Network Management Operations
Internet-Draft
Intended status: Informational
Expires: 30 August 2025

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

26 February 2025

Knowledge Graph Construction from Network Data Sources
draft-marcas-nmop-kg-construct-00

Abstract

This document discusses the mechanisms that support the management and creation of knowledge graphs from data sources specific to the network management domain. The document provides background on core aspects such as ontology development, identifies methodologies and standards, and shares guidelines for integrating network data sources.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://idomingu.github.io/knowledge-graph-yang/draft-marcas-knowledge-graph-yang.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-marcas-nmop-kg-construct/.

Discussion of this document takes place on the Network Management Operations Working Group mailing list (mailto:nmop@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/nmop/. Subscribe at https://www.ietf.org/mailman/listinfo/nmop/.

Source for this draft and an issue tracker can be found at https://github.com/idomingu/knowledge-graph-yang.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

Knowledge graph introduces a new paradigm in data management that facilitates the integration of heterogenous data silos thanks to a semantic layer. In the case of network management, knowledge graphs provide a data integration solution that can cope with the diverse network data sources and telemetry mechanisms [I-D.mackey-nmop-kg-for-netops].

The construction of knowledge graphs is a challenging activity that requires the combination of skills in semantic modelling and data engineering. Semantic data models are represented by ontologies and other forms of structured knowledge, which must be kept in sync with the data pipelines that integrate the different data silos into the knowlede graph. The data integration process is based on the ingestion of raw data from their data sources, the mapping of the raw data to the respective ontologies, and the transformation of the data into a graph structure semantically-annotated.

In this sense, Knowledge Graph Construction (KGC) underpinned by two pillars: i) ontology development; and ii) knowledge graph construction pipeline. These pillars are described in detail in the following sections.

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 integration: Process of combining data from diverse sources into a unified view.

Data mapping: Technique that defines how data from one data model corresponds to another data model.

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

Data virtualization: Technique wherein an intermediate component (i.e., data virtualization layer) exposes data available in a remote data sources without creating an copy of the data. The data virtualization layer keeps pointers to the original location of data, so when a data consumer asks for these data, the virtualization layer collects 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

OWL: Web Ontology Language

RDF: Resource Description Framework

RDFS: RDF Schema

RML: RDF Mapping Language

SAREF: Smart Applications REFerence

SHACL: Shapes Constraint Language

W3C: World Wide Web Consortium

3. 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).

3.1. Standard Development Methodologies

Developing an ontology is a challenging task that requires skills in knowledge management and semantic modelling. To ease this process, a good practice is to follow mature, proven methodologies that provide thorough guidelines and recommend tools that can help in the development of an ontology. An example of these methodologies is Linked Open Terms (LOT) [Poveda-Villalon2022].

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 data 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.

3.2. Automatic Knowledge Extraction from YANG Models

The extraction of knowledge from YANG models could be automated, for example, 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. Knowledge Graph Construction Pipeline

4.1. 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 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].

4.2. Pipeline Steps

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 1.

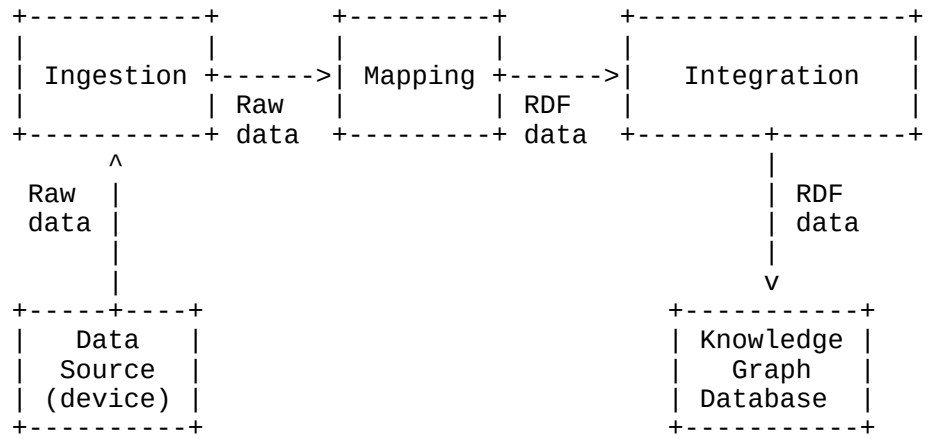

Figure 1: High-level architecture of a Knowledge Graph Construction Pipeline

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

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. For instance, 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.

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-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 captured in one or more ontologies. By applying these mapping rules, the raw data is semantically annotated and transformed into RDF data. Depending on the nature of the raw data, different techniques can applied.

In the case of (semi-)structured data such as tabular data (e.g., CSV, relational databases) or hierarchical data (e.g., JSON, XML) these mappings can be defined by 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 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.

4.2.3. Integration

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 the incorporation of the knwoledge 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 [ETSI-GS-CIM-009] and stored in an NGSI-LD Context Broker.

5. Challenges

Ontology development: 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 data models in a semi-automatic, or even automatic, are desirable. In this sense, 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.

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

6. 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.

7. IANA Considerations

This document has no IANA actions.

8. Open Issues

* 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.

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

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

9.2. Informative References

[ANSA] Pedro Martinez-Julia, Ved P. Kafle, Hitoshi Asaeda., "Application of Category Theory to Network Service Fault Detection. IEEE Open Journal of the Communications Society 5 (2024): 4417-4443.", n.d..

[CSVW] "CSVW - CSV on the Web", n.d., <https://csvw.org>.

[EERVC] Pedro Martinez-Julia, Ved P. Kafle, Hiroaki Harai., "Exploiting External Events for Resource Adaptation in Virtual Computer and Network Systems, IEEE Transactions on Network and Service Management 15 (2018): 555-566.", n.d..

[ETSI-GS-CIM-009] "Context Information Management (CIM); NGSI-LD API", March 2024, <https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.08.01_60/gs_CIM009v010801p.pdf>.

[Fuseki] Apache, "Apache Jena Fuseki", n.d., <https://jena.apache.org/documentation/fuseki2/>.

[GNMI] OpenConfig, "gRPC Network Management Interface (gNMI)", n.d., <https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md>.

[I-D.ietf-ivy-network-inventory-yang] Yu, C., Belotti, S., Bouquier, J., Peruzzini, F., and P. Bedard, "A Base YANG Data Model for Network Inventory", Work in Progress, Internet-Draft, draft-ietf-ivy-network-inventory-yang-04, 5 November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-ivy-network-inventory-yang-04>.

[I-D.ietf-netconf-yang-library-augmentedby]
Lin, Z., Claise, B., and I. D. Martinez-Casanueva, "Augmented-by Addition into the IETF-YANG-Library", Work in Progress, Internet-Draft, draft-ietf-netconf-yang-library-augmentedby-01, 21 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-yang-library-augmentedby-01>.

[I-D.lopez-opsawg-yang-provenance]
Lopez, D., Pastor, A., Feng, A. H., Birkholz, H., and S. Garcia, "Applying COSE Signatures for YANG Data Provenance", Work in Progress, Internet-Draft, draft-lopez-opsawg-yang-provenance-04, 5 January 2025, <https://datatracker.ietf.org/doc/html/draft-lopez-opsawg-yang-provenance-04>.

[I-D.mackey-nmop-kg-for-netops]
Mackey, M., Claise, B., Graf, T., Keller, H., Voyer, D., and P. Lucente, "Knowledge Graph Framework for Network Operations", Work in Progress, Internet-Draft, draft-mackey-nmop-kg-for-netops-01, 21 October 2024, <https://datatracker.ietf.org/doc/html/draft-mackey-nmop-kg-for-netops-01>.

[I-D.netana-nmop-yang-message-broker-integration]
Graf, T. and A. Elhassany, "An Architecture for YANG-Push to Message Broker Integration", Work in Progress, Internet-Draft, draft-netana-nmop-yang-message-broker-integration-00, 22 April 2024, <https://datatracker.ietf.org/doc/html/draft-netana-nmop-yang-message-broker-integration-00>.

[Iglesias-Molina2023]
Iglesias-Molina, A., "The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF", The Semantic Web ISWC 2023 , October 2023, <https://doi.org/10.1007/978-3-031-47243-5_9>.

[JSON-LD] W3C, "JSON-LD 1.1: A JSON-based Serialization for Linked Data", July 2020, <https://www.w3.org/TR/json-ld11/>.

[Neo4j] "rdflib-neo4j - RDFLib Store backed by neo4j", n.d., <https://github.com/neo4j-labs/rdflib-neo4j>.

[Poveda-Villalon2022]
Engineering Applications of Artificial Intelligence, "LOT: An industrial oriented ontology engineering framework", May 2022, <https://doi.org/10.1016/j.engappai.2022.104755>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/rfc/rfc6241>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/rfc/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/rfc8040>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/rfc/rfc8345>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/rfc/rfc8641>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9232] Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", RFC 9232, DOI 10.17487/RFC9232, May 2022, <https://www.rfc-editor.org/rfc/rfc9232>.

[SECDEP] Ana Hermosilla, Jose Manuel Manjn-Cliz, Pedro Martinez-Julia, Antonio Pastor, Jordi Ortiz, Diego R. Lopez, Antonio Skarmeta., "Secure deployment of third-party applications over 5G-NFV ML-empowered infrastructures, the 7th International Conference on Mobile Internet Security (MobiSec ’23), Dec 19-21, 2023, Okinawa, Japan.", n.d..

[TKDP] Pedro Martinez-Julia, Ved P. Kafle, Hitoshi Asaeda., "Telemetry Knowledge Distributed Processing for Network Digital Twins and Network Resilience. NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium (2023): 1-6.", n.d..

[W3C-KGC] W3C, "Knowledge Graph Construction Community Group", n.d., <https://www.w3.org/community/kg-construct/>.

Appendix A. NETCONF Data Sources

This appendix presents a scenario that demonstrates the construction of a knowledge graph based on YANG data collected from a NETCONF server. In particular, the scenario tackles the creation of a data catalog based on a knowledge graph that keeps registry of the YANG data sources and the YANG data models that they implement.

As described in [I-D.ietf-netconf-yang-library-augmentedby], data catalog implementations backed by knowledge graphs provide powerful solutions that can easily incorporate additional context to the catalog. As an evolution of the YANG Catalog service, the resulting knowledge graph facilitates the navigation across dependencies of YANG modules, but more importantly, enables the combination of these data with other data silos such as the network topology [RFC8345] or network hardware inventory [I-D.ietf-ivy-network-inventory-yang].

To create a knowledge graph that supports the data catalog, the proposed approach is based on collecting data from the YANG Library from devices running in the network, in this case, from a NETCONF server. For this, the RML engine queries the NETCONF server to retrieve the YANG Library data, and then, applies the RML mappings to transform the YANG data into RDF according to the target ontology.

This prototype was conducted as part of the paper "Declarative Construction of Knowledge Graphs from NETCONF Data Sources" sent to the Semantic Web Journal (currently under review): https://www.semantic-web-journal.net/content/declarative-construction-knowledge-graphs-netconf-data-sources-0

A.1. Prototype Architecture

A high-level architecture of the prototype that validates the implementation is shown below:

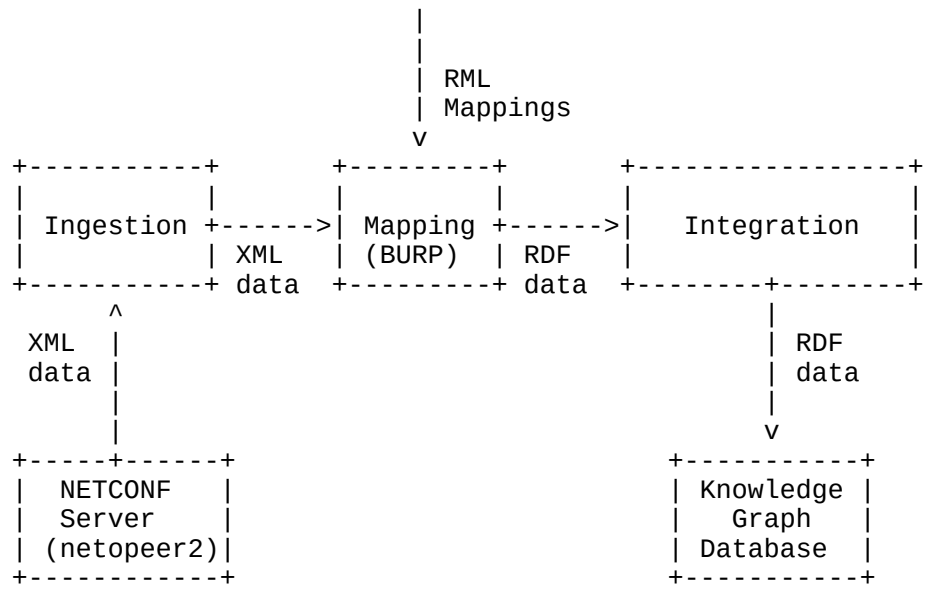

Figure 2: Architecture of prototype to construct knowledge graph from NETCONF data source

BURP was selected as the open-source implementation of an RML engine that was chosen for this prototype. The NETCONF server is emulated using the netopeer2.

A.2. Target Ontology

The YANG Library Ontology was developed to represent the implementation details of YANG module and submodules, along with their interdependencvies, in the different datastores of YANG server. The ontology was developed following the LOT methodology and is publicly available at: https://w3id.org/yang/library

The code of the ontology and all related artifacts are publicly available on GitHub: https://github.com/candil-data-fabric/yang-library-ontology

A.3. KGC Pipeline

In addition to the YANG Library Ontology, the YANG Server Ontology was developed to represent YANG data sources such as NETCONF servers and operations to retrieve data from them such as queries or subscriptions. Similarly, this ontology was developed following the LOT methodology and is publicly available at: https://w3id.org/yang/server

The code of the ontology and all related artifacts are publicly available on GitHub: https://github.com/candil-data-fabric/yang-server-ontology

The YANG Server Ontology is used in combination with the RML vocabulary to describe the access to YANG servers, from which the collected data are transformed into RDF. In this sense, BURP was extendended to support the ingestion of YANG data from NETCONF servers using NETCONF queries.

The following subsections include excerpts of the raw XML data (Figure 3), RML mappings (Figure 4), and final RDF data (Figure 3). The complete examples can be found on: https://github.com/candil-data-fabric/yang-library-ontology/tree/main/knowledge-graph/xpath

A.3.1. Raw data

```
<modules-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <module-set-id>1</module-set-id>
  <module>
    <name>ietf-yang-patch</name>
    <revision>2017-02-22</revision>
    <schema>file:///etc/sysrepo/yang/ietf-yang-patch@2017-02-22.yang</schema>
    <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-patch</namespace>
    <conformance-type>import</conformance-type>
  </module>
  <module>
    <name>ietf-ip</name>
    <revision>2018-02-22</revision>
    <schema>file:///etc/sysrepo/yang/ietf-ip@2018-02-22.yang</schema>
    <namespace>urn:ietf:params:xml:ns:yang:ietf-ip</namespace>
    <conformance-type>implement</conformance-type>
  </module>
</modules-state>
```

Figure 3: Excerpt of YANG Library data collected from a NETCONF server

A.3.2. Mappings

```
@prefix yl: <https://w3id.org/yang/library#> .
@prefix ys: <https://w3id.org/yang/server#> .
@prefix rml: <http://w3id.org/rml/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix core: <https://ontology.unifiedcyberontology.org/uco/core/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix observable: <https://ontology.unifiedcyberontology.org/uco/observable/> .
```

```
@base <https://netconf-rml-demo.org/> .

# Connection details to NETCONF server
<netconf-server-1> a ys:NetconfServer ;
    ys:socketAddress <netconf-server-1/socket-address> ;
    ys:serverAccount <netconf-server-1/account> ;
    ys:hostKeyVerification "false" ;
    ys:capability ys:XpathCapability ,
                  ys:YangLibrary1.0 .

<netconf-server-1/datastores/operational> a ys:OperationalDatastore ;
    ys:server <netconf-server-1> .

<netconf-server-1/datastores/running> a ys:RunningDatastore ;
    ys:server <netconf-server-1> .

<netconf-server-1/socket-address> a observable:SocketAddress ;
    observable:addressValue "localhost:830" .

<netconf-server-1/account> a ys:ServerAccount ;
    ys:username "netconf" ;
    core:hasFacet <netconf-server-1/account/authentication> .

<netconf-server-1/account/authentication> a observable:AccountAuthenticationFacet ;
    observable:password "netconf" ;
    observable:passwordType "plain-text" .

<filters/xpath/yang-library> a ys:XPathFilter ;
    ys:xpathValue "/yanglib:modules-state";
    ys:namespace [ a ys:Namespace ;
      ys:namespacePrefix "yanglib" ;
      ys:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-yang-library" ;
    ];
.

<#TriplesMap> a rml:TriplesMap;
  rml:logicalSource [ a rml:LogicalSource;
    rml:source [ a ys:Query, rml:Source ;
      ys:sourceDatastore <netconf-server-1/datastores/operational> ;
      ys:filter <filters/xpath/yang-library>
    ];
    rml:referenceFormulation [ a ys:NetconfQuerySource ;
      rml:namespace [ a rml:Namespace ;
        rml:namespacePrefix "yanglib" ;
        rml:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-yang-library" ;
      ];
    ];
    rml:iterator "/yanglib:modules-state/yanglib:module";
```

```
  ];
  rml:subjectMap [ a rml:SubjectMap;
    rml:template "http://example.org/module/{yanglib:name/text()}:{yanglib:revision/text(
)}";
    rml:class yl:Module;
  ];
  rml:predicateObjectMap [ a rml:PredicateObjectMap;
    rml:predicateMap [ a rml:PredicateMap;
      rml:constant yl:moduleName;
    ];
    rml:objectMap [ a rml:ObjectMap;
      rml:reference "yanglib:name/text()";
      rml:datatype xsd:string;
    ];
  ];
  rml:predicateObjectMap [ a rml:PredicateObjectMap;
    rml:predicateMap [ a rml:PredicateMap;
      rml:constant yl:revisionDate;
    ];
    rml:objectMap [ a rml:ObjectMap;
      rml:reference "yanglib:revision/text()";
      rml:datatype xsd:date;
    ];
  ];
  rml:predicateObjectMap [ a rml:PredicateObjectMap;
    rml:predicateMap [ a rml:PredicateMap;
      rml:constant yl:namespace;
    ];
    rml:objectMap [ a rml:ObjectMap;
      rml:reference "yanglib:namespace/text()";
      rml:datatype xsd:anyURI;
    ];
  ];
.
```

Figure 4: RML mappings that collect YANG Library from a NETCONF server and map them to the YANG Library Ontology

A.3.3. RDF data

```
<http://example.org/module/ietf-ip:2018-02-22>
  <https://w3id.org/yang/library#moduleName>
  "ietf-ip"^^<http://www.w3.org/2001/XMLSchema#string> .
<http://example.org/module/ietf-ip:2018-02-22>
  <https://w3id.org/yang/library#revisionDate>
  "2018-02-22"^^<http://www.w3.org/2001/XMLSchema#date> .
<http://example.org/module/ietf-ip:2018-02-22>
  <https://w3id.org/yang/library#namespace>
  "urn:ietf:params:xml:ns:yang:ietf-ip"^^<http://www.w3.org/2001/XMLSchema#anyURI> .
<http://example.org/module/ietf-ip:2018-02-22>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <https://w3id.org/yang/library#Module> .
<http://example.org/module/ietf-yang-patch:2017-02-22>
  <https://w3id.org/yang/library#moduleName>
  "ietf-yang-patch"^^<http://www.w3.org/2001/XMLSchema#string> .
<http://example.org/module/ietf-yang-patch:2017-02-22>
  <https://w3id.org/yang/library#revisionDate>
  "2017-02-22"^^<http://www.w3.org/2001/XMLSchema#date> .
<http://example.org/module/ietf-yang-patch:2017-02-22>
  <https://w3id.org/yang/library#namespace>
  "urn:ietf:params:xml:ns:yang:ietf-yang-patch"^^<http://www.w3.org/2001/XMLSchema#anyURI
> .
<http://example.org/module/ietf-yang-patch:2017-02-22>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <https://w3id.org/yang/library#Module> .
```

Figure 5: Excerpt of RDF triples generated using the RML mappings and the YANG Library data

Acknowledgments

This document is based on work partially funded by the EU Horizon Europe projects aerOS (grant agreement no. 101069732) and ROBUST-6G (grant agreement no. 101139068).

The authors would like to thank Med, Benoit, Lionel, and Thomas for their review and valuable comments.

Authors' Addresses

Ignacio Dominguez Martinez-Casanueva
Telefonica
Email: ignacio.dominguezmartinez@telefonica.com

Lucia Cabanillas
Telefonica
Email: lucia.cabanillasrodriguez@telefonica.com

Pedro Martinez-Julia
NICT
Email: pedro@nict.go.jp