Network Management Operations
Internet-Draft
Intended status: Informational
Expires: 24 April 2025

I. D. Martinez-Casanueva
L. Cabanillas
Telefonica
P. M. Martinez-Julia
NICT

21 October 2024

Knowledge Graphs for YANG-based Network Management
draft-marcas-nmop-knowledge-graph-yang-05

Abstract

The success of the YANG language and YANG-based protocols for managing the network has unlocked new opportunities in network analytics. However, the wide heterogeneity of YANG models hinders the consumption and analysis of network data. Besides, data encoding formats and transport protocols ~~will~~ differ depending on the network management protocol supported by ~~the~~ a network device. These challenges call for new data management ~~paradigms~~ approaches ~~that~~ to facilitate the discovery, understanding, integration, and access to silos of heterogenous YANG data, abstracting from the complexities of the network devices.

This document introduces the knowledge graph ~~paradigm~~ concept as a solution to this data management problem, with a focus on YANG-based network management. The document provides background on related topics such as ontologies and graph standards, and shares guidelines for implementing knowledge graphs from YANG data.

a mis en forme : Surlignage

Commenté [MB1]: There are no such guideline in the document

About This Document

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The size and complexity of networks ~~keeps~~ keep increasing, thus the path towards enabling an autonomous network requires ~~the~~ a combination of network telemetry mechanisms [RFC9232]. These mechanisms range from legacy protocols like SNMP to the recent model-driven telemetry (MDT) based on the YANG language [RFC7950] and network management protocols such as NETCONF [RFC6241], RESTCONF [RFC8040], or gNMI [GNMI].

MDT, in particular, has drawn the attention of the network industry owing to the benefits of modeling configuration and status data of the network with a formal data modeling language like YANG. However, since the inception of YANG, the network industry has experienced ~~the~~a massive creation of YANG data models developed by vendors, standards developing organizations (e.g., IETF, BBF, or IEEE), and consortia (e.g., OpenConfig). In turn, these data models target different abstraction layers of the network, namely, network elements, ~~and~~ network, and service [RFC8199]. Additionally, YANG data models may augment (or deviate from) other models to define new features (or remove or adjust existing ones) depending on the implementation. In summary, this trend has resulted into a wide variety of independent YANG data models, hence, the creation of data silos in the network. Refer to Sections 4.1 and 4.4 of [I-D.boucadair-nmop-rfc3535-20years-later] for a discussion on the fragmented YANG ecosystem and the integration complexity issues.

Commenté [MB2]: Note that this has a specific meaning in YANG. I would use another term.

Such amount and heterogeneity of YANG data models has hindered the ~~Collection~~collection, integration, and combination of network data for advanced network analytics. The current landscape shows different YANG models referencing the same concepts in a different way. For example, the IETF "ietf-interface" [RFC8343] and OpenConfig "openconfig-interfaces" [oc-interfaces] follow different structures and syntax, but both reference the same "interface" concept. The same can be observed between proprietary and standard models.

Similarly, there are YANG models, like Service Assurance [RFC9418], that convey semantic relationships with other concepts via identifiers. Figure 1 depicts ~~the~~ a YANG tree diagram [RFC8340] for a subservice augmentation where the leaf "device" hints a relationship between the "subservice" concept and the "device" concept.

Commenté [MB3]: To what?

```
     module: ietf-service-assurance-device

       augment /sain:subservices/sain:subservice/sain:parameter:
         +--rw parameters
           +--rw device    string
```

Figure 1: YANG tree diagram of a subservice augmentation.

The extraction of this hidden knowledge from YANG models would enable the integration of YANG data silos at a conceptual level, regardless of the physical implementation (i.e., the YANG schema, syntax, and encoding format). In this regard, the knowledge graph is a promising

Commenté [MB4]: I would insist on this (operation knowledge, operator savor-faire, etc.)

Commenté [MB5]: Consider adding a reference technology that can be used to link data silos based on common concepts like "device" that are captured in ontologies. ~~Besides~~Also, by transforming ~~the~~ a YANG data into a graph structure, the relationships between data silos are represented as first class citizens in the graph instead of "foreign keys" where the relationship is made implicit. This document provides guidelines for building a knowledge graph for data sources based on the YANG language.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.1. Terminology

This document defines the following terms:

Data materialization: A ~~Technique~~ technique that collects data from ~~remote~~ a data source and persists a copy of the data in a ~~target~~ specific data storage. This process can also be seen as Extract-Transform-Load (ETL).

Data virtualization: A ~~Technique~~ technique wherein an intermediate component (i.e., data virtualization layer) exposes data available in a ~~remote~~ data source~~s~~ without creating ~~an~~ a copy of the data. The data virtualization layer keeps pointers to the original location of data, so that when a data consumer asks for these data, the virtualization layer ~~collects~~ retrieves the data from the source and directly serves the data to the consumer.

Ontology: Formal, shared representation of knowledge in a domain.

2.2. Acronyms

CQ: Competency Question

ETL: Extract-Transform-Load

KG: Knowledge Graph

KGC: Knowledge Graph Construction

LOT: Linked Open Terms

LPG: Labelled Property Graph

OWL: Web Ontology Language

RDF: Resource Description Framework

Commenté [MB6]: ??

a mis en forme : Surlignage

Commenté [MB7]: Is this useful?

a mis en forme : Surlignage

Commenté [MB8]: Do we have a ref to cite here?

RDFS: RDF Schema

RML: RDF Mapping Language

SAREF: Smart Applications REFerence

SHACL: Shapes Constraint Language

W3C: World Wide Web Consortium

3. A ~~Bief~~ Brief Introduction to Knowledge Graphs

3.1. What is a Knowledge Graph?

A knowledge graph contains a collection of facts alongside what it is known ~~we~~ about them and represents following a graph structure. Knowledge graphs enable a contextualized understanding of data as the data (e.g., the individuals, instance) travel with the meaning of the data themselves (e.g., the concepts, knowledge). For example, a knowledge graph may contain data about an interface "eth0", but also, that an interface can be physical or virtual, belongs to a network device, and has a name, description, and an MTU.

Commenté [MB9]: Examples?

a mis en forme : Surlignage

To this end, knowledge graphs build upon on ontologies, which are explicit representations of conceptualizations in a specific ~~domains~~domain.
In other words, ontologies can be seen as representations of conceptual models following a formal logic that allows machines to understand and reason over them. In this regard, a conceptual model ~~model~~, also known as information model, may translate into different data models depending on the data source technology [RFC3444].

By mapping data models (i.e., physical level) with the concepts represented in ontologies (i.e., conceptual level), we can find heterogenous datasets scattered in the network that reference common concepts such as "interface" or "device". Based on this semantic mapping, in addition to the flexibility of the graph structure, knowledge graphs enable the integration of heterogenous data based their semantics is what knowledge graphs can deliver.

3.2. Key Graph Standards

The Resource Description Framework (RDF) [RDF] data model from the W3C Semantic Web has been considered as the standard graph data model given its maturity. For that reason, most of the knowledge graph implementations have relied upon the RDF standard and other standards from the Semantic Web like RDF Schema (RDFS) [RDFS], Ontology Language (OWL) [OWL], Shapes Constraint Language (SHACL) [SHACL], and SPARQL [SPARQL].

However, the late success of graph databases like Neo4j have proved the Labelled Property Graph (LPG) data model as an alternative for implementing knowledge graphs. Aiming to bridge the gap between these two graph data models, the W3C RDF-Star working group is investigating evolving RDF to facilitate the representation of statement about statements.

Similarly, the ETSI ISG CIM defined the NGSI-LD standard [ETSI-GS-CIM-009], which builds upon two:

* An NGSI-LD information model which derives from the Labeled Property Graph (LPG) model and grounds on the RDF for a semantic annotation of the data in the graph.

* The NGSI-LD API, which defines a REST API for building and interacting with the graph.

3.3. Knowledge Objects

The intrinsic nature of knowledge graphs is to connect as much knowledge as possible within certain scope---time and/or space. However, not all processes and operations require whole knowledge graphs. For instance, the communication of a piece of telemetry data, organized according to NTF [RFC9232], can be ~~repreented~~represented as a subset of the knowledge graph of all measurements.

A knowledge object, as defined in [EERVC], consists in a knowledge graph subset of an arbitrary size---from single atoms to tens or hundreds of triples---that is decorated with metadata to facilitate its contextualization.

Knowledge objects are particularly well suited to enable entities that work with knowledge graphs to communicate to each other knowledge pieces, obtained from their knowledge graphs or newly created from other sources, such as monitoring. It has been demonstrated in [SECDEP].

4. Knowledge Graph Construction (KGC)

The construction of a knowledge graph can be divided into two main activities: ontology development (Section 4.1) and knowledge graph construction pipeline (Section 4.2).

4.1. Ontology Development

Ontologies provide the formal representation of the conceptual models that capture the semantics of data, and building on this, the integration of data in the knowledge graph. Ontologies can be developed following different techniques, ranging from manual to fully automated, depending on the characteristics of the data to be integrated in the knowledge graph (e.g., format or schema).

4.1.1. Automatic knowledge Extraction from YANG Models

The extraction of knowledge from YANG models can be automated, in particular, by analyzing YANG identities to generate controlled vocabularies and taxonomies.

[RFC7950] defines a YANG identity as "globally unique, abstract, and untyped identity", therefore, a relation between a YANG identity and a concept is straightforward. Additionally, YANG identities can inherit from other YANG identities via the "base" statement. These ideas align with the notion of a taxonomy, where concepts are hierarchically linked with other concepts.

To support the creation of knowledge structures like taxonomies or thesauri, the W3C standardized the Simple Knowledge Organization System (SKOS). In such ontology, a concept scheme comprises a set of concepts that can be linked with other concepts via hierarchical and associative relations. Typically, a YANG model containing YANG identities can be represented as an instance of the "skos:ConceptScheme" class. Next, all YANG identities included in a YANG model can be represented as "skos:Concept instances" that are contained in the concept scheme. Lastly, those YANG identities that include the "base" statement, the respective SKOS concept will include a relation "skos:broader" whose range is the SKOS concept representing the parent YANG identity.

4.1.2. Standard Development Methodologies

Automating the extraction of all the knowledge from YANG modeled data~~s~~ is impossible, and therefore, manual intervention from domain experts is required. To ease this process, a recommended practice is to develop the ontology by following a standard methodology like Linked Open Terms (LOT) [Poveda-Villalon2022].

Commenté [MB10]: Where? Any authoritative ref to cite?

LOT is an ontology development methodology that adopts best practices from agile software development. The methodology has been widely used in European projects as well as in the creation of the ETSI SAREF ontology and its extensions. Precisely, with SAREF Ontology ETSI tackled a similar problem in the scope of IoT, where there is a heterogeneous variety of standard data models and protocols. The methodology iterates over a workflow of the following four activities:

1. ontology requirements specification

2. ontology implementation

3. ontology publication, and

4. ontology maintenance.

The workflow starts with the specification of requirements that the ontology must fulfill. To that aim, the methodology requires collecting knowledge from domain experts, but also by analyzing the data sources (e.g., network devices) and schemas for the data (e.g., YANG models) to be ingested and integrated in the knowledge graph. LOT recommends several approaches such as competency questions (CQs), natural language statements, or tabular information inspired by METHONTOLOGY.

4.2. Knowledge Graph Construction Pipeline

The construction of a knowledge graph is supported by a data pipeline that follows the archetypical Extract-Transform-Load (ETL), wherein the raw data is collected from the source(s), transformed, and finally, stored for consumption. The knowledge graph creation

pipeline can thus be split into multiple steps as depicted in
Figure 2.

```
    +------------+       +----------+       +------------------+
    |            |       |          |       |                  |
    | Ingestion  +------>| Mapping  +------>| Materialization  |
    |            | Raw   |          | RDF   |                  |
    +------------+ data  +----------+ data  +---------+--------+
          ^      (YANG)                               |
     Raw  |                                           | RDF
     data |                                           | data
    (YANG)|                                           |
          |                                           v
    +-----+----+                                +-----------+
    |   Data   |                                | Knowledge |
    |  Source  |                                |   Graph   |
    | (device) |                                +-----------+
    +----------+
```

Figure 2: High-level architecture of a Knowledge Graph Construction Pipeline

These steps are the following: ingestion, mapping, and materialization.

4.2.1. Ingestion

Represents the first step in the creation of the knowledge graph. This step is realized by means of collectors that ingest raw data from the selected data source. These collectors implement data access protocols which are specific to the technology and type of the data source. When it comes to network management protocols based on YANG, these protocols can be NETCONF [RFC6241], RESTCONF [RFC8040], and gNMI [GNMI].

Two main types of data sources are identified based on the techniques used to ingest the data, namely, batch and streaming. In the case of batch data sources data are pulled (once or periodically) from the data source. This could be represented by queries sent to a ~~YANG~~ server like an SDN controller to fetch the network topology [RFC8345].

Commenté [MB11]: There is no such concept

Regarding streaming data sources, the collector subscribes to a YANG server to receive notifications of YANG data periodically or upon changes in the data source (e.g., a network device whose interface goes down). These subscriptions can be realized, either based on configuration or dynamically, using mechanisms like YANG Push [RFC8641]. But additionally, another common scenario is the use of message broker systems like Apache Kafka for decoupling the ingestion of streams of YANG data [I-D.~~netana~~ietf-nmop-yang-message-broker-integration]. Hence, knowledge graph collectors could also support the ingestion of YANG data from these kinds of message brokers.

4.2.2. Mapping

This second step consists at receiving the raw data ~~data~~ from the

Ingestion step. Here, the raw data is mapped to the concepts capture in one or more ontologies. By applying these mapping rules, the raw data is semantically annotated and transformed into RDF data. These mappings can be declared using declarative languages like RDF Mapping Language (RML) [Iglesias-Molina2023].

RML is a declarative language that is currently being standardized within the W3C Knowledge Graph Construction Community group [W3C-KGC] that allows for defining mappings rules for raw data encoded in semi-structured formats like XML or JSON. The benefits of using a declarative language like RML are twofold:

i) the engine that implements the RML rules is generic, thus the mappings rules are decoupled from the code;

ii) the explicit representation of mapping and transformation rules as part of the knowledge graph provides data lineage insights that can greatly improve data quality and the troubleshooting of data pipelines.

RML is making progress towards becoming a standard, but support of additional YANG encoding formats like CBOR [RFC8949] or Protobuf remains a challenge. The knowledge payload carried ~~by~~ using CBOR and/or Protobuf is organized as knowledge objects transmitted by the mapping entities and received by the materialization entities. The use of knowledge objects allows them to easily "cut" knowledge graphs into smaller pieces, transmit them, and "paste" and/or "glue" the pieces onto the destination knowledge graph. Consistency is retained by making the same ontologies be used with the particular knowledge objects.

a mis en forme : Surlignage

Commenté [MB12]: Is CBOR supported in implems?

4.2.3. Materialization

This is the final step of the knowledge graph creation. This step receives as an input the knowledge object that contains RDF data generated in the Mapping step, which has easily manageable semantic triples---or quadruples---, as well as metadata to contextualize them and ~~facilicate~~facilitate the incorporation of the ~~knwoledge~~knowledge to the local knowledge graph storage element. At this point, the RDF data can be sent to an RDF triple store like Apache Jena Fuseki [Fuseki] for consumption via SPARQL. But alternatively, this step may transform the RDF data into an LPG structure and store the resulting data in a graph database like Neoj4 [Neo4j]. Similarly, the RDF data could also be transformed into the ETSI NGSI-LD standard and stored in an NGSI-LD Context Broker.

5. Sample Knowledge Graph Applications

Network performance KPIs: The integration of data at different abstraction levels ~~of abstraction~~ in the network can facilitate the computation of network performance KPIs, such as throughput or packet loss ratio. By integrating data silos such as the network topology with the status of network interfaces, a network analytics application could ask the knowledge graph to compute the

Commenté [MB13]: Are there public pointer where such app have been demoed?

throughput or packets loss ratio at a specific link in the
network.

Anomaly detection and incident management: Projects like NORIA
[Tailhardat2023] have demonstrated how knowledge graphs can help
in the detection of anomalies in network systems. This approach
links data pertaining to different data silos like network
infrastructure, logs, alarms, and ticketing. In another example,
the combination network topology data with data about network
interface status, consumers of the knowledge graph can detect
network anomalies like link fault because ~~an~~ a network interface
has
been unexpectedly disabled but it was configured to be enabled.

Service assurance: A knowledge graph can enable the implementation
of the service assurance for intent-based networking architecture
defined in [RFC9417]. Precisely, this architecture, and the
companion YANG data models from RFC 9418, define an assurance
graph where dependencies among network services and their
associated health and symptoms are captured. All these data,
which can be further linked with other data silos like network
topology or network interface status, can be naturally integrated
and represented in a knowledge graph.

Network digital twin: Knowledge graphs are considered promising
candidates for the realization of network digital twins
[I-D.irtf-nmrg-network-digital-twin-arch]. Particularly, the
benefits of using knowledge graphs to construct ~~netwokr~~network
digital
twins dedicated to ~~he~~ the detection of network faults is
demonstrated
in [ANSA]. Services of such type benefit from the ability to
integrate heterogenous silos of data, in combination with the
explicit representation of the semantics of the data; making the
knowledge graph a powerful technology for building and connecting
multiple network digital twins. In addition, the representation
of concepts by means of ontologies, produces abstract
representations of network digital twins, regardless of the
complexities of the underlying technologies. For instance, an
abstract representation of a network topology Digital Map
[I-D.~~havel~~ietf-nmop-digital-map-concept] in the knowledge graph can
be
translated into a descriptor or data model that is specific to the
technology used (e.g., KNE, ContainerLab, or OSM).

Evolution of YANG Catalog: The flexibility and extensibility of
knowledge graphs have made them a popular choice for implementing
data catalogs. The purpose of a data catalog is to provide
consumers with a registry of datasets exposed by data sources
where to find data of interest. Additionally, these datasets can
be linked to the (business) concepts that they refer to, so that
consumers can search for datasets based on relevant concepts such
as “interface”. Taking inspiration from these implementations, and
building on a knowledge graph, the YANG Catalog could evolve
towards a data catalog, where the YANG modules represent those
datasets of interest. The dependencies between YANG models
(import, deviations, augments) can be naturally represented in the
knowledge graph. In turn, these YANG models can be linked with

concepts that are represented in ontologies. Additionally, these YANG models, can be combined with the implementation details of network devices yang lib augment [I-D.lincla-netconf-yang-library-augmentation] that could be part of an inventory [I-D.ietf-ivy-network-inventory-yang].

Contextualized telemetry data: Having context of how YANG telemetry data [I-D.ietf-opsawg-collected-data-manifest] is being collected can improve the understanding of the data for network analytics or closed-loop automation. Knowledge graphs can help in this task by linking the collected data with: (i) the metadata that characterizes the platform producing the data; and (ii) the metadata that characterizes how and when the data were metered. As shown in [TKDP], the application of both knowledge graphs and knowledge objects to the management of telemetry data facilitates reducing the level of coupling of multiple tasks and overall operations, which enables the resolution of current network problems through the collaboration of different ~~stakeholers~~stakeholders.

Artificial intelligence particularization to network management: The application of knowledge graphs to telemetry data is particularly necessary for applying artificial intelligence (AI) methods to network management, as supported by [AINEMA]. Multiple AI elements can interoperate coherently when they make use of the same ontology, they manage several sub-graphs of a bigger knowledge graph, and they use knowledge objects to communicate knowledge-based messages to each other. The benefit is demonstrated in typical network scenarios, as discussed in [EERVC].

6. Challenges

Ontology development: A ~~Time~~time-consuming task that requires skills in knowledge management and conceptual modeling. Additionally, ontology developers should maintain a tight coordination with domain owners and ontology users. Following a standard methodology like LOT provides guidance in the process but still, the development of the ontology requires manual work. Tools that can produce or bootstrap ontologies from existing YANG data models in a semi-automatic, or even automatic, are desirable. In this sense, the future release of the YANG language could be extended to facilitate this task at design time. YANG data models could include explicit semantics in the data models, in the same way that JSON-LD [JSON-LD] or CSVW [CSVW] include metadata indicating which concepts from concepts are referenced by the data. In the current version of YANG, this could be achieved at runtime using the YANG Metadata extension [RFC7952]. With this extension, YANG data models could include additional metadata to indicate the ontology concept a YANG data node is referring to, though this approach only works at runtime, and additionally, it would require augmenting existing YANG data models.

Pipeline performance: To integrate the raw data from ~~the~~ an original source into the knowledge graph entails several steps as described before. ~~This~~ These steps add an extra latency before having the data

Commenté [MB14]: I guess this may not be important for some applications stored in the knowledge graph for consumption. This latency can be an important limitation for real-time analytics use cases.

Scalability: The knowledge graph must be able to integrate massive amounts of data collected from the network. Distributed and federated architectures can improve the scalability of a global, composable knowledge graph. However, these architectures add complexity to the management of knowledge graph as well as extra latency when federating requests.

Virtualization: The common approach for data integration is by materializing the data in the knowledge graph, which entails duplicating the data. However, this approach presents multiple limitations in terms of data governance and data cadence. Regarding data governance, having copies of the original data hampers keeping track of all the available data. With respect to data cadence, in particular for batch data sources, data are periodically pulled from the source at particular frequency, which might not be optimal depending on the use case. In this sense, data virtualization introduces a new data access technique that can overcome these limitations. With this technique, the knowledge graph defines pointers to the data at the original source, and the KGC pipeline performs the ingestion and mapping of the data, and eventually the delivery of data to the consumer, only when requested on demand.

Network configuration: This document has focused on integrating telemetry data in the knowledge graph for monitoring purposes. But knowledge graphs could also be leveraged for integrating data related to the configuration of devices and services in the network. This approach could enable closed-loop network management since both configuration and operational data are stored in the knowledge graph.

7. Security Considerations

Access control to data: The knowledge graph becomes an integrator of data, and, in many cases, sensible. Therefore, data access control mechanisms must be present to ensure that only authorized consumers can discover and access data from the knowledge graph. Access control policies based on roles or attributes are common approaches, but additional aspects like sensitivity of data could be included in the policy.

Integrity and authenticity of mappings: The declaration of mappings of raw data to concepts in ontologies is a critical step in the knowledge graph construction. Unauthorized mappings, or even tampered mappings, can lead to security breaches and anomalies producing a great impact on analytics and machine learning applications that consume data from the knowledge graph. To protect consumers from these scenarios, the knowledge graph must include mechanisms that verify the correctness, authenticity, and integrity of the mappings used in the construction of the graph. Only data owners, as accountable of their data, should be authorized to define and deploy mappings for the knowledge graph construction.

Data provenance: Keeping track of the history of data as they go through the knowledge graph construction pipeline can improve the quality of the data of the knowledge graph. As part of the knowledge graph construction, signatures can be appended to the data [I-D.lopez-opsawg-yang-provenance], can help in verifying that such data come from the golden data source, and therefore, that the data can be trusted.

8. IANA Considerations

This document has no IANA actions.

9. Open Issues

* Should RML mappings reference data at the YANG level using XPath or subtree filters? Or references should remain based on the actual encoding format used by the network management protocol, e.g., JSON, XML.

* Should this document provide guidelines for generating URIs of nodes/subjects in the knowledge graph? Take into account there are several levels of abstraction device vs network/service level. For example, the URI that identifies a network interface cannot be generated only from the name of the interface as there could conflicts with other interfaces of other network devices having the same name.

* Definition of YANG data sources with formal vocabulary, similar to what Web of Things ontology has done for MQTT or REST APIs or D2RQ ontology for relational databases. Having the specification of the data source in the knowledge graph improves provenance and decouples the configuration from the implementation, e.g., via custom INI config file.

* Implementations? References to examples based on open-source implementations. Integration with YANG-Push-Kafka architecture. Target future hackathons.

* Document focused on YANG data sources. Should the document open the scope to other kinds of data sources like IPFIX?