Network Management Operations Area Working Group
Internet-Draft
Intended status: Informational
Expires: 8 January 2026

V. Boudia
P. Francois
INSA Lyon
S. Mostafa
HUAWEI

7 July 2025

External Relationship model for SIMAP
draft-vivek-simap-external-relationship-00

Abstract

abstract

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-vivek-simap-external-relationship/.

Discussion of this document takes place on the Network Management Operations Area Working Group 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/.

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 8 January 2026.

Table of Contents

1. Introduction 2
2. Terminology 3
3. Template 4
3.1. Type 4
3.2. Parameter 4
3.3. Request 5
4. Relation 5
4.1. Request supported 6
4.2. Parameter value 6
4.3. Path 6
5. Filtering 7
6. Extensions 7
7. Workflow 7
8. YANG Modules 8
8.1. template 8
8.2. relation 11
8.3. extended example 14
9. Normative References 15
Acknowledgments 16
Authors' Addresses 16

1. Introduction

[RFC8345] defines a base model for representing network topology. This model can be augmented to include additional information, depending on the layer, service, or use case. However, continuously augmenting the base model can lead to excessive complexity and the inclusion of data that is irrelevant to many applications.

In some application contexts, it is impractical to maintain the whole network element information within the simap itself. To preserve an exploitable map of the network, it is thus sometimes preferable to

keep the base topology model minimal and maintain context-specific information separately. Yet, a formal description of the method to be used to retrieve this information from the production network or any other external source should be maintained.

The goal of this document is to provide a common approach to describe such information retrieval methods, in a yang model.

The goal of this effort is to fulfill two key requirements: REQ-SCALES and REQ-SYNCH, defined in [I-D.draft-ietf-nmop-simap-concept], Section 4.

For example, when simulating a what-if scenario involving IGP convergence time after an IS-IS link failure, one might require timer-related attributes for each router. While this information is essential in that context, it is unnecessary for many other use cases. In addition, some data elements may change frequently, making it challenging to keep the model synchronized.

In the general case, the method to retrieve a given type of information among many network elements is the same but for some parameters. We thus propose an approach to describe these methods using templates.

These templates are then associated with network element instances via a separate datastore referred to as the relation datastore. To stay efficient we add filtering mechanisms using for example typing and extra label.

This document first introduces the terminology used throughout the draft. It then describes the concepts of Template and Relation, followed by the filtering mechanisms available. Next, it explains how the model can be extended. The workflow is then presented, and the document concludes with the YANG module.

2. Terminology

Network Element: A Network, Node, Link, or Termination Point (TP), as defined in RFC 8345.

Template: A reusable model that describes how to retrieve a set of data.

Extra label: A set of keywords that describe the nature or category of information that can be retrieved using the Template.

Relation: An association between a Template and a Network Element, indicating that the Template should be applied to the element to retrieve a given set of information.

3. Template

A Template describes how to access data from a Network Element. It is uniquely identified by an id in the form of a uri. The management of the id space is out of the scope of this document.

Each Template includes a textual description that explains its purpose and usage, along with additional key attributes, such as the Template’s type, parameters, and request definitions, as described in the following sub section.

A separation between the template definition and their parameters is used in order to ensure reusability.

3.1. Type

Each Template includes a type attribute:

base-type: Indicates the general category of the data the Template is intended to retrieve. Possible values include:

* config for configuration data,

* state for operational state data,

* metric for performance metrics.

is-historical: A boolean flag indicating whether the Template is used to retrieve historical data (true) or only current data (false).

extra-label: A set of keywords that describe the nature or category of information that can be retrieved using the Template. These labels serve as metadata, enabling filtering, classification, and discovery of Templates based on their purpose or content (e.g., timers, interface-metrics, isis-state).

3.2. Parameter

Consider two routers: one with the identifier R1 and another with R2. If we wish to retrieve specific data for each router, the access paths might be /specific/data/id/R1 and /specific/data/id/R2, respectively. Without a parameterization mechanism, we would need to define a separate Template for each router, which does not scale.

To address this, Templates support parameters. Instead of hardcoding the identifier, the Template encodes the path using a placeholder: /specific/data/id/{router-id}. The actual value of {router-id} is not stored within the Template itself, but is instead provided via the Relation datastore (described in the next section).

This design enables a single Template definition to be reused across multiple Network Elements, with per-instance values injected dynamically through Relations.

3.3. Request

While parameterization helps avoid duplication when all routers support the same protocol, challenges arise when devices support different access protocols. For instance, consider two routers: R1, which supports only RESTCONF, and R2, which supports only NETCONF. In such cases, although the data being retrieved is semantically the same, using a single protocol-specific Template per device would reintroduce redundancy.

To address this, a Template may define multiple protocol-specific access methods for retrieving the same data. Each method corresponds to a different protocol (e.g., RESTCONF, NETCONF) and includes a distinct request format or path.

For example:

The RESTCONF request path might be: /restconf/specific/data/id/{router-id}

The NETCONF XML filter might be: <specific><data><id>{router-id}</id></data></specific>

The selection of which request to use for a given Network Element is determined by information stored in the Relation datastore (discussed in the next section). This allows a single Template to cover multiple access methods, while preserving reusability and minimizing duplication.

4. Relation

A Relation defines the association between a Template and a Network Element. It provides the contextual parameters required to apply the Template to the targeted element.

Each Relation is uniquely identified by an id. To establish the linkage, the Relation contains:

* A reference to the template-id of the associated Template.

* A reference to the network-element-id of the target Network Element.

A Relation contains additional contextual information that will be described in the following sections, including:

* the supported request types,

* the parameter values, and

* the path identifying the target sub-layer (if applicable).

4.1. Request supported

As previously described, a Template may include multiple protocol-specific request definitions. Since not all devices support all protocols, the Relation is responsible for specifying which request(s) are valid for the associated Network Element. This ensures that the correct method is used for data retrieval, without duplicating Template definitions.

4.2. Parameter value

In addition, each Relation stores the parameter values required to instantiate the associated Template. These values are used to replace placeholders (e.g., {router-id}) defined in the Template when constructing the actual request.

If a parameter required by the Template is not provided in the Relation, it is assumed that the user or the consuming system must supply this value at runtime. This design provides flexibility while ensuring that Templates remain reusable across different contexts.

4.3. Path

Depending on the network modeling approach, a Network Element may be encapsulated within multiple hierarchical layers — for example: base → L3 → IS-IS. Since each Network Element is uniquely identified at the base level, additional context may be required to indicate which layer or model a Template applies to.

To address this, the Relation includes an optional path value. This field specifies the relative path from the base element to the targeted sub-layer. For instance:

* If the Template targets IS-IS attributes, the path value might be set to /l3/isis.

* If the Template applies to the base layer, the path is set to an empty string.

This mechanism enables precise scoping of the Template within a layered network model, without introducing ambiguity.

5. Filtering

To support efficient access and scalability, the system provides multiple filtering mechanisms for querying Relations and Templates. These filters allow consumers to retrieve only the relevant associations based on criteria.

Supported filtering options include:

* By Network Element: Retrieve all Relations associated with a given network element identifier.

* By Request Type: Filter Relations or Templates based on supported protocol access methods (e.g., restconf, netconf).

* By Template Type: Select Templates based on their declared base-type (e.g., config, state, metric).

* By Extra Label : Retrieve Templates based on semantic tags (e.g., timers, isis, interface-metrics) that describe the type of information provided.

These filtering capabilities enable flexible integration with automation systems and monitoring platforms, while minimizing unnecessary processing and data retrieval.

6. Extensions

A base set of standard request types is provided as part of the Template request model. However, the model is extensible, allowing users to define custom request types as needed. This can be done by creating new request definitions within the Template, without requiring changes to the underlying data model.

7. Workflow

The following workflow outlines the typical steps involved in defining and using Templates and Relations to retrieve data from network elements of SIMAP :

1) Define a reusable Template that describes how to access a type of data. 2) Create the SIMAP from the network. 3) Establish Relations to bind Templates to Network Elements based on their role. Populate the required parameters, path, and supported request types. 4) Using the filtering cocnept the system retrive the template needed and resolves parameters and selects the correct request type to fetch the desired data.

8. YANG Modules

8.1. template

```
<CODE BEGINS>
module draft-template {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:draft-template";
  prefix dr-tmp;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "INSA Lyon";
  contact
    "Editor:   Vivekananda Boudia
     <mailto:vivekananda.boudia@insa-lyon.fr>";
  description
    "template yang model";

  revision 2025-03-09 {
    description
      "Initial revision";
    reference
      "";
  }

  identity template-type {
    description
      "base identity for template type";
  }

  identity CONFIG {
    base template-type;
    description
      "template used to retrieve configuration data";
```

```
}

identity STATE {
  base template-type;
  description
    "template used to retrieve operational state data";
}

identity extra-label {
  description
    "base identity for extra label";
}

identity request-type {
  description
    "base identity for request type";
}

identity ALL_REQUEST {
  base request-type;
  description
    "all request";
}

identity NETCONF {
  base request-type;
  description
    "request that retrieves data using the NETCONF protocol";
}

grouping netconf-request {
  description
    "netconf request";
  leaf xpath {
    type string;
    description
      "netconf xpath for request";
  }
}

container template {
  description
    "template container";
  list template {
    key "template-id";
    description
      "template list";
    leaf template-id {
```

```
          type inet:uri;
          description
            "uniquely identifies a template";
        }
        leaf description {
          type string;
          description
            "template description";
        }
        container template-type {
          description
            "template type
             used for filtering";
          leaf base {
            type identityref {
              base template-type;
            }
            description
              "template base
               used for filtering";
          }
          leaf is-historical {
            type boolean;
            description
              "check is template is used to get historical data or not
               used for filtering";
          }
          leaf-list extra-label {
            type identityref {
              base extra-label;
            }
            description
              "extra label
               used for filtering";
          }
        }
        list parameter {
          key "param-id";
          description
            "list of parameter used by request";
          leaf param-id {
            type inet:uri;
            description
              "uniquely identifies a parameter";
          }
          leaf description {
            type string;
            description
```

```
              "parameter description";
          }
        }
        list request {
          key "request-type";
          description
            "request list";
          leaf request-type {
            type identityref {
              base request-type;
            }
            description
              "request type";
          }
          container request-builder {
            description
              "request container that allows users to retrieve data
               parameters must be enclosed in brackets.";
          }
        }
        container extra {
          description
            "use for augmentation";
        }
      }
    }

    augment "/template/template/request/request-builder" {
      when "derived-from-or-self(../request-type, 'NETCONF')";
      uses netconf-request;
      description
        "adding netconf request to possible request";
    }
  }
  <CODE ENDS>
```

8.2. relation

```
  <CODE BEGINS>
  module draft-relation {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:draft-relation";
    prefix dr-rel;

    import ietf-inet-types {
      prefix inet;
      reference
        "RFC 6991: Common YANG Data Types";
```

```
}
import ietf-network {
  prefix nw;
  reference
    "RFC 8345: A YANG Data Model for Network Topologies";
}
import ietf-network-topology {
  prefix nt;
  reference
    "RFC 8345: A YANG Data Model for Network Topologies";
}
import draft-template {
  prefix dr-tmp;
}

organization
  "INSA Lyon";
contact
  "Editor:   Vivekananda Boudia
   <mailto:vivekananda.boudia@insa-lyon.fr>";
description
  "relations external of RFC8345";

revision 2025-03-09 {
  description
    "Initial revision";
  reference
    "";
}

container relation {
  description
    "relation container";
  list relation {
    key "relation-id";
    description
      "relation list";
    leaf relation-id {
      type inet:uri;
      description
        "relation id";
    }
    choice network-element-ref {
      description
        "linking to RFC8345";
      leaf network-ref {
        type leafref {
          path "/nw:networks/nw:network/nw:network-id";
```

```
            }
            description
              "linking to network";
          }
          leaf node-ref {
            type leafref {
              path "/nw:networks/nw:network/nw:node/nw:node-id";
            }
            description
              "linking to node";
          }
          leaf link-ref {
            type leafref {
              path "/nw:networks/nw:network/nt:link/nt:link-id";
            }
            description
              "linking to link";
          }
          leaf tp-ref {
            type leafref {
              path "/nw:networks/nw:network/nw:node/nt:termination-point/nt:tp-id";
            }
            description
              "linking to termination point";
          }
        }
        leaf template-ref {
          type leafref {
            path "/dr-tmp:template/dr-tmp:template/dr-tmp:template-id";
          }
          description
            "reference to template";
        }
        leaf-list request-type-supported {
          type identityref {
            base dr-tmp:request-type;
          }
          description
            "template is generic and may include requests that are not supported by the
network element
             here, we specify the types of requests that the network element supports
             if network element supports all template ALL-REQUEST may be used";
        }
        leaf path {
          type string;
          description
            "network element can be augmented and may contain containers nested within o
ther containers.
             path is used for filtering.";
        }
```

```
          list parameter-value {
            key "param-ref request-type";
            description
              "parameter value from network element";
            leaf param-ref {
              type leafref {
                path "/dr-tmp:template/dr-tmp:template/dr-tmp:parameter/dr-tmp:param-id";
              }
              description
                "reference to template parameter";
            }
            leaf request-type {
              type identityref {
                base dr-tmp:request-type;
              }
              description
                "value can be different depending on the request";
            }
            leaf value {
              type string;
              description
                "value of the parameter";
            }
          }
        }
      }
    }
    <CODE ENDS>
```

8.3. extended example

```
    <CODE BEGINS>
    module draft-extended-example {
      yang-version 1.1;
      namespace "urn:ietf:params:xml:ns:yang:draft-extended-example";
      prefix dr-ext-exp;

      import draft-template {
        prefix dr-tmp;
      }

      organization
        "INSA Lyon";
      contact
        "Editor:   Vivekananda Boudia
         <mailto:vivekananda.boudia@insa-lyon.fr>";
      description
        "add external references to RFC8345";
```

```
     revision 2025-03-09 {
       description
         "Initial revision.";
       reference
         "";
     }

     identity AUGMENT-EXAMPLE-TEMPLATE-TYPE {
       base dr-tmp:template-type;
       description
         "augment example for template type";
     }

     identity AUGMENT-EXAMPLE-EXTRA-LABEL {
       base dr-tmp:extra-label;
       description
         "augment example for extra label";
     }

     identity AUGMENT-EXAMPLE-REQUEST-TYPE {
       base dr-tmp:request-type;
       description
         "augment example for request type";
     }

     grouping augment-example-request {
       description
         "augment example for a request";
       leaf foo {
         type string;
         description
           "leaf example";
       }
     }

     augment "/dr-tmp:template/dr-tmp:template/dr-tmp:request/dr-tmp:request-builder" {
       when "derived-from-or-self(../dr-tmp:request-type, 'AUGMENT-EXAMPLE-REQUEST-TYPE')
";
       uses augment-example-request;
       description
         "we add our example to the possible request type";
     }
   }
   <CODE ENDS>
```

9. Normative References

[I-D.draft-ietf-nmop-simap-concept]
Havel, O., Claise, B., de Dios, O. G., and T. Graf, "SIMAP: Concept, Requirements, and Use Cases", Work in Progress, Internet-Draft, draft-ietf-nmop-simap-concept-05, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-nmop-simap-concept-05>.

[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>.

Acknowledgments

Acknowledgments

Authors' Addresses

Vivekananda Boudia
INSA Lyon
Email: vivekananda.boudia@insa-lyon.fr

Pierre Francois
INSA Lyon
Email: pierre.francois@insa-lyon.fr

Sherif Mostafa
HUAWEI
Email: sherif.mostafa@huawei.com