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

Jeroen van der Ham, University of Amsterdam
Freek Dijkstra, SARA
Jason Zurawski, Internet2
Martin Swany, Indiana University

February 2012

Network Markup Language Base Schema version 1

Status of This Document

Community Practice (CP)

Copyright Notice

Abstract

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

Contents

1 introduction

This document describes the object model of the Network Markup Language. These basic objects may be extended, or sub-classed, to represent technology specific classes. These basic objects and extended objects will also be representable in multiple syntaxes, including at least XML and RDF.

1.1 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].

2 NML Topology Schema

The NML Topology schema describes an information model with elements and their relations that describe computer networks. This schema is kept intentionally general, with provisions to extend the base schema to describe layer-specific information.

The URI of the class-objects will become `http://http://schemas.ogf.org/nml/base/yyyy/mm/` where `yyyy/mm` is the four digit year and two digit month of the publication of the schema document.

2.1 Network

This document explicitly does not try to provide a definition of the term 'Network'. The working group has discussed the possible meanings of this term and in the end we were forced to conclude that is not possible to provide a workable precise definition for the term *Network*. 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.

Figure 1 shows an overview of all the objects in the NML schema in a UML class diagram. The figure also shows the relations between the objects, and their cardinalities. In the sections below we discuss each of the elements of the schema.

2.2 Network Object

The basic abstract element of the schema is the *Network Object*. Other basic elements inherit from it.

The *Network Object* can be associated with a *Location* using the *locatedAt* attribute, can be related to other instances via *Relation*s and can have a *Lifetime*. Every Network Object MUST have an *id* attribute, which MUST be a unique URI. These characteristics are inherited by the subclasses of the *Network Object* class.

The base *Network Object* has three related objects that describe the *Network Object* and its relationships:

- Location
- Lifetime
- Relation

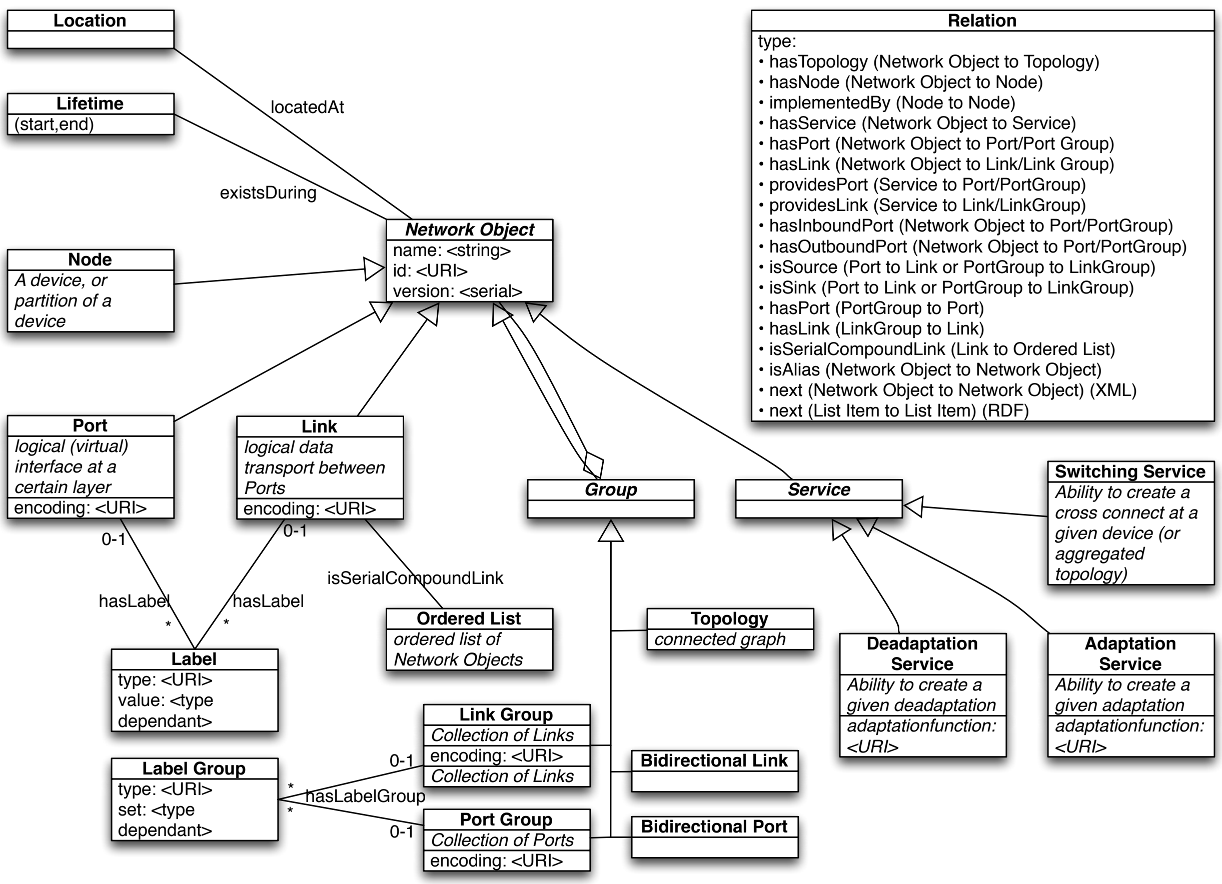

Figure 1: A UML class diagram of the objects in the NML schema and their relations

The location of an object in the physical world can be described using the *Location* object. The actual location is then described using properties of the *Location* object. The Location and a Network object are related to each other using the *locatedAt* relationship.

All *Network Objects* can potentially have a *Lifetime*, that consists of vector of *time* elements, which contain a start time and an end time.

The Relations between different network objects are represented using relation objects. These are discussed in more detail in section 2.21.

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

- Node
- Port
- Link
- Service
- Group

These objects are described in more detail below.

2.3 Node

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

A *Node* may have the following relations:

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

A *Node* may have the following attributes:

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

2.4 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. A *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

2.5 Link

A *Link* object describes that there is a unidirectional connection from one *Port* to another.

A *Link* may have the following relations:

- *existsDuring* to one or more *Lifetime*s
- *hasLabel* to one *Label*
- *hasSink* to one *Port*
- *hasSource* to one *Port*

- *isAlias* to one or more *Link*s
- *isSerialCompoundLink* to one ordered *List* of *Link*s

A *Link* may have the following attributes:

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

2.6 Service

2.7 Switching Service

A *SwitchingService* describes the ability to create cross connects between its ports. *SwitchingService* inherits from *Service*.

A *Switching Service* may have the following relations:

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

A *SwitchingService* may have the following attributes:

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

2.8 Adaptation Service

An *AdaptationService* describes the capability that data from one or more ports can be embedded in the data encoding of one other port. This is commonly referred to as the embedding of client layer ports in a server layer port.

An *AdaptationService* may have the following relations:

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

An *AdaptationService* may have the following attributes:

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

2.9 Deadaptation Service

A *DeadaptationService* describes the capability that data of one or more ports can be extracted from the data encoding of one other port. This is commonly referred to as the extraction of client layer ports from the server layer port.

A *DeadaptationService* may have the following relations:

- *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

2.10 Group

To describe collections of network objects, there is a group element. Any element defined above can be part of a group, including another group.

We also define a set of special groups:

- Bidirectional Link
- Bidirectional Port
- Topology
- Domain
- Network

2.11 Bidirectional Link

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

A *BidirectionalLink* may have the following relations:

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

A *BidirectionalLink* may have the following attributes:

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

2.12 Bidirectional Port

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

A *BidirectionalPort* may have the following relations:

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

A *BidirectionalPort* may have the following attributes:

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

2.13 Topology

A *Topology* is a set of connected Network Objects.

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

2.14 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

2.15 Link Group

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

A *LinkGroup* may have the following relations:

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

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

A *LinkGroup* may have the following attributes:

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

2.16 Label

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

A *Label* may have the following attributes:

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

Technology extensions of NML may define additional attributes.

2.17 Label Group

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

A *LabelGroup* may have the following attributes:

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

Technology extensions of NML may define additional attributes.

2.18 Location

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

A *Location* may have the following attributes:

- *long* is the longitude in WGS84 coordinate system (in decimal degrees)

- *lat* is the lattitude in WGS84 coordinate system (in decimal degrees)
- *alt* is the altitude in WGS84 coordinate system (in decimal meters)
- *unlocode* is the UN/LOCODE location identifier
- *address* is a vCard address
- *name* is a human-readable string

2.19 Lifetime

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

A *Lifetime* may have the following attributes:

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

2.20 Ordered List

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

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

2.21 Relation

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

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

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

hasLabelGroup assigns one *LabelGroup* to a *PortGroup*

hasLabel assigns one *Label* to a *Port*

hasLink is used for:

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

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

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

hasPort is used for:

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

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

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

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

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

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

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

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

isSerialCompoundLink is used to define that a *Link* or*LinkGroup* represents an ordered *List* of *Link*s or*LinkGroup*s. This must include cross-connects. It MAY also be derived from an existing description, for example:

```
Port P1 --isSource--> Link L1
Port P2 --isSink--> Link L1
Port P2 --isSource--> Link L2
Port P3 --isSink--> Link L2
```

```
Port P1 --isSource--> Link L3
Port P3 --isSink--> Link L3
```

isSerialCompoundLink to one ordered *List* of

isSink is the inverse of *hasSource*

isSource is the inverse of *hasSink*

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

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

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

3 Identifiers

3.1 Object Identifiers

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

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

3.2 Instance Identifiers

Section 2.2 requires that instances of Network Objects MUST have an *id* attribute, which MUST be a unique URI.

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

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

Two different Network Objects instance MUST have two different identifiers.

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

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

3.2.1 Lexical Equivalence

Two identifier are lexical equivalent if they are binary equivalent after case-normalisation.

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

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

For example the following identifiers are equivalent:

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

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

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

3.2.2 Further Restrictions

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

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

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

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

3.2.3 Interpreting Identifiers

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

3.2.4 Network Object Attribute Change

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

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

4 Examples

5 Security Considerations

There are important security concerns associated with the generation and distribution of network topology information. For example, ISPs frequently consider network topologies to be proprietary. We do not address these concerns in this document, but implementers are encouraged to consider the security implications of generating and distributing network topology information.

6 Glossary

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

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

D. Martin Swany
Indiana University, School of Informatics and Computing
Lindley Hall Room 215
150 S. Woodlawn Avenue
Bloomington, Indiana 47405-7104
USA
Email: swany@iu.edu

8 Acknowledgments

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

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

References

Normative References

[URN-OGF-NETWORK] Freek Dijkstra. URN:OGF:network specification. GWD-R-P (Work in Progress), May 2011.

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

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

Informative References