optional"/>
  </xs:complexType>


  <xs:complexType name="LifeTimeType">
    <xs:sequence>
      <xs:element name="start" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
```

```
      <xs:element name="end" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>


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


  <!-- Topology -->


  <xs:complexType name="TopologyRelationType">
    <xs:choice>
      <xs:element ref="nml:Port" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:PortGroup" minOccurs="1" maxOccurs="unbounded"/>
      <xs:group ref="nml:Service" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:Topology" minOccurs="1" maxOccurs="unbounded"/>
    </xs:choice>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#hasService"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isAlias"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>


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


  <xs:complexType name="TopologyType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:group ref="nml:BaseTopologyContent"/>
          <xs:element name="Relation" type="nml:TopologyRelationType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
```

```
</xs:complexType>


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


<!-- Link -->


<xs:complexType name="LinkRelationType">
  <xs:sequence>
    <xs:element ref="nml:Link" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isAlias"/>
        <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isSerialCompoundLink"/>
        <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#next"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>


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


<xs:complexType name="LinkType">
  <xs:complexContent>
    <xs:extension base="nml:NetworkObject">
      <xs:sequence>
        <xs:group ref="nml:BaseLinkContent"/>
        <xs:element name="Relation" type="nml:LinkRelationType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="encoding" type="xs:anyURI" use="optional"/>
      <xs:attribute name="noReturnTraffic" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>


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


<!-- Port -->


<xs:complexType name="PortRelationType">
  <xs:choice>
    <xs:element ref="nml:Link" minOccurs="1" maxOccurs="unbounded"/>
    <xs:element ref="nml:Port" minOccurs="1" maxOccurs="unbounded"/>
    <xs:group ref="nml:Service" minOccurs="1" maxOccurs="unbounded"/>
  </xs:choice>
  <xs:attribute name="type" use="required">
```

```
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#hasService"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isAlias"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isSink"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isSource"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>


  <xs:group name="BasePortContent">
    <xs:sequence>
      <xs:element ref="nml:Label" minOccurs="0" maxOccurs="1"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>


  <xs:complexType name="PortType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:group ref="nml:BasePortContent"/>
          <xs:element name="Relation" type="nml:PortRelationType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <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="NodeRelationType">
    <xs:choice>
      <xs:element ref="nml:Node" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:Port" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:PortGroup" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:SwitchingService" minOccurs="1" maxOccurs="unbounded"/>
    </xs:choice>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#hasService"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isAlias"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>


  <xs:complexType name="NodeType">
```

```
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:element ref="nml:Node" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="Relation" type="nml:NodeRelationType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </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>


  <!-- SwitchingService -->


  <xs:complexType name="SwitchingServiceRelationType">
    <xs:choice>
      <xs:element ref="nml:Port" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:PortGroup" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:SwitchingService" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:Link" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:LinkGroup" minOccurs="1" maxOccurs="unbounded"/>
    </xs:choice>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isAlias"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#providesLink"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>


  <xs:complexType name="SwitchingServiceType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:element name="Relation" type="nml:SwitchingServiceRelationType" minOccurs="0" maxOccurs="unbounded
               "/>
        </xs:sequence>
        <xs:attribute name="encoding" type="xs:anyURI" use="optional"/>
        <xs:attribute name="labelSwapping" type="xs:boolean" use="optional"/>
```

```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>


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


  <!-- AdaptationService -->


  <xs:complexType name="AdaptationServiceRelationType">
    <xs:choice>
      <xs:element ref="nml:Port" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:PortGroup" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:AdaptationService" minOccurs="1" maxOccurs="unbounded"/>
    </xs:choice>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#canProvidePort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isAlias"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#providesPort"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>


  <xs:complexType name="AdaptationServiceType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:element name="Relation" type="nml:AdaptationServiceRelationType" minOccurs="0" maxOccurs="
               unbounded"/>
        </xs:sequence>
        <xs:attribute name="adaptationFunction" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>


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


  <!-- DeadaptationService -->


  <xs:complexType name="DeadaptationServiceRelationType">
    <xs:choice>
      <xs:element ref="nml:Port" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:PortGroup" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:DeadaptationService" minOccurs="1" maxOccurs="unbounded"/>
    </xs:choice>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#canProvidePort"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isAlias"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#providesPort"/>
```

```
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>


  <xs:complexType name="DeadaptationServiceType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:element name="Relation" type="nml:DeadaptationServiceRelationType" minOccurs="0" maxOccurs="
               unbounded"/>
        </xs:sequence>
        <xs:attribute name="adaptationFunction" 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:complexType name="LinkGroupRelationType">
    <xs:sequence>
      <xs:element ref="nml:LinkGroup" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isAlias"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isSerialCompoundLink"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>


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

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


  <xs:complexType name="LinkGroupType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:group ref="nml:BaseLinkGroup"/>
          <xs:element name="Relation" type="nml:LinkGroupRelationType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>


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


  <!-- PortGroup -->


  <xs:complexType name="PortGroupRelationType">
    <xs:choice>
      <xs:element ref="nml:PortGroup" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="nml:LinkGroup" minOccurs="1" maxOccurs="unbounded"/>
    </xs:choice>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isAlias"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isSink"/>
          <xs:enumeration value="http://schemas.ogf.org/nml/2013/05/base#isSource"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>


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


  <xs:complexType name="PortGroupType">
    <xs:complexContent>
      <xs:extension base="nml:NetworkObject">
        <xs:sequence>
          <xs:group ref="nml:BasePortGroup"/>
          <xs:element name="Relation" type="nml:PortGroupRelationType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <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.

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://schemas.ogf.org/nml/2013/05/base#"
     xml:base="http://schemas.ogf.org/nml/2013/05/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/2013/05/base#">
    <owl:Ontology rdf:about="http://schemas.ogf.org/nml/2013/05/base#">
        <rdfs:label>NML Schema</rdfs:label>
    </owl:Ontology>


    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->


    <!-- http://schemas.ogf.org/nml/2013/05/base#adaptationFunction -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#adaptationFunction">
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#AdaptationService"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#DeadaptationService"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#address -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#address">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Location"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#canProvidePort -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#canProvidePort">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Service"/>
        <rdfs:range>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Port"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#PortGroup"/>
```

```
                </owl:unionOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#encoding -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#encoding">
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Link"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#LinkGroup"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Port"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#PortGroup"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#SwitchingService"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#existsDuring -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#existsDuring">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Lifetime"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#hasInboundPort -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
        <rdfs:range>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Port"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#PortGroup"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#hasLabel -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#hasLabel">
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Link"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Port"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Label"/>
    </owl:ObjectProperty>

```

```
<!-- http://schemas.ogf.org/nml/2013/05/base#hasLabelGroup -->

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#hasLabelGroup">
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#LinkGroup"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#PortGroup"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#LabelGroup"/>
</owl:ObjectProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#hasLink -->

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#hasLink">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Group"/>
    <rdfs:range>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Link"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#LinkGroup"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:range>
</owl:ObjectProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#hasNode -->

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#hasNode">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Node"/>
</owl:ObjectProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort -->

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    <rdfs:range>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Port"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#PortGroup"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:range>
</owl:ObjectProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#hasPort -->

<owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#hasPort">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Group"/>
    <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Port"/>
</owl:ObjectProperty>
```

```
    <!-- http://schemas.ogf.org/nml/2013/05/base#hasService -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#hasService">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Service"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#hasTopology -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#hasTopology">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Topology"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#implementedBy -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#implementedBy">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#isAlias -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#isAlias">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#isSerialCompoundLink -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#isSerialCompoundLink">
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Link"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#LinkGroup"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#ListItem"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#isSink -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#isSink">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
        <rdfs:range>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Link"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#LinkGroup"/>
                </owl:unionOf>
            </owl:Class>
```

```xml
        </rdfs:range>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#isSource -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#isSource">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
        <rdfs:range>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Link"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#LinkGroup"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#item -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#item">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#ListItem"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#labeltype -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#labeltype">
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Label"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#LabelGroup"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#locatedAt -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#locatedAt">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Location"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#next -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#next">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#ListItem"/>
        <rdfs:range rdf:resource="http://schemas.ogf.org/nml/2013/05/base#ListItem"/>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#providesLink -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#providesLink">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Service"/>
```

```
        <rdfs:range>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Link"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#LinkGroup"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#providesPort -->

    <owl:ObjectProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#providesPort">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Service"/>
        <rdfs:range>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Port"/>
                    <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#PortGroup"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>


    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Data properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->


    <!-- http://schemas.ogf.org/nml/2013/05/base#alt -->

    <owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#alt">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Location"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    </owl:DatatypeProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#end -->

    <owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#end">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Lifetime"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
    </owl:DatatypeProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#labelSwapping -->

    <owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#labelSwapping">
        <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#SwitchingService"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    </owl:DatatypeProperty>


    <!-- http://schemas.ogf.org/nml/2013/05/base#lat -->
```

```
<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#lat">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Location"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#long -->

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#long">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Location"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#name -->

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#name">
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#Location"/>
                <rdf:Description rdf:about="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#noReturnTraffic -->

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#noReturnTraffic">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Link"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#parameter -->

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#parameter">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#start -->

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#start">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Lifetime"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
</owl:DatatypeProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#unlocode -->

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#unlocode">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Location"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

```
<!-- http://schemas.ogf.org/nml/2013/05/base#value -->

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#value">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Label"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#values -->

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#values">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#LabelGroup"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>


<!-- http://schemas.ogf.org/nml/2013/05/base#version -->

<owl:DatatypeProperty rdf:about="http://schemas.ogf.org/nml/2013/05/base#version">
    <rdfs:domain rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
</owl:DatatypeProperty>


<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://schemas.ogf.org/nml/2013/05/base#AdaptationService -->

<owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#AdaptationService">
    <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Service"/>
</owl:Class>


<!-- http://schemas.ogf.org/nml/2013/05/base#BidirectionalLink -->

<owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#BidirectionalLink">
    <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Group"/>
</owl:Class>


<!-- http://schemas.ogf.org/nml/2013/05/base#BidirectionalPort -->

<owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#BidirectionalPort">
    <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Group"/>
</owl:Class>


<!-- http://schemas.ogf.org/nml/2013/05/base#DeadaptationService -->

<owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#DeadaptationService">
    <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Service"/>
```

```
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#Group -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#Group">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#Label -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#Label"/>


    <!-- http://schemas.ogf.org/nml/2013/05/base#LabelGroup -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#LabelGroup">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#Lifetime -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#Lifetime"/>


    <!-- http://schemas.ogf.org/nml/2013/05/base#Link -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#Link">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#LinkGroup -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#LinkGroup">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Group"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#ListItem -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#ListItem">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#Location -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#Location">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#NetworkObject -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
```

```
    <!-- http://schemas.ogf.org/nml/2013/05/base#Node -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#Node">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#Port -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#Port">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#PortGroup -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#PortGroup">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Group"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#Service -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#Service">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#NetworkObject"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#SwitchingService -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#SwitchingService">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Service"/>
    </owl:Class>


    <!-- http://schemas.ogf.org/nml/2013/05/base#Topology -->

    <owl:Class rdf:about="http://schemas.ogf.org/nml/2013/05/base#Topology">
        <rdfs:subClassOf rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Group"/>
    </owl:Class>
</rdf:RDF>
```

Appendix C Relation to G.800

This section describes the relation between terms defined in this recommendation and those defined in the ITU-T G.800 recommendation [G.800].

NML Term	**G.800 Term**
Port	Unidirectional Port
Link	Either Link Connection or Network Connection
SwitchingService	Subnetwork
AdaptationService	Combined Adaptation Function and Termination Function
BidirectionalPort	Port
Label	Resource label

References

Normative References

[GFD.202] Freek Dijkstra, and Jeroen van der Ham. A URN Namespace for Network Resources. GFD 202 (Community Practise), June 2013. URL http://www.ogf.org/documents/GFD.202.pdf.

[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.

[G.800] Unified functional architecture of transport networks. ITU-T Recommendation G.800, February 2012. URL http://www.itu.int/rec/T-REC-G.800/.

[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.

[Unicode] The Unicode Consortium. The Unicode Standard, Version 6.2.0. Mountain View, CA, USA. November 2012. ISBN 978-1-936213-07-8 Section 5.18, Case Mappings. Paragraph about Caseless Matching. Normative URL http://www.unicode.org/versions/Unicode6.2.0/ Informative URL ftp://ftp.unicode.org/Public/UNIDATA/CaseFolding.txt

[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.

[GFD.165] Paola Grosso, Aaron Brown, Aurélien Cedeyn, Freek Dijkstra, Jeroen van der Ham, Anand Patil, Pascale Primet, Martin Swany, and Jason Zurawski. Network Topology Descriptions in Hybrid Networks GFD 165 (Informational), March 2010. URL http://www.ogf.org/documents/GFD.165.pdf.

[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/.