ETSI GR CIM 038 V1.1.1 (2024-11)

Context Information Management (CIM); YANG and NGSI-LD interoperability

Disclaimer

The present document has been produced and approved by the cross-cutting Context Information Management (CIM) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGR/CIM-0038

Keywords
data models, information model, interoperability,
network management, YANG

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Important notice

The present document can be downloaded from the
ETSI Search & Browse Standards application.

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format on ETSI deliver.

Users should be aware that the present document may be revised or have its status changed,
this information is available in the Milestones listing.

If you find errors in the present document, please send your comments to
the relevant service listed under Committee Support Staff.

If you find a security vulnerability in the present document, please report it through our
Coordinated Vulnerability Disclosure (CVD) program.

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.
No recommendation as to products and services or vendors is made or should be implied.
No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.
In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.
The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2024.
All rights reserved.

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) cross-cutting Context Information Management (CIM).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

The present document is intended to provide developers and network operators with guidelines towards achieving interoperability between Yet Another Next Language (YANG) [i.1] and the Next Generation Service Interface Linked Data (NGSI-LD) information model [i.2]. The present document outlines a set of generic rules which can facilitate the translation of YANG data into a knowledge graph based on the NGSI-LD standard [i.2]. Practical examples are included throughout the present document to help readers understand the different types of mappings depending on the characteristics of the network devices.

Introduction

Since the advent of the YANG language within the IETF, the network industry has experienced an evolution towards the management and automation of operation of network devices. Traditional network operations based on CLIs are being replaced with APIs based on the YANG language and network management protocols. The amount and variety of YANG data models developed by vendors, open source projects, and consortia has grown exponentially. This opens the door to vast amounts of network data that operators can collect and integrate to optimize the management of the network. In this sense, the NGSI-LD standard [i.2], with its information model and REST-based API, becomes an enabler for integrating all these network data. To this end, the present document aims to provide guidelines towards improving the interoperability between YANG and NGSI-LD.

Contributions to the present document have been supported by the European Union's Horizon 2020 research project PALANTIR (Grant number 883335) and the European Union's Horizon 2020 research projects aerOS (Grant number 101069732) and ROBUST-6G (Grant number 101139068).

1 Scope

The present document targets developers and operators from the network and telecommunications industry who aim at integrating YANG data in a graph, following the NGSI-LD information model [i.2] and leveraging the capabilities of the NGSI-LD API [i.3]. The present document focuses on the mapping of the YANG metamodel [i.1] to the NGSI-LD information model, proposing generic rules that can achieve an automatic or semi-automatic translation of YANG data onto a graph based on the NGSI-LD standard [i.2]. Interactions with network devices using network management protocols based on YANG are discussed in the present document (see Annex B), albeit implementation details on their usage and mapping to the NGSI-LD API [i.3] are out of scope.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] IETF RFC 7950: "The YANG 1.1 Data Modelling Language".

[i.2] ETSI GS CIM 006: "Context Information Management (CIM); Information Model (MOD0)".

[i.3] ETSI GS CIM 009: "Context Information Management (CIM); NGSI-LD API".

[i.4] IETF RFC 9254: "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)".

[i.5] IETF RFC 7951: "JSON Encoding of Data Modeled with YANG".

[i.6] YANG to Protobuf: "Transformation Specification".

[i.7] W3C®: "SKOS Simple Knowledge Organization System Reference".

[i.8] IETF RFC 8342: "Network Management Datastore Architecture (NMDA)".

[i.9] IETF RFC 8344: "A YANG Data Model for IP Management".

[i.10] IETF RFC 8340: "YANG Tree Diagrams".

[i.11] IETF RFC 7277: "A YANG Data Model for IP Management".

[i.12] OpenConfig: "Vendor-neutral, model-driven network management designed by users".

[i.13] OpenConfig: "openconfig-interfaces YANG data model".

[i.14] IETF RFC 6241: "Network Configuration Protocol (NETCONF)".

[i.15] IETF RFC 8040: "RESTCONF Protocol".

[i.16] OpenConfig: "gRPC Network Management Interface (gNMI)".

[i.17] IETF RFC 3635: "Definitions of Managed Objects for the Ethernet-like Interface Types".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

NGSI-LD attribute: reference to both an NGSI-LD Property and to an NGSI-LD Relationship

NGSI-LD attribute type: categorization of an NGSI-LD Property or NGSI-LD Relationships

NGSI-LD element: any JSON element that is defined by the NGSI-LD API

NGSI-LD entity: informational representative of something that is supposed to exist in the real world, physically or conceptually

NGSI-LD entity type: categorization of an NGSI-LD Entity as belonging to a class of similar entities, or sharing a set of characteristic properties

NGSI-LD name: short-hand string (term) that locally identifies an NGSI-LD Entity Type, Property Type or Relationship Type and which can be mapped to a URI which serves as a fully qualified identifier

NGSI-LD property: description instance which associates a main characteristic, i.e. an **NGSI-LD Value**, to either an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property and that uses the special *hasValue* property to define its target value

NGSI-LD relationship: description of a directed link between a subject which is either an NGSI-LD Entity, an NGSI-LD Property or another NGSI-LD Relationship on one hand, and an object, which is an NGSI-LD Entity, on the other hand, and which uses the special *hasObject* property to define its target object

YANG data node: node in the schema tree that can be instantiated in a data tree

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
CBOR	Concise Binary Object Representation
CLI	Command Line Interface
gNMI	gRPC Network Management Interface
gRPC	Google Remote Procedure Call
HTTP	Hyper Text Transfer Protocol
JSON	JavaScript Object Notation
JSON-LD	JSON Linked Data
NETCONF	Network Configuration protocol
NGS-LD	Next Generation Service Interfaces Linked Data
NMDA	Network Management Datastore Architecture
SSH	Secure SHell
URI	Uniform Resource Identifier
XML	eXtensible Markup Language
YANG	Yet Another Next Generation

4 YANG to NGSI-LD mapping

4.1 Introduction

Clause 4 provides rules for transforming YANG into NGSI-LD. First, generic rules for mapping YANG data nodes into the NGSI-LD meta-model are described. Next, additional rules for managing YANG configuration and operational data are provided.

4.2 Core mapping rules

4.2.0 Foreword

Clause 4.2 introduces rules for mapping YANG data node types to the classes of the NGSI-LD meta-model. The transformation takes as input the effective YANG schema, which means, the compiled set of YANG parent data model, imported data models, and any data models that may introduce augment and deviations to the other data models.

The main advantage with this approach is that mapping rules are defined at the YANG level. This allows for seamless translation of any YANG data to NGSI-LD regardless of the encoding format used for transporting YANG data such as XML, CBOR [i.4], JSON [i.5], or Protobuf [i.6].

Figure 4.2-1 depicts a high level mapping of the main YANG data node types to the core classes of the NGSI-LD meta-model.

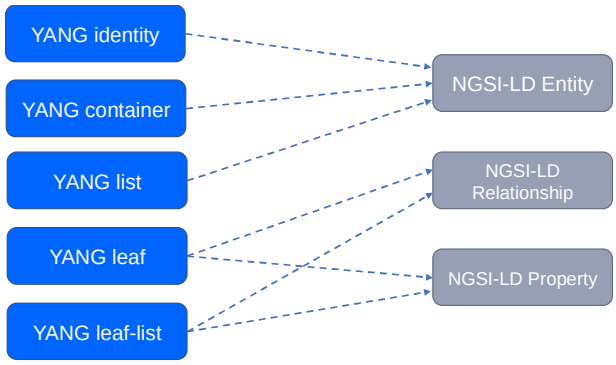

Figure 4.2-1: High-level mapping of main YANG data node types to NGSI-LD meta-model

In general, YANG identity, YANG container, and YANG list will map to NGSI-LD Entity Type, whereas YANG leaf and YANG leaf-list will map either to NGSI-LD Relationship or to NGSI-LD Property.

In the following clauses, the rules for mapping each YANG data node type to its respective NGSI-LD meta-model class are described in detail.

4.2.1 YANG identity

YANG identity introduces a special case as it represents an abstract concept that may inherit from other YANG base identities. In this sense, YANG identities will map to NGSI-LD Entities of a new Entity Type called *YANGIdentity*, which is a subclass of the *Concept* class as defined by the standard SKOS Ontology [i.7]. This allows for managing YANG identities as SKOS concepts, and therefore, composing taxonomies by using a standardized vocabulary. The new *YANGIdentity* NGSI-LD Entity Type defines the following NGSI-LD Properties:

- **namespace:** the namespace URI of the YANG data model where the YANG identity has been defined.
- **identifier:** the identifier of the YANG identity has defined in the YANG data model.
- **description:** the description of the YANG identity has defined in the YANG data model.

And leverages the following relationship inherited from the SKOS Concept:

- **broader**: the target object of the relationship is determined by the *base* statement in the definition of the YANG identity.

Figure 4.2.1-1 depicts the schema of the proposed of the *YANGIdentity* type.

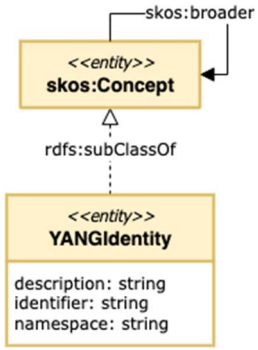

Figure 4.2.1-1: Schema of the new *YANGIdentity* Entity Type

A detailed example of YANG identities mapped to NGSI-LD can be found in clause A.1.

4.2.2 YANG container

A YANG non-presence container that includes child data nodes maps to an NGSI-LD Entity Type. When the YANG container does not contain child nodes but includes the *presence* statement (i.e. a YANG presence container), then the container maps to an NGSI-LD Entity Type.

There are scenarios when a YANG non-presence container has a YANG list as its only child. This modeling technique, commonly known as "enclosing container", has been widely used in YANG as it provides YANG clients with the means to retrieve all members of the child YANG list. For these scenarios, the container can be omitted as it carries no semantic meaning, therefore, it will not be mapped to NGSI-LD. As a result, this design choice makes the graph more compact, thus improving its usability.

Additionally, if the YANG container is the child of another YANG data node (container or list), its mapped NGSI-LD Entity Type will include an NGSI-LD Relationship that points to the NGSI-LD Entity Type representing the parent YANG node. This type of NGSI-LD Relationship captures the hierarchical (child-to-parent) connection between both YANG data nodes. This Relationship carries no semantics but simply a particular structure for organizing the data.

The NGSI-LD Name of the mapped NGSI-LD Entity Type derives from the identifier of the YANG container, following the upper camel case convention. The associated JSON-LD URI for the NGSI-LD Entity Type is generated by combining the namespace of the YANG module that defines the YANG data node with the XPath that identifies the YANG container. In the case of having a hierarchical relationship, the standardized *isPartOf* from Dublin Core ontology is leveraged to represent this NGSI-LD Relationship. Taking the example from clause A.2, the YANG container *statistics*, which is child of the YANG list *interface*, translates to an NGSI-LD Entity Type *Statistics* that includes an NGSI-LD Relationship *ispartOf* with target an NGSI-LD Entity Type *Interface*.

4.2.3 YANG list

A YANG list maps to an NGSI-LD Entity Type, where there will be an NGSI-LD Entity for every member of the YANG list.

Similar to YANG containers, a YANG list might also be contained in a parent YANG container or YANG list. In this case, the NGSI-LD Entity Type representing the child YANG list includes an NGSI-LD Relationship that points to the NGSI-LD Entity Type that represents the parent YANG node.

The NGSI-LD Name of the mapped NGSI-LD Entity Type derives from the identifier of the YANG list, following the upper camel case convention. The associated JSON-LD URI for the NGSI-LD Entity Type is generated by combining the namespace of the YANG module that defines the YANG data node with the XPath that identifies the YANG list. When representing the hierarchical relationship (child-to-parent), the same rule defined in clause 4.2.2 will apply, i.e. the *isPartOf* relationship standardized in the Dublin Core ontology is leveraged.

4.2.4 YANG leaf

A YANG leaf maps to an NGSI-LD Attribute of an NGSI-LD Entity Type that represents the parent YANG data node (container or list) of the child YANG leaf. In this case, the data type of the YANG leaf will determine whether it will map to an NGSI-LD Property or an NGSI-LD Relationship:

- When the data type of the YANG leaf is a considered a **literal**, then the YANG leaf maps to an NGSI-LD Property. Table 4.2.4-1 captures the JSON-LD data types to be used when converting from the different YANG built-in data types. Note in the table that custom data type *yang:date-and-time* (or *oc-type:timeticks64* in data models from OpenConfig [i.12]) has been included as it requires a special mapping. When the data type of the YANG leaf uses this custom data type, then the leaf maps to an NGSI-LD Property whose value will be of *DateTime* type as mandated by clause 4.6.3 in ETSI GS CIM 009 [i.3].

Table 4.2.4-1: Data type conversion

YANG built-in data types	JSON-LD data types (based on JSON Schema)
Integer (see note 1)	Integer
decimal64	Number
string (see note 2)	String (format pattern, if any)
boolean	Boolean
enum	String
bit	String[]
binary	String
empty (see note 3)	String
ietf-yang-types:date-and-time (see note 4)	String (format date-time)
NOTE 1: Integer built-in types as per clause 9.2 in IETF RFC 7950 [i.1] include the following: int8, int16, int32, int64, uint8, uint16, uint32, uint64.	
NOTE 2: If YANG string includes a regex pattern, the mapping to JSON-LD will also include the same string pattern.	
NOTE 3: The *empty* YANG datatype maps to a literal in JSON-LD represented by an empty string. The reason for this choice is that "null" value is reserved in NGSI-LD for deleting NGSI-LD Attributes.	
NOTE 4: Custom data type defined by IETF for representing timestamps. OpenConfig defines its own data type "openconfig-types:timeticks64", which will have the same mapping to JSON-LD string date-time format.	

- A YANG leaf that uses the ***leafref*** data type will map to an NGSI-LD Relationship. The target of the NGSI-LD Relationship is an NGSI-LD Entity Type that is identified by the path of the YANG leafref. Note that if the YANG leaf referenced in the *leafref* is not the key of a YANG list, or belongs to YANG container, then the target of the NGS-LD Relationship may contain multiple NGSI-LD Entities that match the value of the YANG *leafref*.

- When the data type of a YANG leaf is an ***identityref***, then the YANG leaf will map to an NGSI-LD Relationship whose target is an NGSI-LD Entity of *YANGIdentity* type that represents the YANG identity.

- Lastly, a YANG leaf that uses the ***instance-identifier*** data type maps to an NGSI-LD Relationship. In this case, the target of the Relationship is a particular NGSI-LD Entity identified through the path of the instance-identifier.

The NGSI-LD Name of the mapped NGSI-LD Attribute derives from the identifier of the YANG leaf, following the lower camel case convention. For those identifiers using terms that are reserved in NGSI-LD (i.e. "id", "type"), the final Name of the Attribute appends the Name of the Entity in lower camel case convention to plus the original Name of the Attribute. For example, a *type* YANG leaf within *interface* YANG container becomes *interfaceType* in NGSI-LD. The associated JSON-LD URI for the NGSI-LD Attribute Type is generated by combining the namespace of the YANG module that defines the YANG data node with the XPath that identifies the YANG leaf.

4.2.5 YANG leaf-list

Mapping for YANG leaf-list follows the same mapping rules defined for YANG leaf as mandated in clause 4.2.4, but in this case the value of the NGSI-LD Attribute will be an array.

4.2.6 YANG anydata

YANG anydata node encapsulates a set of YANG data nodes that are unknown upon design time of the YANG module. Therefore, a YANG anydata node will map to an NGSI-LD Entity Type whose content will depend on the YANG data nodes included by the YANG anydata node at the implementation time. These included YANG data nodes will, in turn, follow the above described mapping rules accordingly.

4.2.7 YANG anyxml

This type of YANG data node may represent any portion of XML data. Therefore, the contents of the YANG anyxml node will translate into an NGSI-LD Property whose value is the XML portion captured as a string.

4.3 Mapping YANG configuration and operational state data

4.3.0 Foreword

The YANG language enables specifying within a YANG data model which data is meant for configuration and which data represents the operational state in a YANG server, e.g. a network device. YANG data nodes that are specified as configuration represent the data that a YANG client can write as configuration to be applied in the YANG server. YANG data nodes that are marked as operational state data represent data that a YANG client can only read from the server. To store these different types of data, YANG servers implement databases called YANG datastores.

4.3.1 IETF-style data models

4.3.1.1 Data models compatible with NMDA

Network Management Datastore Architecture (NMDA) [i.8] defines a unified architecture for defining YANG data model. The idea behind this architecture is that a single YANG data model can have multiple instantiations in different YANG datastores depending on the purpose of data, e.g. configuration or operational. When interacting with YANG servers that support NMDA, network management protocols like NETCONF can select a particular datastore in the YANG server. Therefore, YANG clients see a single YANG data model, whose configuration or operational state data can be then selected by specifying the target datastore.

Following the NMDA approach, NGSI-LD can accommodate the representation of YANG data pertaining to different datastores for the same YANG data model. The NGSI-LD standard [i.2] specifies the *datasetId* sub attribute as a mechanism to identify different values for the same NGSI-LD Attribute depending on the data source. In this sense, the *datasetId* sub attribute can be leveraged to identify the source datastore of the mapped YANG data node in its NGSI-LD representation. Thus, a specific *datasetId* can be used to differentiate YANG configuration data from the YANG operational state data.

Mapping a YANG data model compatible with NMDA to NGSI-LD is a straightforward process. The mapping to NGSI-LD (following rules mandated by clause 4.2) will model the resulting NGSI-LD Attribute only once, which in turn, may contain multiple NGSI-LD Attribute instances with different *datasetId,* identifying the source YANG data store. Figure 4.3.1.1-1 shows a snapshot of the latest release of the *ietf-ip* YANG module [i.9], where the *forwarding* YANG leaf maps to the *forwarding* NGSI-LD Property. When interacting with this YANG leaf in the configuration datastore, there will be an instance of the *forwarding* NGSI-LD Property whose *datasetId* would point to "configuration", whereas when the interaction occurs in the YANG operational datastore, the *datasetId* of the *enabled* NGSI-LD Property instance would point to "operational".

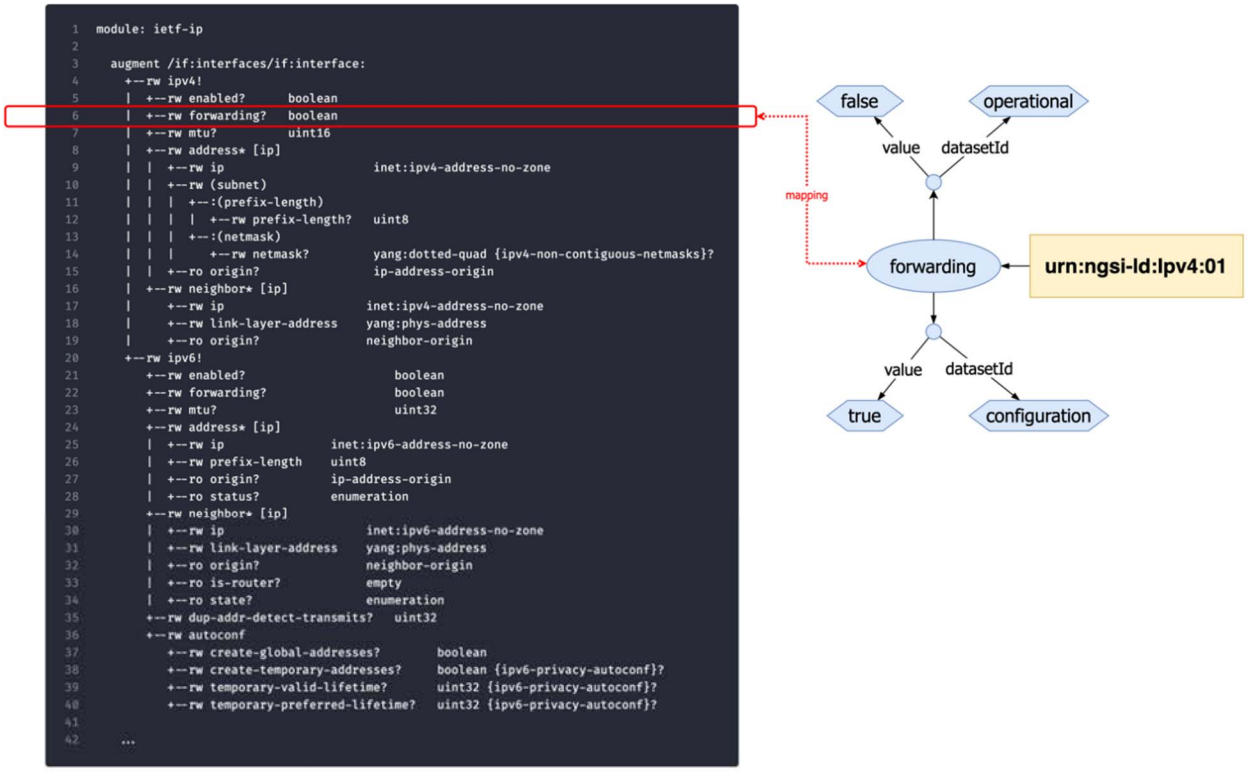

Figure 4.3.1.1-1: Example of mapping NMDA data model to NGSI-LD

The left part of Figure 4.3.1.1-1 shows the YANG tree representation [i.10] of the *ietf-ip* YANG module. The right part of Figure 4.3.1.1-1 depicts the *forwarding* NGSI-LD Property with different values per datasetId.

It needs to be noted that only the "configuration" and "operational" YANG datastores have been mentioned thus far. However, NMDA-compatible devices can implement additional datastores that contain YANG data structured with the same YANG data model. The values of the NGSI-LD *datasetId* could be extended to reference these additional datastores such as "candidate" (configuration data stored in the device but not applied) or "startup" (configuration data stored in the device which will be applied upon the next reboot).

4.3.1.2 Traditional data models (split layout)

Before NMDA came on the scene, network management protocols could not select a particular YANG datastore when interacting with network devices. To overcome this limitation, YANG data models would follow a split layout to enable identifying the datastore containing the data. These data models are split into two branches: one containing configuration data and another containing the operational data. Each branch begins with a YANG container, where the second, which is differentiated by the suffix "-state", includes a replica of the contents of the first container.

Mapping these data models to NGSI-LD follows a similar process as described in clause 4.3.1.2, but this time it will additional logic for parsing the split layout. Similarly, the goal is to have a unified NGSI-LD model where both branches are merged, thus the mapped NGSI-LD Attributes are modelled only once. However, now the translation of each NGSI-LD Attribute instance needs to consider which branch of the YANG data model is used in order to identify the source YANG datastore, and therefore, the *datasetId*.

Figure 4.3.1.2-1 depicts a fragment of the first release of the *ietf-ip* YANG module [i.11], which in this case follows a split layout wherein the *interfaces* and *interfaces-state* containers are placed at the root of the data model. The mapping to NGSI-LD will again result into only one Entity Type *Ipv4* with an NGSI-LD Property *forwarding*, however, now the Property instance with *datasetId* "configuration" maps to */interfaces/interface/ipv4/forwarding* leaf, whereas the Property instance with *datasetId* "operational" maps to */interfaces-status/interface/ipv4/forwarding* leaf.

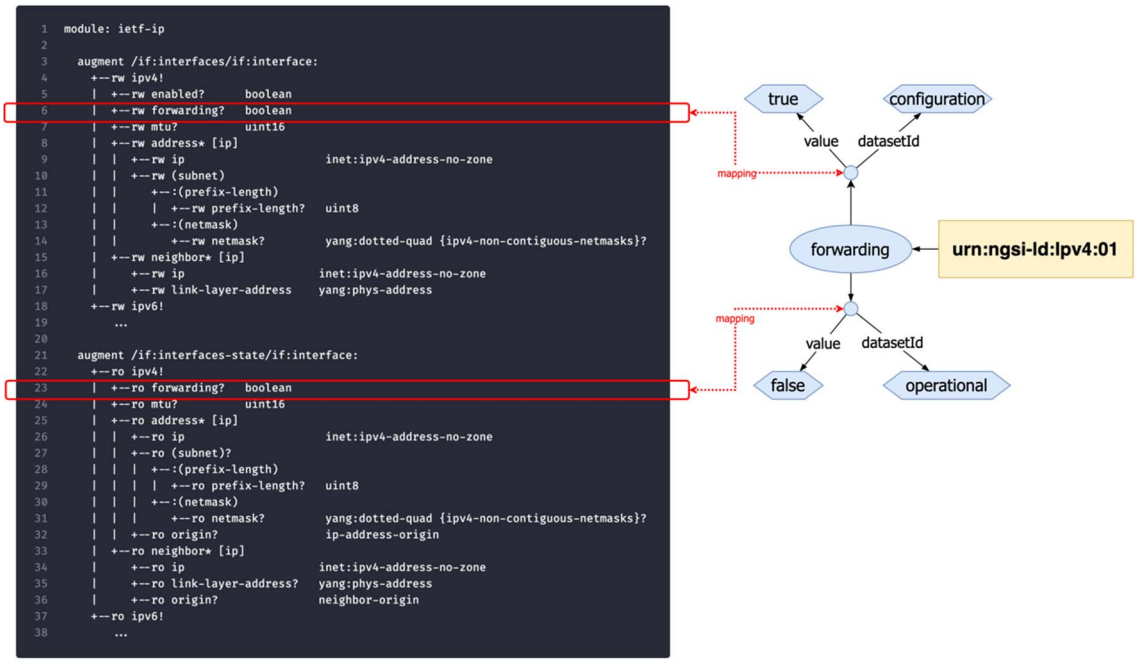

Figure 4.3.1.2-1: Example of mapping split-based data model to NGSI-LD

The left part of Figure 4.3.1.2-1 shows the YANG tree representation of the *ietf-ip* YANG module [i.11]. The right part of Figure 4.3.1.2-1 depicts the *forwarding* NGSI-LD Property with different values per datasetId.

4.3.2 OpenConfig data models

OpenConfig [i.12] is an open-source project built around a community of network vendors, network operators, and cloud providers. The goal of the project is defining and implementing common, vendor-independent software layer for managing network devices.

One of the contributions from OpenConfig is the creation of standard YANG data models for network management. However, OpenConfig data models follow a different structure from the structures followed in IETF data models as shown previously in clause 4.3.1. The main difference in OpenConfig YANG data models is that YANG leaf and leaf-lists are mirrored in *config* and *state* containers. To some extent, this approach is similar to the split-based structure from IETF, but in this case the configuration and operational state branches (*config* and *state* containers respectively) are modelled at every level of the data model. Additionally, when modeling YANG lists, OpenConfig data models include a reference to the YANG leaf that represents the key of YANG list in its configuration branch.

Figure 4.3.2-1 depicts a fragment of the *openconfig-interfaces* module [i.13] and an example mapping to NGSI-LD. The *interface* YANG list maps to the *Interface* NGSI-LD Entity Type. The top *name* YANG leaf is omitted, and the two child *config* and *state* containers are merged into a unified data model. For example, the resulting *enabled* NGSI-LD Property will include a Property instance with *datasetId* "configuration" mapping to */interfaces/interface/config/enabled* leaf, whereas the Property instance with *datasetId* "operational" maps to */interfaces/interface/state/enabled* leaf.

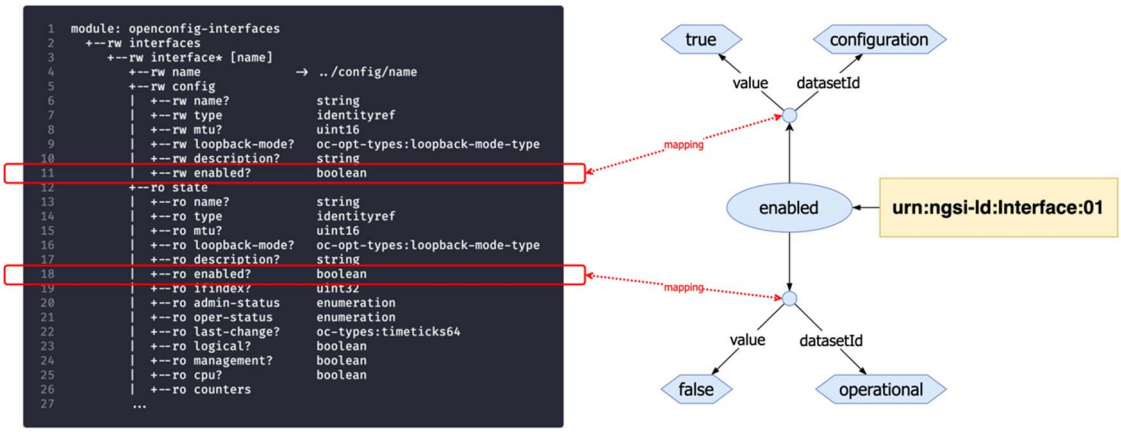

Figure 4.3.2-1: Example of mapping an OpenConfig data model to NGSI-LD

Annex A:
Mapping examples

A.1 Mapping YANG identities

Two example YANG modules are represented below. The first module "example-crypto-base" defines a base YANG identity, while the second module "example-des" defines another YANG identity that inherits the previous YANG identity from the first module:

```
module example-crypto-base {
  yang-version 1.1;
  namespace "urn:example:crypto-base";
  prefix "crypto";

  identity crypto-alg {
    description
      "Base identity from which all crypto algorithms
      are derived.";
  }
}
```

```
module example-des {
  yang-version 1.1;
  namespace "urn:example:des";
  prefix "des";

  import "example-crypto-base" {
    prefix "crypto";
  }

  identity des {
    base "crypto:symmetric-key";
    description "DES crypto algorithm.";
  }
}
```

For this example, the NGSI-LD entities generated for this example are displayed below in the NGSI-LD normalized format:

```
[
  {
    "id": "urn:ngsi-ld:YANGIdentity:1",
    "type": "YANGIdentity",
    "namespace": {
      "type": "Property",
      "value": "urn:example:crypto-base"
    },
    "identifier": {
      "type": "Property",
      "value": "crypto-alg"
    },
    "description": {
      "type": "Property",
      "value": "Base identity from which all crypto algorithms are derived"
    }
  },
  {
    "id": "urn:ngsi-ld:YANGIdentity:2",
    "type": "YANGIdentity",
    "namespace": {
      "type": "Property",
      "value": "urn:example:des"
    },
    "identifier": {
      "type": "Property",
      "value": "des"
    },
    "description": {
      "type": "Property",
      "value": "DES crypto algorithm."
    },
    "broader": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:YANGIdentity:1"
```

```
    }
  }
]
```

And their respective @context vocabulary would be:

```
{
  "@context": [
    {
      "ietf-yang": "http://ietf.yang.org#",
      "YANGIdentity": "ietf-yang:YANGIdentity",
      "description": "ietf-yang:description",
      "identifier": "ietf-yang:identifier",
      "namespace": "ietf-yang:namespace",
      "broader": "http://www.w3.org/2004/02/skos/core#broader"
    },
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.6.jsonld"
  ]
}
```

A.2 Mapping "ietf-interfaces" YANG data model

This clause shows how the "ietf-interfaces" data model [i.9] is mapped to NGSI-LD. To improve readability the model is presented below in YANG tree format:

```
module: ietf-interfaces
   +--rw interfaces
      +--rw interface* [name]
         +--rw name                          string
         +--rw description?                  string
         +--rw type                          identityref
         +--rw enabled?                      boolean
         +--rw link-up-down-trap-enable?     enumeration {if-mib}?
         +--ro admin-status                  enumeration {if-mib}?
         +--ro oper-status                   enumeration
         +--ro last-change?                  yang:date-and-time
         +--ro if-index                      int32 {if-mib}?
         +--ro phys-address?                 yang:phys-address
         +--ro higher-layer-if*              interface-ref
         +--ro lower-layer-if*               interface-ref
         +--ro speed?                        yang:gauge64
         +--ro statistics
            +--ro discontinuity-time    yang:date-and-time
            +--ro in-octets?            yang:counter64
            +--ro in-unicast-pkts?      yang:counter64
            +--ro in-broadcast-pkts?    yang:counter64
            +--ro in-multicast-pkts?    yang:counter64
            +--ro in-discards?          yang:counter32
            +--ro in-errors?            yang:counter32
            +--ro in-unknown-protos?    yang:counter32
            +--ro out-octets?           yang:counter64
            +--ro out-unicast-pkts?     yang:counter64
            +--ro out-broadcast-pkts?   yang:counter64
            +--ro out-multicast-pkts?   yang:counter64
            +--ro out-discards?         yang:counter32
            +--ro out-errors?           yang:counter32
```

Figure A.2-1 depicts the resulting schema after mapping the "ietf-interfaces" YANG data model to NGSI-LD:

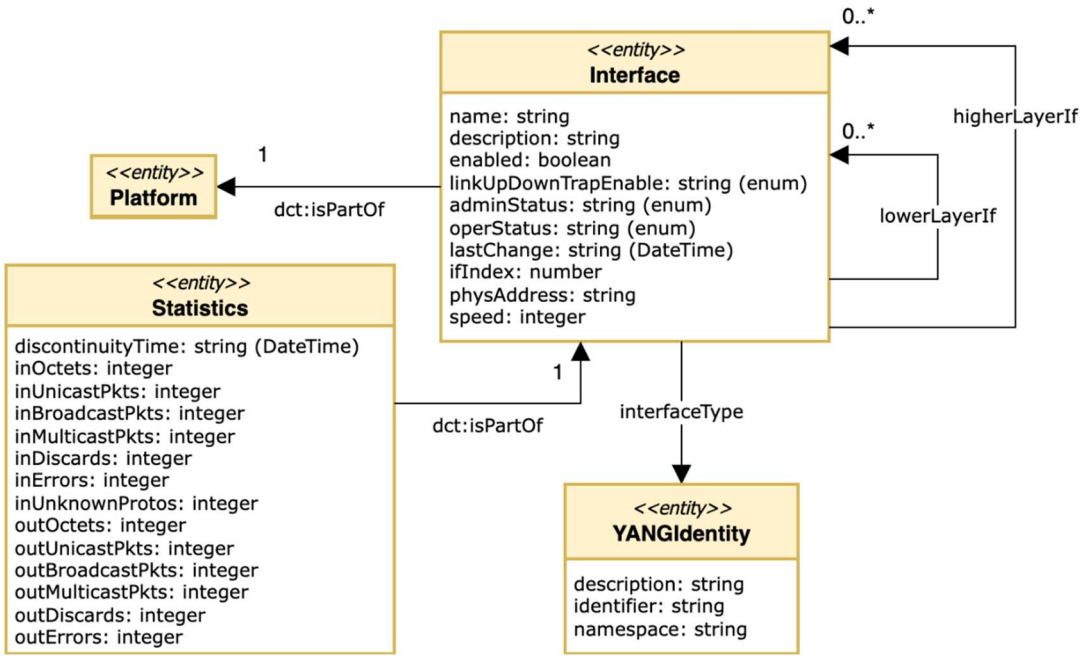

Figure A.2-1: NGSI-LD schema for ietf-interfaces YANG data model

A YANG data sample of "ietf-interfaces" that has been collected through NETCONF is shown below:

```
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
        <name>GigabitEthernet0/3/9</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <link-up-down-trap-enable>enabled</link-up-down-trap-enable>
        <admin-status>up</admin-status>
        <oper-status>down</oper-status>
        <last-change>2022-10-20T19:24:39Z</last-change>
        <if-index>20</if-index>
        <phys-address>5C:21:FB:FF:15:91</phys-address>
        <speed>1000000000</speed>
        <higher-layer-if>GigabitEthernet0/3/9.55</higher-layer-if>
        <statistics>
            <discontinuity-time>2022-07-13T17:22:06Z</discontinuity-time>
            <in-octets>29990214</in-octets>
            <in-unicast-pkts>32496</in-unicast-pkts>
            <in-broadcast-pkts>35089</in-broadcast-pkts>
            <in-multicast-pkts>914</in-multicast-pkts>
            <in-discards>0</in-discards>
            <in-errors>0</in-errors>
            <in-unknown-protos>0</in-unknown-protos>
            <out-octets>28978061</out-octets>
            <out-unicast-pkts>32520</out-unicast-pkts>
            <out-broadcast-pkts>12</out-broadcast-pkts>
            <out-multicast-pkts>27354</out-multicast-pkts>
            <out-discards>0</out-discards>
            <out-errors>0</out-errors>
        </statistics>
    </interface>
    <interface>
        <name>GigabitEthernet0/3/9.55</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:l2vlan</type>
        <enabled>true</enabled>
        <link-up-down-trap-enable>enabled</link-up-down-trap-enable>
        <admin-status>up</admin-status>
        <oper-status>down</oper-status>
        <last-change>2022-10-20T19:24:39Z</last-change>
        <if-index>274</if-index>
        <phys-address>5C:21:FB:FF:15:91</phys-address>
        <speed>1000000000</speed>
        <lower-layer-if>GigabitEthernet0/3/9</lower-layer-if>
        <statistics>
            <discontinuity-time>2022-10-13T18:34:26Z</discontinuity-time>
            <in-octets>0</in-octets>
```

```
            <in-unicast-pkts>0</in-unicast-pkts>
            <in-broadcast-pkts>0</in-broadcast-pkts>
            <in-multicast-pkts>0</in-multicast-pkts>
            <in-discards>0</in-discards>
            <in-errors>0</in-errors>
            <in-unknown-protos>0</in-unknown-protos>
            <out-octets>0</out-octets>
            <out-unicast-pkts>0</out-unicast-pkts>
            <out-broadcast-pkts>0</out-broadcast-pkts>
            <out-multicast-pkts>0</out-multicast-pkts>
            <out-discards>0</out-discards>
            <out-errors>0</out-errors>
        </statistics>
    </interface>
</interfaces>
```

Based on the mapping rules defined in clause 4.2, the NGSI-LD Entities generated for this example are displayed below in normalized format. Note that *ethernetCsmacd* and *l2vlan* YANG identities also have been represented for completeness:

```
[
  {
    "id": "urn:ngsi-ld:YANGIdentity:ianaift:ethernetCsmacd",
    "type": "YANGIdentity",
    "description": {
      "type": "Property",
      "value": "For all Ethernet-like interfaces, regardless of speed, as per IETF RFC 3635."
    },
    "identifier": {
      "type": "Property",
      "value": "ethernetCsmacd"
    },
    "namespace": {
      "type": "Property",
      "value": "urn:ietf:params:xml:ns:yang:iana-if-type"
    },
    "broader": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:YANGIdentity:ianaift:iana-interface-type"
    },
  },
  {
    "id": "urn:ngsi-ld:Interface:GigabitEthernet0/3/9",
    "type": "Interface",
    "isPartOf": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:Platform:Huawei:NE40-X2"
    },
    "name": {
      "type": "Property",
      "value": "GigabitEthernet0/3/9"
    },
    "interfaceType": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:YANGIdentity:ianaift:ethernetCsmacd"
    },
    "enabled": {
      "type": "Property",
      "value": true
    },
    "linkUpDownTrapEnable": {
      "type": "Property",
      "value": "enabled"
    },
    "adminStatus": {
      "type": "Property",
      "value": "up"
    },
    "operStatus": {
      "type": "Property",
      "value": "down"
    },
    "lastChange": {
      "type": "Property",
      "value": {
        "@type": "DateTime",
        "@value": "2022-10-20T19:24:39Z"
      }
    },
    "ifIndex": {
```

```
      "type": "Property",
      "value": 20
    },
    "physAddress": {
      "type": "Property",
      "value": "5C:21:FB:FF:15:91"
    },
    "speed": {
      "type": "Property",
      "value": 1000000000
    },
    "higherLayerIf": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:Interface:GigabitEthernet0/3/9.55"
    },
  {
    "id": "urn:ngsi-ld:Statistics:GigabitEthernet0/3/9",
    "type": "Statistics",
    "isPartOf": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:Interface:GigabitEthernet0/3/9"
    },
    "discontinuityTime": {
      "type": "Property",
      "value": {
        "@type": "DateTime",
        "@value": "2022-10-13T18:34:26Z"
      }
    },
    "inOctets": {
      "type": "Property",
      "value": 29990214
    },
    "inUnicastPkts": {
      "type": "Property",
      "value": 32496
    },
    "inBroadcastPkts": {
      "type": "Property",
      "value": 35089
    },
    "inMulticastPkts": {
      "type": "Property",
      "value": 914
    },
    "inDiscards": {
      "type": "Property",
      "value": 0
    },
    "inErrors": {
      "type": "Property",
      "value": 0
    },
    "inUnknownProtos": {
      "type": "Property",
      "value": 0
    },
    "outOctets": {
      "type": "Property",
      "value": 28978061
    },
    "outUnicastPkts": {
      "type": "Property",
      "value": 32520
    },
    "outBroadcastPkts": {
      "type": "Property",
      "value": 12
    },
    "outMulticastPkts": {
      "type": "Property",
      "value": 27354
    },
    "outDiscards": {
      "type": "Property",
      "value": 0
    },
    "outErrors": {
      "type": "Property",
      "value": 0
    },
  },
  {
```

```
    "id": "urn:ngsi-ld:YANGIdentity:ianaift:l2vlan",
    "type": "YANGIdentity",
    "description": {
      "type": "Property",
      "value": "Layer 2 Virtual LAN using 802.1Q."
    },
    "identifier": {
      "type": "Property",
      "value": "l2vlan"
    },
    "namespace": {
      "type": "Property",
      "value": "urn:ietf:params:xml:ns:yang:iana-if-type"
    },
    "broader": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:YANGIdentity:ianaift:iana-interface-type"
    },
  },
  {
    "id": "urn:ngsi-ld:Interface:GigabitEthernet0/3/9.55",
    "type": "Interface",
    "isPartOf": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:Platform:Huawei:NE40-X2"
    },
    "name": {
      "type": "Property",
      "value": "GigabitEthernet0/3/9.55"
    },
    "interfaceType": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:YANGIdentity:ianaift:l2vlan"
    },
    "enabled": {
      "type": "Property",
      "value": true
    },
    "linkUpDownTrapEnable": {
      "type": "Property",
      "value": "enabled"
    },
    "adminStatus": {
      "type": "Property",
      "value": "up"
    },
    "operStatus": {
      "type": "Property",
      "value": "down"
    },
    "lastChange": {
      "type": "Property",
      "value": {
        "@type": "DateTime",
        "@value": "2022-10-20T19:24:39Z"
      }
    },
    "ifIndex": {
      "type": "Property",
      "value": 274
    },
    "physAddress": {
      "type": "Property",
      "value": "5C:21:FB:FF:15:91"
    },
    "speed": {
      "type": "Property",
      "value": 1000000000
    },
    "lowerLayerIf": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:Interface:GigabitEthernet0/3/9"
    },
  },
  {
    "id": "urn:ngsi-ld:Statistics:GigabitEthernet0/3/9.55",
    "type": "Statistics",
    "isPartOf": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:Interface:GigabitEthernet0/3/9.55"
    },
    "discontinuityTime": {
      "type": "Property",
```

```
      "value": {
        "@type": "DateTime",
        "@value": "2022-10-13T18:34:26Z"
      }
    },
    "inOctets": {
      "type": "Property",
      "value": 0
    },
    "inUnicastPkts": {
      "type": "Property",
      "value": 0
    },
    "inBroadcastPkts": {
      "type": "Property",
      "value": 0
    },
    "inMulticastPkts": {
      "type": "Property",
      "value": 0
    },
    "inDiscards": {
      "type": "Property",
      "value": 0
    },
    "inErrors": {
      "type": "Property",
      "value": 0
    },
    "inUnknownProtos": {
      "type": "Property",
      "value": 0
    },
    "outOctets": {
      "type": "Property",
      "value": 0
    },
    "outUnicastPkts": {
      "type": "Property",
      "value": 0
    },
    "outBroadcastPkts": {
      "type": "Property",
      "value": 0
    },
    "outMulticastPkts": {
      "type": "Property",
      "value": 0
    },
    "outDiscards": {
      "type": "Property",
      "value": 0
    },
    "outErrors": {
      "type": "Property",
      "value": 0
    },
  }
]
```

Annex B:
Mapping network management operations to the NGSI-LD API

B.1 Architecture

Previously, clause 4 showed the rules for mapping YANG to the NGSI-LD meta-model, thus allowing YANG to be linked with other data from heterogenous sources. But, the mechanisms for accessing and interacting with the YANG data will also be taken into account.

Network platforms (e.g. devices, controllers, orchestrators) that implement YANG data models additionally support network management protocols for interacting with the YANG data. Such protocols enable operators and applications to configure and monitor network platforms based on the YANG data models that these platforms implement. Depending on the network management protocol, different transport mechanisms and different YANG serialization formats will be used when interacting with network platforms. The main protocols implemented in the industry are the following:

- **Network Configuration Protocol (NETCONF**) [i.14]: Standardized by IETF. Relies on SSH as the transport protocol and XML as the encoding format for YANG data.
- **RESTCONF** [i.15]**:** Standardized by IETF. Transport protocol based on REST (HTTP) with JSON (IETF representation) as encoding format.
- **gRPC Network Management Interface (gNMI)** [i.16]: Defined by OpenConfig. Relies on gRPC (HTTP) as the transport protocol. Supports multiple encoding formats for YANG data such as JSON (OpenConfig representation) or Protocol Buffer (a.k.a. Protobuf).

Given the variety of the existing network management protocols, the NGSI-LD API could be leveraged as a standard interface that unifies management of YANG data in network platforms. The NGSI-LD API can abstract operators from the actual protocol used for interacting with the YANG data. In this sense, the Context Source role can be leveraged to implement such an abstraction layer as depicted in Figure B.1-1.

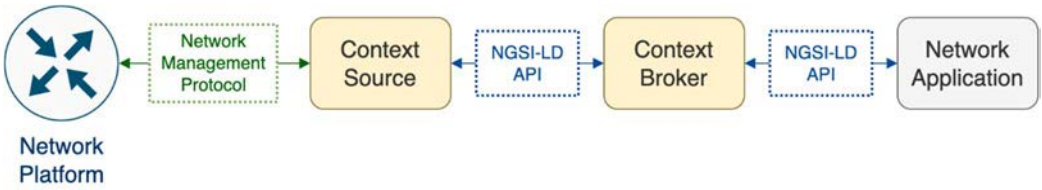

Figure B.1-1: Context Source working as a proxy between NGSI-LD and network management protocols

Once a new network platform is registered in the network, depending on the YANG data models that this platform implements, the mapping rules defined in clause 4 are generated and implemented by a Context Source. Based on the resulting mapping, the Context Source will register in the Context Registry the data it can expose from the network platform. In addition, the Context Source implements a client compatible with the network management protocol supported the network platform. Therefore, the Context Source is prepared to work as a proxy that translates NGSI-LD API operations into YANG-based network management operations, and vice-versa.

History

Document history		
V1.1.1	November 2024	Publication