NMOP
Internet-Draft
Intended status: Informational
Expires: 6 March 2026

M. Mackey
Huawei
B. Claise
Everything-Ops
T. Graf
Swisscom
H. Keller
Deutsche Telekom
D. Voyer
Bell Canada
P. Lucente
NTT
I. D. Martinez-Casanueva
Telefonica
2 September 2025

Knowledge Graph Framework for Network Operations
draft-mackey-nmop-kg-for-netops-03

Abstract

This document describes some of the problems in modern operations and management systems and how knowledge graphs and RDF can be used to solve closed loop system, in an automatic way.

Discussion Venues

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

Source for this draft and an issue tracker can be found at https://github.com/mike-mackey.

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 6 March 2026.

Table of Contents

1. Introduction

The IAB organized a workshop in June 2002 to establish a dialog between network operators and protocol developers, to guide IETF when working on network management protocols and data models. The outcome of that workshop was documented in the "Overview of the 2002 IAB Network Management Workshop" [RFC3535] which identified 14 operator requirements for consideration in future network management protocol design and related data models, along with some recommendations for the IETF.

The RFC3535 requirements were instrumental in developing first the NETCONF protocol (in the NETCONF Working Group) [RFC6241], the associated YANG data modelling language (in the NETMOD Working Group) [RFC7950], RESTCONF [RFC8040], and most recently CORECONF [I-D.ietf-core-comi].

A new IAB workshop, Next Era of Network Management Operations (NEMOPS), is getting organized to tackle the next big challenges in the world of network management. Exactly like the previous workshops, operator challenges and requirements will be documented. The new set of requirements will hopefully guide the Operational and Management Area (OPS) https://datatracker.ietf.org/group/ops/about/ future directions.

This document describes the challenges in network operations, and proposes a new framework based on knowledge graph, to solve (some of) those operational challenges, mainly how to automatically assure networks.

As an introduction, let's review the difference between information model and data model. Quoting RFC 3444 [RFC3444], "The main purpose of an information model is to model managed objects at a conceptual level, independent of any specific implementations or protocols used to transport the data. The degree of specificity (or detail) of the abstractions defined in the information model depends on the modelling needs of its designers. In order to make the overall design as clear as possible, an information model should hide all

protocol and implementation details. Another important characteristic of an information model is that it defines relationships between managed objects."

An information model, typically expressed in a language such as Unified modelling Language (UML) do not generate the full APIs, as it lacks some of the implementation- and protocol-specific details; for example, rules that explain how to map managed objects onto lower-level protocol constructs.

A data model, on the other end, can directly be used for network automation. As an example, YANG data models [RFC7950] can generate APIs, to be accessed by protocols such as NETCONF and RESTCONF.

2. Challenges

This section covers the current operational challenges.

2.1. Data Overload from Network Operations

Modern network operators are inundated with vast amounts of data generated from various sources within the network. This data encompasses information from the management plane, control plane, and data plane. Each contributing to a massive influx of information. The sheer volume of data is staggering, making it challenging even for advanced computer systems to process and analyze effectively.

2.2. Difficulties in Data Analysis and Insight Extraction

Data analysts with network domain knowledge play a crucial role in leveraging this data to predict faults, perform Root Cause Analysis (RCA), and implement automatic remediation. However, they often struggle to extract useful information due to the overwhelming volume, data standardization and complexity of the data. The challenge lies not only in processing the data but also in finding meaningful patterns and correlations that can derive actionable insights.

2.3. Complex Data Correlation Requirements

A significant challenge in modern network operations is the need to correlate data from different planes - management, control, and data. Each plane generates its own set of data, often stored in disparate repositories and formats. Linking this information together is essential to gain a holistic view of network operations but is inherently difficult.

2.4. Service and Customer Correlation

Ultimately, all collected data must be correlated back to specific connectivity services (and its customers). This correlation is critical for understanding the impact of network events on service quality and customer experience. However, the process of linking management plane data with control plane and data plane information, is to provide a coherent connectivity service with customer perspective is extremely challenging. Even as a simpler challenge, it’s not always easy to correlate the configuration management plane information with the streamed operational data: YANG, as a data modelling
language simplifies the situation, if used for both config and streaming but some extra correlation might anyway be required beyond the data model.

2.5. Data Storage and Format Disparities

Data is frequently stored in multiple repositories, each potentially using different formats. This fragmentation makes it difficult to manage the links between different data sets. In some cases, these links, relationships, may be lost within the engines of the management and analytics systems, leading to incomplete or incorrect analyses.

For example, data is stored using a variety of data model languages, sometimes different schemas for a specific data model language (for example YANG), which are sometimes different per router vendors, often siloed in different applications and storage platforms. Establishing relationships between this data, especially when disconnected from the service context and original intent, is exceedingly difficult. This fragmentation hinders the ability to maintain a cohesive understanding of network operations and their impacts on service quality.

2.6. Contextual Understanding and Relationship Mapping

To reduce the problem space and facilitate automated decision-making, it is essential to understand the context and semantic of the data, the relationships between different data sets, and how this data relates to the overall network. By developing a clear understanding of these relationships, network operators can make more informed and quicker decisions, which is crucial for achieving autonomous operations.

2.7. Loss of Context in Data Collection

The context of the collected data is often lost, complicating the task of network monitoring and analysis. For instance, it can be challenging to determine what a particular interface represents (in other words, its role): is it a Provider Edge (PE), Provider (P), Customer Edge (CE), or an interface on an Autonomous System Boundary Router (ASBR) (or ABR)? Based on this particular context, the intent is obviously different, and, as a consequence, this context is key to interpret the collected data. For example, the IP addresses observed on CE router, PE router, or P router have different context (and on the PE router, it actually depends if we refer to CE-facing or PE-facing interface). As a different example, understanding whether a link serves as a primary connection for certain customers or a backup for others is critical but frequently ambiguous.

2.8. Data Collection Methods and Interpretation

The methods and intervals at which data is collected also vary, which contributes in adding another layer of complexity. Data might be sampled within specific time windows, on-change, on-demand, or periodically. It might represent an aggregation or calculation of other values. Understanding how the router or analytics engine computes this data is vital for accurate analysis and troubleshooting.

2.9. Organizational Silos

In many organizations, network configuration and operations teams function as separate entities. This division can lead to a disconnect where the rationale behind network changes is lost or miscommunicated between teams. This lack of cohesion can further complicate data correlation and analysis efforts.

2.10. Multiple Sources of Truths

What is the network intent? Is it owned in the controller, or the current network is the network intent? What if the network is configured at the same time from the controller and from the CLI (for quick network anomaly resolution), does it imply that the network intent is partially in the controller, and partially in the network state? There are actually multiple sources of truth in networking:

* the controller configuration (intended state)

* the network itself (applied state, described by the Digital Map)

* the inventory, typically stored across different systems, with a different ID (sometimes UUID [RFC7950])

* the IP Address Management (IPAM) is another other source of truth

* etc.

2.11. Machine Readable Knowledge

While we mentioned multiple data sources, with different data modelling languages, the requirement is to have one data sources in a machine readable way, with the ability to correlate and link information.
Note that, sometimes, the modelling language is simply not existent, as protocols such as BMP or BGP-LS directly stream PDUs.

3. IETF Initiatives

To help with the different challenges mentioned in the previous section, the IETF standardized some RFCs, while some IETF drafts are currently being worked on:

TO DO: once we have the exact challenge tags, we are going to refer to those

* Service Assurance for Intent Based Networking [RFC9417] [RFC9418]

* Network Telemetry framework [RFC9232] explains the different telemetry mechanisms

* draft-ietf-nmop-yang-message-broker-integration-03 (https://datatracker.ietf.org/doc/draft-ietf-nmop-yang-message-broker-integration/)
specifies an Architecture for YANG-Push to Message Broker Integration is helping with the data collection aspects.

* draft-ietf-opsawg-collected-data-manifest (https://datatracker.ietf.org/doc/draft-ietf-opsawg-collected-data-manifest/) documents the metadata that ensure that the streaming collected data can be interpreted correctly.

* draft-ietf-nmop-network-anomaly-architecture (https://datatracker.ietf.org/doc/draft-ietf-nmop-network-anomaly-architecture/) defines an architecture for detecting anomalies in the network

* The basic concepts of the Digital Map are mentioned in draft-havel-nmop-digital-map-concept (https://datatracker.ietf.org/doc/draft-havel-nmop-digital-map-concept/)

These are building blocks, to help towards the goals of autonomous networking. However, these building blocks are not sufficient.

4. The Difficult and Costly Data Models Integration with Different Silos Protocol & Data Models

4.1. Understanding And Using Different Models In A Solution

Even excluding the vast amount of vendor specific models, the telecommunication industry is drowning in models from many different SDOs (IETF, TMF, ETSI, ONF, MEF, 3GPP). All of them fulfill a need and all of them are optional depending on the requirements of the solution/operator. Some of the models are designed to be extended, therefore even though a model is based on a standard it can deviate or add new information. This model and API soup results in confusion and in some cases (mis)interpretation that can differ per implementation.

There have been attempts to address this, to converge models and APIs towards a single model. These initiatives have largely failed. There are many reasons for this (technical debt, separation of concerns/responsibility between SDOs) but the reality is that this remains (and will likely always remain) unachievable. So what is needed is a way to understand how these different models connect and how these models were interpreted by the solution designer.

4.2. Example: Onboarding A New Device

In order to onboard a new device, what features and models that device supports must be known, how that device will map to any existing internal models for resource management e.g. https://github.com/Open-Network-Models-and-Interfaces-ONMI/TAPI, Assurance is mapped to existing assurance and collection models (data collection/message broker/TSDBs), existing health assurance pipelines (as described in [draft-ietf-nmop-network-anomaly-architecture]).

If I onboard a new device will it work with my data processing pipeline If I receive observe a problem in my service, can I trace quickly to find the related network configuration, if I decide I need to modify that network configuration do I know the values that must change at the resource API in order for it to happen.

The cost of onboarding a new device or indeed upgrading to a new software/ hardware version is buried within the different applications, the mapping code that transforms data between models, the impact on existing models (is a new instance enough or does the application schema need to be extended)

What if we could answer all of those questions by querying the connections between schemas and data. What if we could trace the connections across different applications and different design artefacts.

4.3. Different Models For Different Jobs

On the other side, with different technology domains and different protocols, come different data models. In order to assure cross domain use cases, the network management system and network operators must integrate all the technologies, protocols, and therefore data models as well. In other words, it must perform the difficult and time-consuming job of integrating & mapping information from different data models. Indeed, in some situations, there exist different ways to model the same type of information.

This problem is compounded by a large, disparate set of data sources: * MIB modules [RFC3418] for monitoring, * YANG models [RFC7950] for configuration and monitoring, * IPFIX information elements [RFC7011] for flow information, * syslog plain text [RFC3164] for fault management, * TACACS+ [RFC8907] or RADIUS [RFC2865] in the AAA (Authorization, Authentication, Accounting) world, * BGP FlowSpec [RFC5575] for BGP filter, * BMP - BGP Monitoring protocol [RFC7854] * BPG-LS for IGP monitoring * etc. or even simply the router CLI for router management.

Some networking operators still manage the configuration with CLI while they monitor the operational states with SNMP/MIBs. How difficult is that in terms of correlating information? Some others moved to NETCONF/YANG for configuration but still need to transition from SNMP/MIBs to Model-driven Telemetry.

When network operators deal with multiple data models, the task of mapping the different models is time-consuming, hence expensive, and difficult to automate.

4.4. Example: Whats An Interface ?

To make it crystal clear, let’s illustrate this with a very simple and well known networking concept: a simple interface. Let’s start with a simple CLI command: "show ip interface" for basic interface information.

* Between MIB module and YANG model, fortunately, we have the same ifIndex concept (ifIndex in MIB and if-index in YANG). This facilitates the mapping.

* In the context of IPFIX models, even if the ingressInterface and egressInterface report the famous ifIndex values within the flow record, the interface semantic changed. We have one specific field for the ingress and another one for egress traffic. The MIB object ifIndex doesn't make that distinction. While it's not difficult to map the IPFIX interface information elements with the MIB and YANG interface ones, the different semantic must be hardcoded in the NMS.

* For the protocols from the AAA world, TACACS+ and RADIUS interfaces are called "ports" to use the right terminology. Those have nothing to do with the networking ifIndex definitions, even if it's perfectly fine to host TACACS+ or RADIUS in routers.

* How to map with interface concept with the syslog message, where the syslog message might not have the exact same interface id (Gigabit Ethernet versus gigE versus gigEthernet ... X/Y.Z as an example) for a machine to read.

At this point, these concepts are known inside network engineer's heads, their network domain knowledge, but how to convey this information to a data scientist lacking the network domain knowledge but capable to analyze data systematically? The cost of documenting this information for the different ways it can be configured/used is enormous. Ideally protocol designers should understand that there will be an network management/automation cross domain use case that will require the integration and the potentially mapping of those different data models.

4.5. How To Connect Information For Closed Loop

Going one step further, understanding that an anomaly defined in [I-D.netana-nmop-network-anomaly-lifecycle] is connected to a symptom created by SAIN [RFC9418], which has an alert that defines an expression based on a set of metrics that are IETF but also vendor defined. The Service(s) that are experiencing an anomaly defined using a Service Intent that was defined using /[TMF921A/] but full filled using [RFC9182] that maps to one or more vendor models.

In order to be able to automate any closed loop action, the relationship between the Service Intent (defined using /[TMF921A/]), the error/policy condition that caused the symptom and the fulfillment provided at the device level via [RFC9182] all have to be understood.

Many of the operators (at the time of writing) who are trying to document and manage these relationships are trying to do it using various spreadsheets and version control systems. Instead, we could provide a way to define and query those relationships in a manner that could instantly answer all the questions that an operator has.

What if we could provide this information not only in a format the operator can understand but in a way a machine can easily interpret and make decisions.

This is the key to moving from Automation to Autonomy and one of the keys to unlocking autonomy is to leverage Knowledge Engineering and Knowledge Based Systems (KBS)

4.6. The Limits of YANG as THE Model Language

While YANG is deployed data model for configuration and monitoring, YANG has limitations that prevent it to be THE model to which other data models will be mapped. Instead, the different data models will still have a life of their own (ex: the IPFIX model does a good job for its purpose). Therefore let use introduce knowledge graph concepts, which will address the challenges.

TO BE COMPLETED.

5. Knowledge Graph Framework

Addressing these challenges mentioned in section 2 requires innovative approaches that can handle the scale and complexity of the data while ensuring accurate correlation and analysis. By leveraging advanced data management techniques and semantic technologies, network operators can unlock the full potential of their data, paving the way for more efficient and autonomous network operations. Understanding the context and relationships of the data collected is essential to overcoming the limitations of traditional siloed approaches and achieving seamless, automated network management.

A knowledge-based system (KBS) is a computer program that leverages a knowledge base and an inference engine to solve complex problems. These systems are designed to simulate human decision-making and problem-solving capabilities, making them valuable tools in various domains. Here are the key features of a knowledge-based system:

5.1. Knowledge Base

The knowledge base is the core component of a KBS, where all the explicit knowledge about the domain is stored. This knowledge is usually represented in a structured and formalized manner to facilitate easy access and manipulation. Key characteristics of the knowledge base include:

* *Explicit Representation*: Knowledge is represented in a way that can be easily interpreted and processed by the system.

* *Structured Data*: Information is organized in a structured format, such as rules, facts, and relationships.

* *Rich Semantics*: The knowledge base captures not only raw data but also the meaning and context of that data.

5.2. Inference Engine

The inference engine is the reasoning component of a KBS. It processes the information stored in the knowledge base to derive new knowledge, make decisions, or solve problems. Key functions of the inference engine include:

* *Reasoning*: Applies logical rules to the knowledge base to infer new information or conclusions.

* *Decision-Making*: Uses the inferred knowledge to make decisions or recommend actions.

* *Problem Solving*: Solves complex problems by systematically exploring possible solutions based on the knowledge base.

  In some proposed architectures the inference engine is split into individual agents that have responsability for a decomposed aspect of the Service/ Network lifecycle (e.g. anomaly detection, assurance remediation, solution proposal, solutions evaluation, solution actuation etc). The Agents (which could be AI Agents) can communicate/collaborate via the Knowledge Base.

5.3. Formal Ontology

A formal ontology is a crucial element in capturing facts about the world in which the KBS operates. It provides a structured and formal description of the concepts and relationships within a specific domain. Key aspects of formal ontologies include:

* *Concepts and Relationships*: Defines the key concepts and the relationships between them within a particular domain.

* *Standardized Vocabulary*: Provides a common vocabulary for the domain, ensuring consistency and interoperability.

* *Formal Specification*: Uses formal logic to specify the properties and constraints of the concepts and relationships, allowing for precise and unambiguous interpretation.

5.4. Comprehensive and Dynamic Knowledge

Knowledge-based systems are designed to handle a comprehensive range of knowledge within their domain and adapt to new information. This includes:

* *Extensibility*: The ability to add new knowledge and rules to the system as the domain evolves.

* *Adaptability*: The capability to update and refine the knowledge base based on new insights or changes in the environment.

* *Integration*: Combining knowledge from multiple sources to provide a holistic understanding of the domain.

6. FAIR data

FAIR Data Principles were defined in a 2016 research paper by a consortium of scientists and organizations in Nature.
The authors intended to provide guidelines to improve the Findability, Accessibility, Interoperability, and Reuse of digital assets. They have since published their principles in https://www.go-fair.org/fair-principles/.

6.1. Findability (F)

Networking systems generate large amounts of configuration, telemetry, and state data, which needs to be easily discoverable by network operators, engineers, or automated systems.

6.2. Accessibility (A)

Data within the networking domain needs to be accessible to both human operators and automated systems, with well-defined access mechanisms and protocols.

6.3. Interoperability (I)

Networking environments typically involve many devices and protocols from different vendors. Ensuring interoperability across systems is crucial for seamless network operations and management.

6.4. Reusability (R)

Reusability ensures that network data can be used and repurposed across different contexts, applications, and scenarios.

The FAIR principles have been widely cited, endorsed and adopted by a broad range of stakeholders since their publication in 2016. By intention, the 15 FAIR guiding principles do not dictate specific technological implementations, but provide guidance for improving Findability, Accessibility, Interoperability and Reusability of digital resources.

6.5. Creating And Using FAIR Knowledge Graphs

There are two major approaches to implementing knowledge graphs. Property graphs and RDF. In recent years Property graphs have gained a lot of traction with successful with companies like Neo4j and others attempting to standardize on an approach.

Property graphs are great to use in a closed application but face a number of issues when moving to large scale and open data that are designed to be FAIR. For example:

* *Schema*: Property Graphs do not have a schema. This can be considered a positive as well as negative, but having a schema that you can validate against can limit issues and bugs during implementations

* *Validation*: Property graphs do not define a way to validate data but the W3C standard, SHACL (SHApes Constraint Language) to specify constraints in a model driven fashion.

* *Globally Unique Identifiers*: The identifiers in Property Graphs are strictly local. They don’ t mean anything outside the context of the immediate database.

* *Resolvable Identifiers*: Because URI/IRIs are so similar to URLs, and indeed in many situations are URLs it makes it easy to resolve any item in RDF graph.

* *Federation*: While there are proprietary mechanisms for federating property graphs across databases e.g. neo4j fabric, federation is built into SPARQL the W3C standard for querying “triple stores” or RDF based Graph Databases.

There are ways for these two worlds to converge though, there is current work within the w3c to add properties to edges in RDF. This work RDF-star (https://w3c.github.io/rdf-star/cg-spec/editors_draft.html) (RDF-_) and SPARQL-star (SPARQL-_) at time of writing is ongoing in the W3C.

Similarly, Neo4j has a plugin "Neosemantics" that enables the use of RDF data and some of the RDF stack (OWL,RDFS,SHACL) inside of Neo4j but crucially using Cipher and not SPARQL for querys.

So as of now, RDF Knowledge Graphs have FAIR baked in and are part of the standard. Property graph approaches have proprietary solutions to help make things FAIR but there is no standard. These two worlds and approaches do seem to be converging though.

7. Introduction to the Semantic Web Technology Stack

The Semantic Web technology stack enables more sophisticated data integration, sharing, and retrieval across diverse information systems. At its core, it provides a framework for defining, linking, and querying data on the web, allowing for more intelligent and interconnected systems. Here is an overview of the key components of the Semantic Web technology stack:

7.1. URI/IRI: Uniform Resource Identifier/Internationalized Resource Identifier

A Uniform Resource Identifier (URI) is a unique sequence of characters that identifies a logical or physical resource used by web technologies. URIs may be used to identify anything, including real-world objects such as people and places, concepts, or information resources like web pages and books. The Internationalized Resource Identifier (IRI) extends the URI to include a wider range of characters from different languages, facilitating global use and interoperability.

7.2. RDF: Resource Description Framework

The Resource Description Framework (RDF) allows you to link resources (concepts) together in a way that forms a directed graph. For example, you could represent the statement "John is a person" using RDF. However, RDF alone does not provide the means to classify objects or establish complex relationships such as saying that a person is a subclass of human beings.

7.3. RDFS: RDF Schema

RDF Schema (RDFS) builds upon RDF by providing more expressive vocabulary to classify resources and establish hierarchical relationships. Using RDFS, you can define classes and subclasses (using rdfs:class and rdfs:subclass), and set restrictions on properties (relationships) within your domain knowledge using rdfs:domain and rdfs:range. This allows for a more structured and meaningful representation of data.

7.4. OWL: Web Ontology Language

The Web Ontology Language (OWL) enhances the expressiveness of RDFS by introducing more detailed and complex constraints on data. OWL categorizes properties into object properties (relationships between two resources) and data properties (relationships between a resource and a data value). It also allows you to add restrictions on properties, such as specifying cardinality constraints or defining equivalent and disjoint classes, enabling more precise and sophisticated knowledge representation.

7.5. Query: SPARQL Protocol and RDF Query Language

SPARQL (SPARQL Protocol and RDF Query Language) is an RDF query language designed for querying and manipulating data stored in RDF format. SPARQL is powerful because it can automatically join all objects in a graph from a single query, allowing for complex and efficient data retrieval. This capability makes SPARQL an essential tool for accessing and integrating diverse datasets within the Semantic Web framework.

Semantic Web technologies have significantly evolved beyond their initial web-based applications, extending into various industries (Healthcare, Finance, Manufacturing, Government and Public Sector) as a powerful means to define, integrate, and retrieve knowledge.

7.6. Validation: Shapes Constraint Language (SHACL)

SHACL (Shapes Constraint Language) can be used to enhance an RDF-based solution by providing a formal mechanism for validating the structure, content, and constraints of RDF data that models network devices, configurations, and relationships.

By using SHACL, you can ensure that the data adheres to predefined business rules, network policies, and industry standards, thus improving data quality and consistency within a management system.

7.6.1. Ensuring Data Consistency

For instance, you can use SHACL to enforce that each network device has a deviceType (e.g., router, switch) and an associated IP address, and that routers have specific attributes such as a bgpAsn (BGP Autonomous System Number).

7.6.2. Validating Relationships

For relationships between objects, SHACL allows you to validate these interconnections by specifying shapes that ensure the correct relationships are maintained.

7.6.3. Enforcing Network Policies

In network management, policies can dictate how devices are configured and how they interact. For example, a policy may require that certain VLANs are used in specific types of networks or that certain subnets are restricted to specific devices.

SHACL allows you to encode these policies as constraints and automatically validate the RDF data against them. For instance, if a policy requires that only certain VLANs are allowed on specific switches, you can define a SHACL shape to validate this.

7.6.4. Automating Configuration Compliance

In large-scale networks, automating compliance checks is essential to ensure devices are configured according to standards and policies. SHACL can be used to automatically validate the entire RDF dataset representing network configurations against predefined shapes. This can flag potential issues such as missing configurations, incorrect relationships, or policy violations, enabling network administrators to quickly take corrective action.

7.6.5. Error Reporting and Diagnostics

SHACL provides detailed error reporting and diagnostics, which can help network administrators quickly identify and fix issues in the network configuration. For each violation of a SHACL shape, SHACL can generate meaningful error messages, such as missing required properties, invalid data types, or relationships that do not conform to the network topology.

8. Why Semantic Web is Right for the Networking World?

8.1. Handling Vast Amounts of Data

Modern network operators collect extensive data from various network planes - Management, Control, and Data. Semantic Web technologies are designed to handle large datasets efficiently:

* *RDF (Resource Description Framework)* enables the modelling of data as a directed graph, allowing for flexible and scalable data representation.

* *SPARQL* can efficiently query large datasets, automatically joining relevant data points to provide comprehensive insights.

* *URI/IRI* allow data to be referenced and enriched using markup/metadata without modifying the existing systems. Information can referenced and retrieved using the schema/metadata defined in RDF.

8.2. Improved Data Correlation and Integration

Semantic Web technologies excel at linking disparate data sources and formats, which is crucial for network operations:

* *URI/IRI* provides a unique way to identify and link resources across different data silos, ensuring that all data points can be correlated accurately.

* *RDFS (RDF Schema)* and *OWL (Web Ontology Language)* provide mechanisms to classify and relate data, enabling the creation of detailed ontologies that represent network components and their relationships.

8.3. Contextual Understanding and Enhanced Metadata

One of the major challenges is the loss of context in the data collected from network operations:

* *RDFS* and *OWL* allow operators to define rich metadata about network elements. For instance, they can specify that an interface is a Provider Edge (PE) or Customer Edge (CE), and whether a link is primary or backup.

* *OWL* provides advanced features to define and enforce data properties and relationships, ensuring that the context of data is maintained and understood correctly.

8.4. Data Interoperability Across Multiple Repositories

Network data often resides in multiple repositories with different schemas and formats:

* *RDF* provides a common framework for data representation, enabling interoperability between diverse data sources.

* *Linked Data principles* can connect data from various repositories, making it easier to integrate and query across different systems.

* *Consume Or Reference Data From Multiple Sources in Multiple Formats* using well known patterns within the semantic web ecosystem for connecting external databases, whether relational, hierarchical, tabular or other graph formats.

* *Deterministic URIs* can allow data can be referenced remotely without being consumed or replicated inside a knowledge store. Thus allowing only data that is enriched to be created and stored in the knowledge and for it to reference the existing data in externally repositories.

8.5. Enhanced Fault Prediction and Automated Remediation

To predict faults and automate remediation, operators need to link and analyze data from different network planes:

* *SPARQL* can query across multiple datasets, linking management, control, and data plane information to provide a holistic view of the network.

* *OWL & RDF(S)* can define rules, relationships and constraints that breakdown the barriers between the network planes and link this data with all relevant information (e.g. context based on topologies represented by [I-D.havel-nmop-digital-map], configuration based on network element YANG).

8.6. Bridging Organizational Silos

Network configuration and operations teams often work in silos, leading to miscommunication and data fragmentation:

* *Ontologies* defined using *RDFS* and *OWL* can standardize the terminology and relationships used across teams, ensuring consistent understanding and communication.

* *Semantic annotations* can capture the intent and rationale behind network changes, preserving context and facilitating better collaboration. Importing existing schemas (whether from YANG or Relational or something else) and defining the connections between them allow the semantic of any object or value to fully known and traced within the system.

8.7. Managing Schema and Format Disparities

Different data formats (YANG, IPFIX, BMP) and schemas can make data integration challenging:

* *Semantic Web technologies* provide a unified model to represent data from various formats, enabling seamless integration and retrieval.

* *Schema mapping* using *RDF* and *OWL* can reconcile differences between schemas, providing a coherent view of the data.

It’s not only about protocol and models (IETF), we can link to the NMS/OSS layers, from the top down to the bottom up ... but also (business) intent, the BSS.

9. YANG and RDF

As mentioned above, there are more than enough models already in the telecom domain. Chief among them (from the IETF point of view) is YANG. The YANG modelling language already has many ways to augment and extend the model, but these extensions are very formal and not very dynamic.

9.1. Data catalog for YANG data sources

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” .

Knowledge graphs can enable the YANG Catalog to evolve towards a data catalog, where the YANG modules represent 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].

9.2. Translation of YANG to RDF

Since the original YANG specification [RFC6020], IETF has embraced YANG as the way to define any new models or APIs, from the device all the way up the Service model [RFC8299]. How easy would it be to convert these vast model set to RDF ?.

RDF has its roots in semantic web and is defined by the w3c, who are also the owners of the XML standard. The original [RFC6020] definition for YANG has XML deeply embedded, defining serialization formats for the YANG (YIN) in XML as well as of course instances of YANG data used by NETCONF.

Translation of XML to RDF is a well described problem and many tools exist in order to achieve it, w3c themselves have R2RML that allow custom definitions of mappings.

Generation of IRIs for YANG schema objects can be created using schema paths and IRIs for YANG instance data using XPaths. There are several advantages to this approach, the most important being that the IRI is deterministic and could be generated externally by a knowledge system without need to access the real data. There is a second advantage that it is human readable and therefore easy to browse.

The main disadvantage being the possible length of IRIs and the volume of data being processed could be lead to memory/performance issues. There are obvious trade offs to be explored but it can be seen that YANG is very much a good fit for modelling as knowledge in RDF give both formats close association to XML.

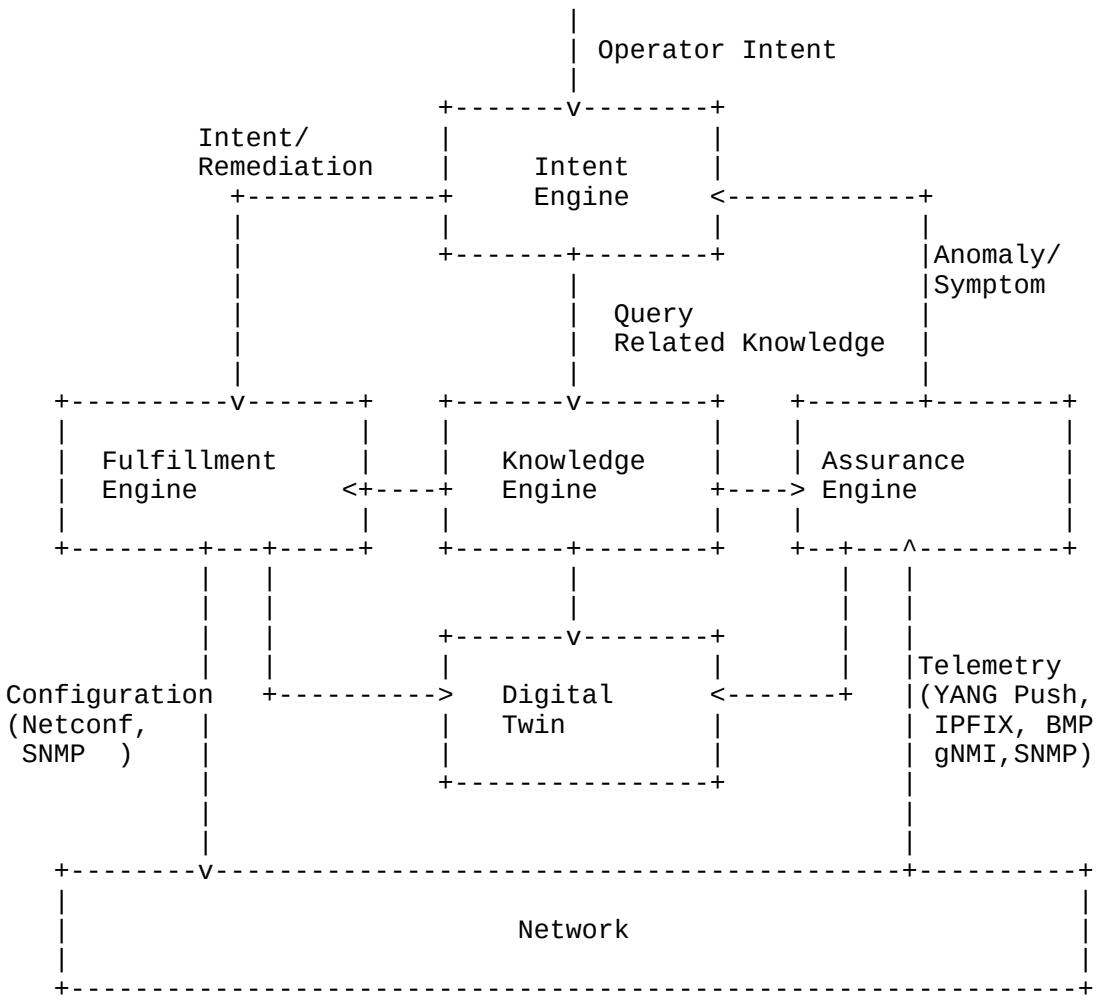
10. Knowledge Engine Positioning And Architecture

Below shows the basic positioning of the Knowledge Engine in any OSS system. As you can see the Knowledge Engine is at the heart of any decision making process.

Key to this is the ability to consume and connect data from multiple different datasources and to connect them in a single semantic layer.

Note as mentioned above, there are already many models that exist in telecommunication systems, the goal maybe not to create a new model but to provide a simple way to navigate between the existing models. Understanding that the value on the intent API that is mapped to a value on the fulfillment API that is used to configure this part of the device is connected to the Telemetry metric that was received where an anomaly was observed.

This 360 degree view of the network is the only way the secrets of the network can be unlocked and autonomous networks enabled.



10.1. Key Use Cases For Knowledge Engine

The above shows the target for Autonomous networks and automated decision processing, but for each step the Knowledge Engine can play a key role.

10.1.1. Service Intent Translation

A knowledge graph can facilitate intent translation the operator intent to the network intent by providing a unified way to query the digital twin [I-D.irtf-nmrg-network-digital-twin-arch].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 data across different datasource.

The capability to represent abstract concepts by means of ontologies, enables the representations of a generic network digital twins, regardless of the complexities of the underlying technologies. For example, an abstract representation of a network topology Digital Map [I-D.havel-nmop-digital-map] in the knowledge graph can be translated into a descriptor or data model that is specific to the technology used.

10.1.2. 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.

10.1.3. Anomaly detection and incident management

Knowledge graphs can help in the detection of anomalies in network systems by linking event/metric data (e.g. logs, alarms, and ticketing) with the context (digitial twin/network configuration). The Knowledge graph enables the connecting of these events using the context to link and explore for new connections.

10.1.4. Network Rectification:

Combing all of the above to enable the OSS to respond to issues in the network and automatically generate a change in the network to rectify the problem.

10.2. Accessing Existing Data

A key enabler that allows the information in all these systems to be exposed and connected is the Virtual Knowledge Graph. Originally called Ontology Based Data Access (OBDA), it is a collection of techniques and technologies that help overcome the challenge of combining data from different sources and formats. It uses ontologies to create a unified view of this data, and mappings to link the ontology with the individual data sources.

OBDA has evolved through three main stages:

* Materialization: Initially, OBDA focused on translating data into a common format and storing it in a central location, similar to data warehousing.

* Query Translation: To avoid the limitations of materialization, OBDA shifted towards translating queries over the unified view into queries specific to each data source.

* Declarative Mappings: The latest generation of OBDA uses declarative mapping languages, like R2RML, to define how data sources relate to the ontology. This improves flexibility and simplifies the integration process.

Virtual Knowledge Graphs provide significant advantages for data integration:

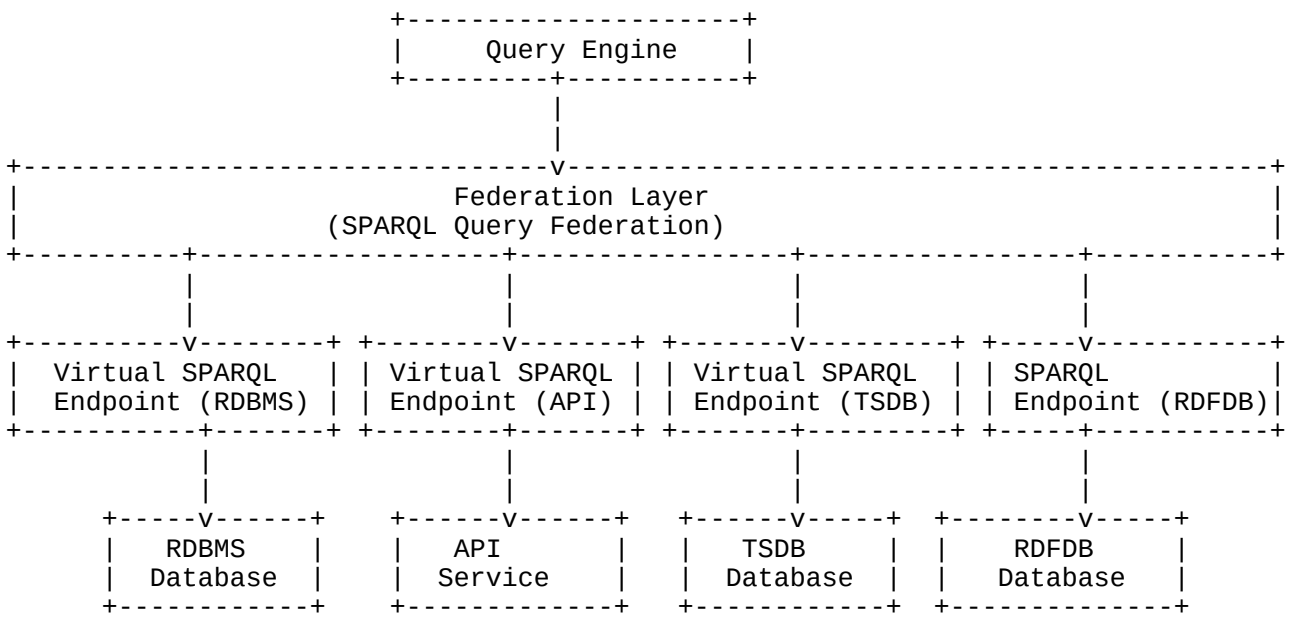
* Real-time Data Access: Data remains in its original sources, ensuring that queries always reflect the latest updates.

* Reduced Costs: Avoid the expense of building and maintaining a separate, materialized data store.

* Simplified Integration: Leverages your existing data infrastructure and expertise.

* Increased Agility: Supports an incremental approach to integration, making it easier to adapt to changes and add new data sources over time.

Unlike traditional materialized approaches that require ETL to create a unified data copy, Virtual Knowledge Graphs offer a more dynamic solution. Instead of moving data, Virtual KGs leave data in place and access it on demand. This eliminates data duplication, reduces latency, and simplifies maintenance, making it a powerful alternative for modern data integration needs.



In the Virtual Knowledge Graph (or Ontology Based Data Access) the remote schema can be imported as RDF/OWL data and used for both query and for creating new relationships over existing data. In this way existing models can be imported and the connections between silos can overlaid as extra knowledge. These relationships can be created manually or programmatically.

At query time, these relationships can be exploited to join data from different sources, allowing the connection between anomaly to assurance metric to inventory object to digital map to configuration to be traced seamlessly regardless of where the information is being stored.

10.3. What is materialised in RDF and what is Virtual ?

Given the above, you may ask what is materialised in the triple store and what is virtual, of course the answer as always is, it depends. If you have a read API that lets you access the remote data anyway you want it e.g. JDBC then maybe virtual is good enough. If however you have a restricted API that makes it difficult to query and join data (e.g. Netconf/Restconf) you may want to import that data into the graph in order to satisfy all of your queries. For this reason we have focused the initial implementation on materializing YANG schema and YANG Instance data.

11. Implementation Status

At IETF Hackathon 121 (Dublin) we successfully demonstrated an approach for translation of YANG schema models to a representation in RDF. This code is available here: https://github.com/Huawei-IOAM/yang2rdf.

At IETF Hackathon 122 (Bangkok) we will demonstrate how that YANG RDF Schema data can be used to create RDF versions of configuration data.

12. Some pointers to existing work for linked data

Linked data and semantic web has already been embraced by TMF and ETSI, their reasons for adoption are both valid both for them and for the IETF when aiming to link data in different systems and to capture knowledge.

12.1. NGSI-LD (Next Generation IoT Services Layer - Lightweight Data)

NGSI-LD is an ETSI standardized framework for representing and managing context information in the Internet of Things (IoT) domain.

Context Information Model: Represents context information as entities with properties and relationships, inspired by property graphs. *Semantic Foundation:* Based on RDF (Resource Description Framework) for formal semantics and linked data principles.

12.2. TMF921A Intent Management API

TMF tmf921a is an API for intent based networking. It aims to provide a standardized interface for expressing and managing high-level network intents, enabling automated network configuration and optimization.

Knowledge Based Reasoning: Core to the TMF approach is the ability for the intent management functions to "use reason intrinsically by examining the relationships between facts". Therefore their decision to model the intents as knowledge and to use *RDF* to define the intents.

13. Next Steps for the Industry

It’s important to note, the authors are not proposing creating a new model for the network, there is much work being done in all of the existing SDOs with numerous models managing all aspects of the network and business. The authors are proposing ways to connect all of this data and through those connections find the semantic of the information and allow it to unlock the knowledge already being

generated by the network.

To this end, the industry must come together to define ways to describe the connections between data (either at the instance level or at the schema level). Agree on formats for importing existing protocol schemas into RDF e.g. in IETF: YANG, IPFIX, BMP but also other models in different SDOs

Crucial to this is to define a way to create deterministic IRIs for both model and instance data so that information in disparate systems and repositories can be referenced, linked and enriched using the power of Knowledge graphs and linked data.

14. Security Considerations

TBC

15. IANA Considerations

This document has no actions for IANA.

16. Reference

16.1. Normative References

16.2. Informative References (to be included)

* RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, 2014.
* RDF Schema 1.1. W3C Recommendation, 2014.
* OWL 2 Web Ontology Language Document Overview. W3C Recommendation, 2012.
* SPARQL 1.1 Query Language. W3C Recommendation, 2013.
* SHACL - Shapes Constraint Language. W3C Recommendation, 2017.
* R2RML - RDB to RDF Mapping Language. W3C Recommendation, 2012.
* RML - Extend R2RML to allow mapping from any data source. v1.1.2, 2024.

17. References

17.1. Normative References

[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>.

17.2. 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..

[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..

[I-D.havel-nmop-digital-map] Havel, O., Claise, B., de Dios, O. G., Elhassany, A., and T. Graf, "Modeling the Digital Map based on RFC 8345: Sharing Experience and Perspectives", Work in Progress, Internet-Draft, draft-havel-nmop-digital-map-02, 21 October 2024, <https://datatracker.ietf.org/doc/html/draft-havel-nmop-digital-map-02>.

[I-D.ietf-core-comi] Veillette, M., Van der Stok, P., Pelov, A., Bierman, A., and C. Bormann, "CoAP Management Interface (CORECONF)", Work in Progress, Internet-Draft, draft-ietf-core-comi-20, 6 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-20>.

[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-08, 22 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ivy-network-inventory-yang-08>.

[I-D.ietf-opsawg-collected-data-manifest] Claise, B., Quilbeuf, J., Lopez, D., Martinez-Casanueva, I. D., and T. Graf, "A Data Manifest for Contextualized Telemetry Data", Work in Progress, Internet-Draft, draft-ietf-opsawg-collected-data-manifest-09, 21 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-collected-data-manifest-09>.

[I-D.irtf-nmrg-network-digital-twin-arch]
Zhou, C., Yang, H., Duan, X., Lopez, D., Pastor, A., Wu, Q., Boucadair, M., and C. Jacquenet, "Network Digital Twin: Concepts and Reference Architecture", Work in Progress, Internet-Draft, draft-irtf-nmrg-network-digital-twin-arch-11, 6 July 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-nmrg-network-digital-twin-arch-11>.

[I-D.lincla-netconf-yang-library-augmentation]
Lin, Z., Claise, B., and I. D. Martinez-Casanueva, "Augmented-by Addition into the IETF-YANG-Library", Work in Progress, Internet-Draft, draft-lincla-netconf-yang-library-augmentation-01, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-lincla-netconf-yang-library-augmentation-01>.

[I-D.netana-nmop-network-anomaly-lifecycle]
Riccobene, V., Roberto, A., Graf, T., Du, W., and A. H. Feng, "An Experiment: Network Anomaly Lifecycle", Work in Progress, Internet-Draft, draft-netana-nmop-network-anomaly-lifecycle-05, 3 November 2024, <https://datatracker.ietf.org/doc/html/draft-netana-nmop-network-anomaly-lifecycle-05>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <https://www.rfc-editor.org/rfc/rfc2865>.

[RFC3164] Lonvick, C., "The BSD Syslog Protocol", RFC 3164, DOI 10.17487/RFC3164, August 2001, <https://www.rfc-editor.org/rfc/rfc3164>.

[RFC3418] Presuhn, R., Ed., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, DOI 10.17487/RFC3418, December 2002, <https://www.rfc-editor.org/rfc/rfc3418>.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/rfc/rfc3444>.

[RFC3535] Schoenwaelder, J., "Overview of the 2002 IAB Network Management Workshop", RFC 3535, DOI 10.17487/RFC3535, May 2003, <https://www.rfc-editor.org/rfc/rfc3535>.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <https://www.rfc-editor.org/rfc/rfc5575>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/rfc/rfc6020>.

[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>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/rfc/rfc7011>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <https://www.rfc-editor.org/rfc/rfc7854>.

[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>.

[RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <https://www.rfc-editor.org/rfc/rfc8299>.

[RFC8907] Dahm, T., Ota, A., Medway Gash, D.C., Carrel, D., and L. Grant, "The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol", RFC 8907, DOI 10.17487/RFC8907, September 2020, <https://www.rfc-editor.org/rfc/rfc8907>.

[RFC9182] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., Munoz, L., and A. Aguado, "A YANG Network Data Model for Layer 3 VPNs", RFC 9182, DOI 10.17487/RFC9182, February 2022, <https://www.rfc-editor.org/rfc/rfc9182>.

[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>.

[RFC9417] Claise, B., Quilbeuf, J., Lopez, D., Voyer, D., and T. Arumugam, "Service Assurance for Intent-Based Networking Architecture", RFC 9417, DOI 10.17487/RFC9417, July 2023, <https://www.rfc-editor.org/rfc/rfc9417>.

[RFC9418] Claise, B., Quilbeuf, J., Lucente, P., Fasano, P., and T. Arumugam, "A YANG Data Model for Service Assurance", RFC 9418, DOI 10.17487/RFC9418, July 2023, <https://www.rfc-editor.org/rfc/rfc9418>.

[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..

Appendix A. Acknowledgments

The authors would like to thank Peter Cautley and Anatolii Pererva for providing the appendix example. Professor Declan O’Sullivan and Brad Peters for providing review comments.

Appendix B. Appendix

Appendix C. Resource Description Framework (RDF) schema

The RDF Schema defines the different types of relationship (RDF properties). They are defined hierarchically. That allows specific relationships to be grouped as more generalized relationships. Queries can be applied at different levels of specificity.

```
:ipfixRefersTo a rdf:Property ;
    rdfs:comment "IPFIX reference to a leaf" ;
    rdfs:subPropertyOf :refersTo .

:ipfixRefersToInterface a rdf:Property ;
    rdfs:comment "IPFIX reference to a leaf" ;
    rdfs:subPropertyOf :ipfixRefersTo .
```

Appendix D. SPARQL Protocol and RDF Query Language (SPARQL)

SPARQL finds different relationships that exist between data sources e.g. The relationship "ipfixRefersToInterface" (defined in the RDF schema) will find relationships between any IPFIX data field and the Device Interface. A more generalized relationship "ipfixRefersTo" would find all known relationships between IPFIX and Device (underlay, overlay) Configuration.

D.1. Example Of IPFIX Relationship Query

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX yang: <http://localhost/yang#>
SELECT ?source_path ?relationship ?target_path
WHERE {
  ?source ?relationship ?target .
  ?relationship rdfs:subPropertyOf* yang:ipfixRefersToInterface .
  ?source yang:path ?source_path .
  ?target yang:path ?target_path .
}
```

The result for the above query shows - the source: IPFIX fields aligned with IANA "IP Flow Information Export (IPFIX) Entities" - ingressInterface element Id 10 - egressInterface element Id 14 - the type of relationship: (specified in the RDF schema) - the target: YANG path specified in Device Configuration

source	relationship	target
"ingressInterface"	<http://localhost/yang#ipfixRefersToInterface>	"ifm/interfaces/interface/index"
"egressInterface"	<http://localhost/yang#ipfixRefersToInterface>	"ifm/interfaces/interface/index"

D.2. Example Query To Find Impacted Services & Customers

This query walks from the :Alarm through the affected link to any :Device (via interfaces), then finds services those devices provide, and finally the customers.

```
PREFIX :    <http://example.com/kg/net#>

SELECT DISTINCT ?service ?serviceLabel ?customer ?custLabel WHERE {
  # 1. Identify the outage alarm
  ?alarm     a :Alarm ;
             :affects ?link .

  # 2. Find the two interfaces on that link
  ?link      :connectsEndpoints ?intf .

  # 3. Find devices owning those interfaces
  ?intf      :partOf ?device .

  # 4. Find services provided by those devices
  ?device    :provides ?service .
  ?service   rdfs:label ?serviceLabel .

  # 5. Find the customer consuming each service
  ?service   :consumedBy ?customer .
  ?customer  rdfs:label ?custLabel .
}
```

What this does:

* Selects the alarm and its :affects link.

* Follows :connectsEndpoints to both interfaces on that link.

* Jumps from interfaces to their parent devices (:partOf).

* Gathers all services those devices :provides.

* Retrieves the customers (:consumedBy) of each service.

Running this over the above data would return: | service | serviceLabel | customer | custLabel | | ---------- | ------------------ | -------- | ------------ | | :L3VPN-100 | "Customer VPN 100" | :CustA | "Customer A" | | :L3VPN-200 | "Customer VPN 200" | :CustB | "Customer B" |

Appendix E. SHACL

SHACL (Shapes Constraint Language) can be used to enhance an RDF-based solution for network management by providing a formal mechanism for validating the structure, content, and constraints of RDF data that models network devices, configurations, and relationships. By using SHACL, you can ensure that the data adheres to predefined business rules, network policies, and industry standards, thus

improving data quality and consistency within a network management system.

Example of SHACL to validate every router that uses the BGP protocol has a valid BGP ASN configured. ~~~~ ex:BGPComplianceShape a sh:NodeShape ; sh:targetClass ex:Router ; sh:property [sh:path ex:routingProtocol ; sh:hasValue "bgp" ;] ; sh:property [sh:path ex:bgpAsn ; sh:minCount 1 ; sh:datatype xsd:integer ; sh:message "Routers using BGP must have a BGP ASN defined." ;] ; ~~~~

SHACL can also be used to validate relationships between objects, here is an example of a rule that says each router must be connected to at least one switch.

```
ex:RouterSwitchConnectionShape a sh:NodeShape ;
    sh:targetClass ex:Router ;
    sh:property [
        sh:path ex:connectedTo ;
        sh:class ex:Switch ;
        sh:minCount 1 ;
        sh:message "Each router must be connected to at least one switch." ;
    ] ;
```

Authors' Addresses

Michael Mackey
Huawei
Ireland
Email: michael.mackey@huawei.com

Benoit Claise
Everything-Ops
Belgium
Email: Benoit@everything-ops.net

Thomas Graf
Swisscom
Binring 17
CH-8045 Zurich
Switzerland
Email: thomas.graf@swisscom.com

Holger Keller
Deutsche Telekom
Germany

Email: Holger.Keller@telekom.de

Dan Voyer
Bell Canada
Canada
Email: daniel.voyer@bell.ca

Paolo Lucente
NTT
Veemweg 23
3771 Barneveld
Netherlands
Email: paolo@ntt.net

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