Workgroup: NETMOD
Internet-Draft:
draft-jouqui-netmod-yang-full-include
Published: 6 November 2023
Intended Status: Standards Track
Expires: 9 May 2024
Authors: T. Joubert J. Quilbeuf B. Claise
Huawei Huawei Huawei

YANG Full Include

Abstract

YANG lacks re-usability of models defined outside of the grouping and augmentation mechanisms. For instance, it is almost impossible to reuse a model defined for a device in the context of the network, i.e by encapsulating it in a list indexed by device IDs. [RFC8528] defines the YANG mount mechanism, partially solving the problem by allowing to include schemas at deploy or runtime. This document aims to provide the same mechanism at design time.

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 9 May 2024.

Table of Contents

1. Introduction

[RFC8528] introduces the challenges of reusing existing YANG modules, especially when they need to be included under a specific node of another module. In that RFC, three different phases of data model life cycle are identified: "design time", "implementation time" and "run time". Only the last two are covered. We focus here on the first phase of the life cycle, that is inserting modules at design time.

We identified some use cases that require this design time definition of which modules need to be included in the top-level module. The have in common the need to re-use YANG modules defined for the devices in the context of a network-level module. Also, they both aim to define a model that is independent of the underlying devices.

*The use case that triggered the creation of this document is [I-D.ietf-opsawg-collected-data-manifest]. In this draft, the goal is to provide a YANG model giving the context in which YANG-push [RFC8641] data are collected so that they can be exploited a posteriori. To get the full context, we need the hardware and os version of each device, but also the list of YANG modules supported by the devices and the parameters for the YANG-push subscriptions. For the last two items YANG Library [RFC8525] and

YANG Push [RFC8641] provide good and standard modules for representing this information at the device level. However, the data manifests need to be considered at the network level, so that we can distinguish between the devices from which they come. In YANG, that means including them in a list indexed by the device id, which proves out to be difficult without copy-pasting the original modules.

*A similar use case is the digital map [I-D.havel-opsawg-digital-map], where the goal is to build a model of the network. In particular, to model the devices a lot of standard modules have already been defined by the IETF and there is a need to reuse these modules to build this larger network model. The IVY workgroup might also rely on the pattern of re-using device level modules into a network model.

YANG Schema Mount [RFC8528] and Peer Mount [I-D.clemm-netmod-peermount] focus on mounting a given part of a an existing data instance into another data instance. Although the final goal is the same: being able to reuse modules defined elsewhere in order to avoid redefining them, the approach is more focused on the runtime than the design time. In the first case, the mapping between the mount points and the existing modules to be included at that mount point is left to the NETCONF [RFC6241] server. Thus, to guarantee that the contents under a given mount point conforms to a predefined schema requires the proper configuration of the server. In the case of Peer mount, the focus is on synchronizing a given subtree of a remote server with a subtree of the local server. Again, the contents under the local subtree cannot be enforced from the design time.

In this document, we propose a new extension, named full include. This extension enables reusing imported modules by rooting them at an arbitrary point of the data model. The concept of mount point from [RFC8528] is replaced by a list of "full:include" statement, each statement corresponding to the inclusion of one imported module at that location. In that sense, the design time solution is a pure YANG solution that does not rely on external configuration to specify the list of mounted modules, hence the term full include rather than mount.

The obtained data model that we want to associate to our construct is similar to the one obtained by specifying a mount point and binding it to the same set of modules. Therefore, we can reuse the concepts of the YANG schema mount to define the semantics of our new extension.

2. Terminology

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.

The following terms are defined in [RFC7950]:

*data model

*data node

The following terms are defined in [RFC7950]:

*mount point

3. Full Include

The full include mechanism defined in this document completes [RFC8528], by providing a mechanism to "mount" modules at design time. Supporting mounting modules at this step of the data model life cycle is left out of scope in [RFC8528].

The approach for supporting the full include mechanism is to keep the semantics of [RFC8528] for the resulting data model. In [RFC8528], the list of modules to mount in each mount point is left to the NETCONF server. In this document, we propose the full include mechanism to define this mapping directly in the YANG module that includes the mounted modules.

In the sequel, we use "full" as the prefix for the module 'ietf-yang-full-include' (see Section 4). Thus "full:include" refers to the extension 'include' defined in that module.

3.1. Definition

The "full:include" statement can appear as a sub-statement of the following statements:

*container

*list

The "full:include" statement takes as an argument a prefix, that must be the prefix associated to an imported module. Modules can contain multiple uses of the "full:include" statement. The "container" and "list" statements MAY contain multiple uses of the "full:include" statement on the same level. These multiple uses define the full list

of modules to be included, rooted in node where the "full:include" statement is used.

The "full:include" statement can be interpreted using YANG Schema Mount [RFC8528], by following these steps:

1. For each set of "full:include" statement located in the same node, replace them by a single "mount-point" with a unique label.

2. Declare each of these "mount-points" as "shared-schema" in the data model defined in [RFC8528].

3. In the instance corresponding to each "mount-point", define the ietf-yang-library [RFC8525] to include a module-set (at '/yang-library/module-set/) with the following. The list 'module' contains an entry for every module referred to in the set of "full:include" statements corresponding to the "mount-point". Additionally, the list 'module' contains an entry for "ietf-yang-library" as it is needed by YANG Schema mount. As usual, the list 'imported-modules' contains the list of dependencies needed by the modules in the 'module' list.

An example of module using "full:include" and its translation into a similar YANG Schema mount version is presented in Appendix B.

3.2. Limitations

*A module MUST NOT use the "full:include" statement with its own prefix as argument. This rule prevents any infinite recursion in the mounted schemas.

*Modules used as arguments for the "full:include" statement MUST be valid and compile successfully, independently of the module it is used in.

4. ietf-full-include YANG module

We present in this section the YANG module defining the "full-include" extension. The module in itself defines solely the 'include' extension. A module importing this extension SHOULD the prefix 'full', so that the statement reads "full:include" when used in the code.

```
<CODE BEGINS> file "ietf-full-include@2023-11-03.yang"

module ietf-yang-full-include {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-full-include";
  prefix full;

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Editor:   ";
  description
    "This module defines a YANG extension statement that can be used
     to incorporate data models defined in other YANG modules in a
     module.

     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 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.

     Copyright (c) 2023 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Revised BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX;
     see the RFC itself for full legal notices.";

  revision 2023-11-05 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: YANG Full Include";
  }

  extension include {
    argument prefix;
    description
      "The argument 'prefix' MUST be the prefix of a module imported
       by the calling module.
```

```
        The 'include' statement MUST NOT be used in a YANG
        version 1 module, neither explicitly nor via a 'uses'
        statement.
        The 'include' statement MAY be present as a substatement
        of 'container' and 'list' and MUST NOT be present elsewhere.

        Whenever a sequence of 'include' statements is used, the
        schema tree defined by the set of the included modules is
        inserted in the schema tree of the calling module, at the
        place where the sequence is declared";
  }
}
```

<CODE ENDS>

5. Security Considerations

TODO

6. IANA Considerations

TODO

7. Contributors

8. Open issues

*What name should we give to this draft? Any suggestions instead of full include?

*Do we want to support the parent-nodes mechanism from [RFC8528]?

*Do we allow full include from within a grouping?

*Does this mechanism already exist?

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/info/rfc2119>.

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

[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/info/rfc8174>.

[RFC8528] Bjorklund, M. and L. Lhotka, "YANG Schema Mount", RFC 8528, DOI 10.17487/RFC8528, March 2019, <https://www.rfc-editor.org/info/rfc8528>.

9.2. Informative References

[I-D.clemm-netmod-peermount] Clemm, A., Voit, E., Guo, A., and I. D. Martinez-Casanueva, "Mounting YANG-Defined Information from Remote Datastores", Work in Progress, Internet-Draft, draft-clemm-netmod-peermount-02, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-clemm-netmod-peermount-02>.

[I-D.havel-opsawg-digital-map] Havel, O., Claise, B., de Dios, O. G., Elhassany, A., Graf, T., and M. Boucadair, "Modeling the Digital Map based on RFC 8345: Sharing Experience and Perspectives", Work in Progress, Internet-Draft, draft-havel-opsawg-digital-map-01, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-havel-opsawg-digital-map-01>.

[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-02, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-collected-data-manifest-02>.

[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/info/rfc6241>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/

RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

[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/info/rfc8641>.

Appendix A. Changes between revisions

No revisions

Appendix B. Examples

In this section we present some minimalistic examples in order to illustrate the "full:include" statement. For these examples, we are in a situation where we have a device-level module already defined and we want to have a network-level module that represent a list of device, each having an independent instance of the device-level module. This situation might arise if we want to simplify the network management by presenting a unified model for the network. In that case, the heterogeneity of the devices should be handled by mapping their model to the device-level module (which is clearly out of scope for this draft).

In our simplistic example, the device-level module simply exposes the hostname and the cpu-usage of the device. Note that we cannot modify this device-level module, because in a more realistic example we would be reusing standard modules. The tree representation ([RFC8340]) of the 'device-level' module is depicted in Figure 1.

```
module: device-level
  +--rw hostname     string
  +--ro cpu-usage?   int8
```

Figure 1: YANG Tree representation of the device-level module.

For the network-level module, we have a list of devices indexed by their 'device-id'. The tree representation ([RFC8340]) of such a module is depicted in Figure 2.

```
module: network-level-stub
  +--rw devices
     +--rw device* [device-id]
        +--rw device-id    string
```

Figure 2: YANG Tree representation of a stub for the network-level module

The goal is now to complete this stub so that the full contents of the 'device-level' is added under the "device" list.

B.1. Example using YANG Full Include

We propose in this section a YANG module for 'network-level'. The YANG code is presented in Figure 3.

```
module network-level {
  yang-version 1.1;
  namespace "urn:network-level";
  prefix "net-l";

  import "ietf-yang-full-include" {
    prefix "full";
  }

  import "device-level" {
    prefix "dev-l";
  }

  container devices {
    list device {
      key device-id;
      leaf device-id {
        type string;
      }
      full:include "dev-l";
    }
  }
}
```

Figure 3: Version of the network-level module using full:include

At the moment, this code is accepted by the YANG compilers, but since the extension is not implemented, it simply ignores it. Note that all the information (which modules to include, where to include them) is defined in this module. More specifically, the line 'full:include "dev-l";' states that the full schema of the 'device-level' module, identified by its prefix "dev-l" must be included at that location. By adding more occurrences of "full:include" there, one can define a more complex schema to be included at that location.

B.2. Using YANG Schema Mount

In this section, we show how a similar result could be attained using YANG Schema Mount. The network-level module is presented in Figure 4.

```
module network-level {
  yang-version 1.1;
  namespace "urn:network-level";
  prefix "net-l";

  import ietf-yang-schema-mount {
    prefix yangmnt;
  }

  container devices {
    list device {
      key device-id;
      leaf device-id {
        type string;
      }
      yangmnt:mount-point "device-schema";
    }
  }
}
```

Figure 4: Version of the network-level module using Schema Mount

As explained in Section 3.1, the yang-library corresponding to the modules to include, as well as the data required by 'ietf-yang-mount' needs to be specified in some other files. Using the 'yanglint' tool from libyang (https://github.com/CESNET/libyang), this module can be compiled to provide a tree representation as shown in Figure 5.

```
module: network-level
  +--rw devices
     +--mp device* [device-id]
        +--rw hostname/    string
        +--ro cpu-usage/?  int8
        +--rw device-id    string
```

Figure 5: Full tree of both network- and device-level using Schema Mount

The command for obtaining that schema is 'yanglint -f tree -p . -x extension-data.xml -Y network-level-yanglib.xml yang/network-

level.yang', assuming all the YANG modules and the two xml files are in the current folder. The file 'network-level-yanglib.xml' contains the YANG Library data for the network-level module. The file 'extension-data.xml' contains the YANG Library data defining the schema to use at the mount point, as well as the data required by YANG Schema Mount. Both are reproduced in Appendix B.3.

B.3. Support Files

The code of the 'device-level' module is given in Figure 6. Then the data files 'network-level-yanglib.xml' and 'extension_data.xml' are provided. These files are needed to compile the Schema Mount version of our example with yanglint.

```
module device-level {
  yang-version 1.1;
  namespace "urn:device-level";
  prefix mnt;

  leaf hostname {
    type string;
    mandatory true;
  }
  leaf cpu-usage {
    type int8;
    config false;
  }
}
```

Figure 6: device-level YANG module

```
<CODE BEGINS> file "network-level-yanglib.xml"

<yang-library xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"
     xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
   <module-set>
     <name>main-set</name>
     <module>
       <name>ietf-datastores</name>
       <revision>2018-02-14</revision>
       <namespace>urn:ietf:params:xml:ns:yang:ietf-datastores</namespace
     </module>
     <module>
       <name>ietf-yang-library</name>
       <revision>2019-01-04</revision>
       <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-library</namespa
     </module>
     <module>
       <name>ietf-yang-schema-mount</name>
       <revision>2019-01-14</revision>
       <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount</na
     </module>
     <module>
       <name>network-level</name>
       <namespace>urn:network-level</namespace>
     </module>
     <import-only-module>
       <name>ietf-yang-types</name>
       <revision>2013-07-15</revision>
       <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-types</namespace
     </import-only-module>
     <import-only-module>
       <name>ietf-inet-types</name>
       <revision>2013-07-15</revision>
       <namespace>urn:ietf:params:xml:ns:yang:ietf-inet-types</namespace
     </import-only-module>
   </module-set>
   <schema>
     <name>main-schema</name>
     <module-set>main-set</module-set>
   </schema>
   <datastore>
     <name>ds:running</name>
     <schema>main-schema</schema>
   </datastore>
   <datastore>
     <name>ds:operational</name>
     <schema>main-schema</schema>
   </datastore>
   <content-id>1</content-id>
```

```
 </yang-library>
 <modules-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
   <module-set-id>2</module-set-id>
 </modules-state>


<CODE ENDS>
```

```
<CODE BEGINS> file "extension_data.xml"

<yang-library xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"
              xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <module-set>
        <name>mountee-set</name>
        <module>
            <name>device-level</name>
            <namespace>urn:device-level</namespace>
        </module>
        <module>
            <name>ietf-datastores</name>
            <revision>2018-02-14</revision>
            <namespace>urn:ietf:params:xml:ns:yang:ietf-datastores</name
        </module>
        <module>
            <name>ietf-yang-library</name>
            <revision>2019-01-04</revision>
            <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-library</na
        </module>
        <import-only-module>
            <name>ietf-yang-types</name>
            <revision>2013-07-15</revision>
            <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-types</name
        </import-only-module>
        <import-only-module>
            <name>ietf-inet-types</name>
            <revision>2013-07-15</revision>
            <namespace>urn:ietf:params:xml:ns:yang:ietf-inet-types</name
        </import-only-module>
    </module-set>
    <schema>
        <name>test-schema</name>
        <module-set>mountee-set</module-set>
    </schema>
    <datastore>
        <name>ds:running</name>
        <schema>test-schema</schema>
    </datastore>
    <datastore>
        <name>ds:operational</name>
        <schema>test-schema</schema>
    </datastore>
    <content-id>2</content-id>
</yang-library>
<modules-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
   <module-set-id>2</module-set-id>
</modules-state>
<schema-mounts xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount
```

```
<mount-point>
    <module>network-level</module>
    <label>device-schema</label>
    <shared-schema/>
</mount-point>
</schema-mounts>


<CODE ENDS>
```

Acknowledgements

TODO: acknowledgements

Authors' Addresses

Thomas Joubert
Huawei

Email: thomas.joubert1@huawei-partners.com

Jean Quilbeuf
Huawei

Email: jean.quilbeuf@huawei.com

Benoit Claise
Huawei

Email: benoit.claise@huawei.com