Network Working Group
Internet-Draft
Updates: 7950 (if approved)
Intended status: Standards Track
Expires: June 18, 2018

B. Claise
J. Clarke
Cisco Systems, Inc.
B. Lengyel
Ericsson
K. D'Souza
AT&T

December 15, 2017

New YANG Module Update Procedure
draft-clacla-netmod-yang-model-update-03

Abstract

This document specifies a new YANG module update procedure in case of backward-incompatible changes, as an alternative proposal to the YANG 1.1 specifications. This document updates RFC 7950.

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 http://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 June 18, 2018.

Copyright Notice

Table of Contents

1. Introduction

The YANG data modeling language [RFC7950] specifies strict rules for updating YANG modules (see section 11 "Updating a Module"). Citing a few of the relevant rules:

1. "As experience is gained with a module, it may be desirable to revise that module. However, changes to published modules are not allowed if they have any potential to cause interoperability problems between a client using an original specification and a server using an updated specification."

2. "Note that definitions contained in a module are available to be imported by any other module and are referenced in "import"

statements via the module name. Thus, a module name MUST NOT be changed. Furthermore, the "namespace" statement MUST NOT be changed, since all XML elements are qualified by the namespace."

3. "Otherwise, if the semantics of any previous definition are changed (i.e., if a non-editorial change is made to any definition other than those specifically allowed above), then this MUST be achieved by a new definition with a new identifier."

4. "deprecated indicates an obsolete definition, but it permits new/ continued implementation in order to foster interoperability with older/existing implementations."

What are the consequences?

1. Ideally, the YANG module names should not be changed due the importance of not changing the automation code in case of import statements or service composition at the orchestration layer.

2. When the same YANG module name is kept, its new revision must be updated in a backward-compatible way.

3. While most of the non-backward compatible changes are prohibited, a client still does not know if a changed module is backward compatible, as a server may remove parts of a module after marking it deprecated or obsolete.

2. The Problems

This section lists a series of problems, hopefully listed in a logical order, which leads to the solution in the next section.

2.1. Slow Standardization

The conclusions drawn in the introduction lead to the logical conclusion that the standardized YANG modules have to be perfect on day one (at least the structure), which in turn might explain why all the IETF YANG modules take so long to standardize. Shooting for perfection (at least in structure) is obviously a noble goal, but if the perfect standard comes too late, it doesn’t help the industry.

2.2. Some YANG Modules are Not Backward Compatible

As we learn from our mistakes, we’re going to face more and more backward-incompatible YANG modules. An example is the YANG data model for L3VPN service delivery [RFC8049], which, based on implementation experience, must be updated in a backward-incompatible way with draft-wu-l3sm-rfc8049bis [I-D.wu-l3sm-rfc8049bis].

While Standards Development Organization (SDO) YANG modules are obviously better for the industry, we must recognize that many YANG modules are actually generated YANG modules (for example, from internal databases), also known as native YANG modules, or vendor modules [RFC8199]. From time to time, the new YANG modules are not backward-compatible.

In such cases, it would be better to indicate how backward-compatible a given YANG module actually is.

2.3. Non-Backward Compatible Errors

Sometimes small errors force us to make non-backward compatible updates. As an example imagine that we have a string with a complex pattern (e.g., an IP address). Let’s assume the initial pattern incorrectly allows IP addresses to start with 355. In the next version this is corrected to disallow addresses starting with 355. Formally this is an non-backward compatible change as the value space of the string is decreased. In reality an IP address and the implementation behind it was never capable of handling an address starting with 355. So practically this is a backward compatible change, just like a correction of the description statement. Still current YANG rules would force a module name change.

2.4. A Zoo of YANG Modules

Even if we focus on the IETF, we have to observe that many SDOs, opensource fora, and vendors develop YANG modules. This should be considered a success for an IETF developed technology. However, the operators are faced with this problem: how to select the YANG modules to take into account for their service developments.

The site <https://www.yangcatalog.org> (and the YANG catalog that it provides: YANG module for yangcatalog.org, [I-D.clacla-netmod-model-catalog]) is an attempt to become a reference for all YANG modules available in the industry, for both YANG developers to search on what exists already) and for operators (to discover the more mature YANG models to automate services). This YANG catalog should not only contain pointers to the YANG modules themselves, but also contain metadata related to those YANG modules: What is the module type (service model or not?); what is the maturity level? (e.g., for the IETF: is this an RFC, a working group document or an individual draft?); is this module implemented?; who is the contact?; is there open-source code available? And we expect many more in the future. The industry has begun to understand that the metadata related to YANG models become equally important as the YANG models themselves.

The yangcatalog.org instantiation of the catalog provides a means for module authors and vendors implementing modules to upload their metadata, which is then searchable via an API, as well as using a variety of web-based tools. The instructions for contributing and searching for metadata can be found at <https://www.yangcatalog.org/contribute.php>.

The issue is actually the number of YANG modules the operators are offered. At the time of writing this document, the number of unique YANG modules in the catalog is exactly 2596 (and that number keeps growing), while the IETF has standardized or is busy standardizing a small subset of those. Therefore, it’s important to distinguish the relevant YANG modules with the pack and to understand the relationship between the YANG modules.

2.5. YANG Modules Obsolete Relationship

So the operators use the yangcatalog.org to discover which YANG modules they can use NOW. They base their selection not only on the YANG module content, but also on the related metadata. When faced with the zoo of the YANG modules, it’s difficult to understand the relationship between YANG modules. As an example: how could an operator discover that YANG-MODULE-B obsoletes YANG-MODULE-A? Indeed, both have different YANG module names. The only available information is an "obsolete" tag in the published RFC containing YANG-MODULE-B: this tag would point to YANG-MODULE-A. In the world of automation, going through a published RFC as a level of indirection to understand the YANG module obsolete relationship is a non-starter. Food for thought: the IETF should stop thinking that the metric for success is an RFC number, as opposed to the contained YANG module(s).

We need an automatic way to discover that a YANG-MODULE-B obsoletes YANG-MODULE-A, so that YANG-MODULE-A should not be given any attention.

The following example is not an automatic way.

```
    description
        "This YANG module defines a generic service configuration
        model for Layer 3 VPNs. This model is common across all
        vendor implementations. This obsoletes the RFC8049 YANG
        module, ietf-l3vpn-svc@2017-01-2";
    revision 2017-09-14 {
        description
        "First revision of RFC8049.";
        reference
        "RFC xxxx: YANG Data Model for L3VPN Service Delivery";
```

Along the same lines, while going through an out-of-band tool such as the yangcatalog.org in order to discover the obsolete relationship is a possible automatic way, it is not ideal.

2.6. YANG Module Transition Strategy

Let’s assume for a moment that we change the YANG module, with the specific example of ietf-routing, which some propose to update to ietf-routing-2.

Here are all the ietf-routing dependent YANG modules (those modules that depend on ietf-routing) <https://www.yangcatalog.org/yang-search/impact_analysis.php?modules[]=ietf-routing&recurse=0&rfcs=1&show_subm=1&show_dir=dependents>. So many YANG modules.

Let’s look at the difference for ietf-routing-2: <https://www.yangcatalog.org/yang-search/impact_analysis.php?modules[]=ietf-routing-2&recurse=0&rfcs=1&show_subm=1&show_dir=dependents>.

Changing the module name from ietf-routing to ietf-routing-2 implies that the we have to warn all draft authors of ietf-routing YANG dependent modules. First, to make sure they are aware of ietf-routing-2 (publishing a RFC8022bis mentioning in the module description that this module is not compatible with the NMDA architecture, and providing a pointer to ietf-routing-2 ... is not an automatic way... so barely useful). And second, to ask them to change their import (or service composition) to ietf-routing-2. Hopefully, in the ietf-routing case, most dependent YANG modules are part of the IETF, so the communication is a manageable. For the already existing dependent vendor modules the problem is worse.

Changing the ietf-interfaces YANG module name would be a different challenge, as it’s used throughout the industry: <https://www.yangcatalog.org/yang-search/impact_analysis.php?modules[]=ietf-interfaces&recurse=0&rfcs=1&show_subm=1&show_dir=dependents>

2.7. Need to Allow Non-Backward Compatible changes

As described in the previous sections, there is a need to allow non-backward compatible changes without changing a module’s name. This would avoid many of the above problems. In most cases even after non-backward compatible updates a module should keep its name. However, for really major changes renaming the module is still the proper way to go:

when splitting a module into two separate modules

when removing 80% of a module's schema

when a standard module is moved from one organization to another (e.g., from ietf to ieee)

when a company's name is changed and new versions of the module are renamed to reflect that

Allowing non-backward compatible changes to happen without a module name change will decrease the number of separate modules to handle and will make it a trivial task to track these non-backward compatible changes.

2.8. Problematic Handling of Status Statement

The current definition of deprecated and obsolete in [RFC7950] (as quoted below) is problematic and should be corrected.

- "deprecated" indicates an obsolete definition, but it permits new/ continued implementation in order to foster interoperability with older/existing implementations.

- "obsolete" means that the definition is obsolete and SHOULD NOT be implemented and/or can be removed from implementations.

YANG is considered an interface contract between the server and the client. The current definitions of deprecated and obsolete mean that a schema node that is either deprecated or obsolete may or may not be implemented. The client has no way to find out which is the case except for by trying to write or read data at the leaf in question. This probing would need to be done for each separate data-node, which not a trivial thing to do. This "may or may not" is unacceptable in a contract. In effect, this works as if there would be an if-feature statement on each deprecated schema node where the server does not advertise whether the feature is supported or not. Why is it not advertised?

2.9. No way to easily decide whether a change is Backward Compatible

A management system, SDN controller or any other user of a module should be capable of easily determining the compatibility between two module versions. Higher level logic for a network function, something that can not be implemented in a purely model driven way, is always dependent on a specific version of the module. If the client finds that the module has been updated on the network node, it has to decide if it tries to handle it as it handled the previous

version of the model or if it just stops, to avoid problems. To make this decision the client needs to know if the module was updated in a backward compatible way or not.

This is not possible to decide today because of the following:

- It is possible to change the semantic behavior of a data node, action or rpc while the YANG definition does not change (with the possible exception of the description statement). In such a case it is impossible to determine whether the change is backward compatible just by looking at the YANG statements. Its only the human model designer that can decide.

- Problems with the status statement, Section 2.8

- Modelers might decide to violate YANG 1.1 update rules for some of the reasons above

Finding status changes or violations of update rules need a line by line comparision of the old and new modules, no easy task.

2.10. Early Warning about Removal

If a schema part is considered old/bad we need to be able to give advance warning that it will be removed. As this is an advance warning the part shall still be present and usable in the current revision; however, it will be removed in one of the next revisions. We need the advance warning to allow users of the module time enough to plan/execute migration away from the deprecated functionality. Often deprecation will be accompanied by information whether the functionality will just disappear or that there is an alternative, possibly more advanced solution that should be used.

Vendors use such warnings often, but the NMDA related redesign of IETF modules is also an example where it would be useful. (As another example see the usage of deprecated in the Java programing language.) The current definition of the deprecated status does not serve this purpose as described in Section 2.8. The definition of "deprecated" in the status statement shall be changed to address this issue.

3. The Solution

The solution is composed of four parts, a semantic versioning YANG extension, updates to the YANG 1.1. status statement and module update rules and the import by version statement. An optional additional check, validating the semantic versioning from a syntactic point of view, can either assist in determining the correct semantic

versioning values, or can help in determining the values for YANG modules that don’t support this extension.

3.1. SEMVER Semantic Versioning

The semantic versioning solution proposed here has already been proposed in [I-D.openconfig-netmod-model-catalog] (included here with the authors permission)which itself is based on [openconfigsemver]. The goal is to indicate the YANG module backwards (in)compatibility, following semver.org semantic versioning [semver]:

"The SEMVER version number for the module is introduced. This is expressed as a semantic version number of the form: x.y.z

- x is the MAJOR version. It is incremented when the new version of the specification is incompatible with previous versions.
- y is the MINOR version. It is incremented when new functionality is added in a manner that is backward-compatible with previous versions.
- z is the PATCH level. It is incremented when bug fixes are made in a backward-compatible manner.

Along these lines, we propose the following YANG 1.1 extension for a more generic semantic version. The formal definition is found at the end of this document.

```
extension module-version {
    argument "semver" {
        yin-element false;
    }
}
```

The extension would typically be used this way:

```
       module yang-module-name {

         namespace "name-space";
         prefix "prefix-name";

         import ietf-semver { prefix "semver"; }

         description
           "to be completed";


         revision 2017-10-30 {
           description
             "Change the module structure";
           semver:module-version "2.0.0";
         }

         revision 2017-07-30 {
           description
             "Added new feature XXX";
           semver:module-version "1.2.0";";
         }

         revision 2017-04-03 {
           description
             "Update copyright notice.";
           semver:module-version "1.0.1";;
         }

         revision 2017-04-03 {
           description
             "First release version.";
           semver:module-version "1.0.0";;
         }

         revision 2017-01-26 {
           description
             "Initial module for inet types";
           semver:module-version "0.1.0";;
         }


         //YANG module definition starts here
```

See also "Semantic Versioning and Structure for IETF Specifications" [I-D.claise-semver] for a mechanism to combine the semantic versioning, the github tools, and a potential change to the IETF process.

3.2. Updates to YANG 1.1 status statement

RFC 7950 section 11, must be updated to change the definition of deprecated and obsolete. In both cases the client must be able to determine whether the relevant parts are implemented or not without probing. The following definition is proposed:

- Deprecated schema nodes MUST still work as defined. The deprecated status serves only as a a warning that the schema node will be removed or obsoleted in the future.
- Obsolete schema nodes MUST be removed from the implementation. Requests concerning these schema nodes MUST be rejected with:
  - error-tag: operation-failed
  - error-app-tag: obsolete element

If there is a need to allow the server to decide whether a deprecated/obsolete schema part is implemented YANG already has a facility for that: the feature statement. Use it!

3.3. Updating the YANG 1.1 Module Update rules

RFC 7950 section 11, must be updated to express:

"As experience is gained with a module, it may be desirable to revise that module. Changes to published modules are allowed, even if they have some potential to cause interoperability problems, if the module-version YANG extension is used in the revision statement to clearly indicate the nature of the change."

3.4. The Derived Semantic Version

The YANG catalog contains not only the most up-to-date YANG module revision of a given module, but keeps all previous revisions as well. With APIs in mind, it’s important to understand whether different YANG module revisions are backward compatible (this is specifically imported for native YANG modules, i.e. the ones where generated-from = native), even for the YANG modules that don’t support the YANG extension specified in this document.

Two distinct leaves in the YANG module [I-D.clacla-netmod-model-catalog] contain this semver notation:

- the semantic-version leaf contains the value embedded within a YANG module (if it is available).

- the derived-semantic-version leaf is established by examining the the YANG module themselves. As such derived-semantic-version only takes syntax into account as opposed to the meaning of various elements when it computes the semantic version.

- The algorithm used to produce the derived-semantic-version is as follows:

  1. Order all modules of the same name by revision from oldest to newest. Include module revisions that are not available, but which are defined in the revision statements in one of the available module versions.

  2. If module A, revision N+1 has failed compilation, bump its derived semantic MAJOR version. For unavailable module versions assume non-backward compatible changes were done., thus bump its derived semantic MAJOR version.

  3. Else, run "pyang --check-update-from" on module A, revision N and revision N+1 to see if backward-incompatible changes exist.

  4. If backward-incompatible changes exist, bump module A, revision N+1's derived MAJOR semantic version.

  5. If no backward-incompatible changes exist, compare the pyang trees of module A, revision N and revision N+1.

  6. If there are structural differences (e.g., new nodes), bump module A, revision N+1's derived MINOR semantic version.

  7. If no structural differences exist, bump module A, revision N+1's derived PATCH semantic version.

Note that the absolute numbers in the semantic-version and derived-semantic-version are actually meaningless by themselves. That is, one must compare two different semver values for a given module to understand the compatibility between them.

3.5. Import by Semantic Version

If a module is imported by another one, it is usually not specified which version of the imported module should be used. However not all versions may be acceptable. Today YANG 1.1 allows us to specify the revision date of the imported module, but that is too specific, as even a small spelling correction of the imported module results in a change to its revision date, thus making the module revision ineligible for import.

Using semantic versioning to indicate the acceptable imported module versions is much more flexible.

- We might indicate that any compatible module-version after e.g. 3.1.0 is acceptable

- We might indicate that any compatible module-version of the 3.1.0, 4.0.0 or 5.0.0 major versions is acceptable. Later depending on updates in the 6.0.0 series we might allow those revisions also to be imported. As an non-backward compatible change in the 6.0.0 line might just change a small part of the imported module, the non-backward compatible changes may or may not affect the importer.

The module-version statement SHOULD be a substatement of the import statement. An import statement MUST NOT contain both a module-version and a revision substatement. The use of the revision substatement of import should be discouraged/deprecated.

4. Open Issues

There are a number of open issues to be disccused. These include the following:

- Do we need include-by-Semver?

- Should IETF/IANA officially generate derived semantic versions for their own modules? As they are the owner of the modules it should be their responsibility, but how to document it?

- There are cases where the module’s name should be changed but we still want to formally document the connection between the old and the new module names. For these cases shall we introduce a new YANG extension statement

- "replaces-module ietf-vlan;" ?

- We could consider a new naming convention for module files. Today, module files are named using a module@revision.yang notation. We could consider a module%semver.yang variant. Re-using the ’@’ for version is not ideal, so another separator character should be used. In this manner, both version and revision could be used.

- Taking another page from Openconfig, the notion of a module bundle could be considered. That is, there may need to be a way to enumerate modules that are part of a bundle and are known to

interoperate. This may not be as critical if a rich import-by-version is defined.
While the issue is interesting, it will be not be handled in this document.

- Similarly, the concept of a feature bundle should be considered. Typically, operators combine and test YANG modules to build value-add services. These bundles form releases for specific features or services, and it is critical to ensure as the modules evolve, the bundles can coherently evolve with them.
While the issue is interesting, it will be not be handled in this document.

- When we'll start using this new procedure for a new YANG module revision, will we have to update all the dependent YANG modules to start using this new procedure, along with the new import statement? Is this a moot point, as a new YANG module name would suffer from the same symptoms?
We see no need for updating other dependent modules. It is a good idea to update them, as they will benefit from using SEMVER, however there is no specific need to update them.

5. Semantic Version Extension YANG Module

The extension described in this module is defined in the YANG module below.

```
<CODE BEGINS> file "ietf-semver@2017-12-15.yang"
  module ietf-semver {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-semver";
    prefix semver;

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

       Author:   Benoit Claise
                 <mailto:bclaise@cisco.com>

       Author:   Joe Clarke
                 <mailto:jclarke@cisco.com>

       Author:   Kevin D'Souza
                 <mailto:kd6913@att.com>
```

```
       Author:   Balazs Lengyel
                 <mailto:balazs.lengyel@ericsson.com>";
    description
      "This module contains a definition for a YANG 1.1 extension to
       express the semantic version of YANG modules.";

    revision 2017-12-15 {
      description
        "Initial revision.";
      reference "draft-clacla-netmod-yang-model-update:
                 New YANG Module Update Procedure";
      semver:module-semver 0.1.1;
    }

    extension module-version {
      argument semver;
      description
        "The version number for the module revision it is used in.
        This is expressed as a semantic version string in the form:
          x.y.z
         where:
          * x corresponds to the major version,
          * y corresponds to a minor version,
          * z corresponds to a patch version.

         A major version number of 0 indicates that this model is still
         in development, and is potentially subject to change.

         Following a release of major version 1, all modules will
         increment major revision number where backwards incompatible
         changes to the model are made.

         The minor version is changed when features are added to the
         model that do not impact current clients use of the model.
         When major version is stepped, the minor version is reset to 0.

         The patch-level version is incremented when non-feature changes
         (such as bugfixes or clarifications to human-readable
         descriptions that do not impact model functionality) are made
         that maintain backwards compatibility.
         When major or minor version is stepped, the patch-level is
         reset to 0.

         The version number is stored in the module meta-data.

         By comparing the module-version between two revisions of a
         given module, one can know if revision N+1 is backwards
         compatible or not relative to revision N, as well as
```

```
          whether or not new features have been added to revision N+1.

          If a module contains this extension it indicates that for this
          module the updated status and update rules as this described in
          RFC XXXX are used.

          The statement MUST only be a substatement of the revision,
          import or include statements.
          Zero or One module-version statement is allowed per parent
          statement. NO substatements are allowed.
          ";

        reference "http://semver.org/ : Semantic Versioning 2.0.0";
      }

      augment /yanglib:modules-state/yanglib:module {
          leaf module-version {
              type string {
                  pattern "[0-9]+.[0-9]+.[0-9]+";
              }
          }
      }

      augment /yanglib:modules-state/yanglib:module/yanglib:submodule {
          leaf submodule-version {
              type string {
                  pattern "[0-9]+.[0-9]+.[0-9]+";
              }
          }
      }

    }
<CODE ENDS>
```

6. Contributors

- Anees Shaikh, Google
- Rob Shakir, Google

7. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

8. IANA Considerations

No IANA action is requested.

9. References

9.1. 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/info/rfc7950>.

9.2. Informative References

[I-D.clacla-netmod-model-catalog]
Clarke, J. and B. Claise, "YANG module for yangcatalog.org", draft-clacla-netmod-model-catalog-02 (work in progress), October 2017.

[I-D.claise-semver]
Claise, B., Barnes, R., and J. Clarke, "Semantic Versioning and Structure for IETF Specifications", draft-claise-semver-01 (work in progress), July 2017.

[I-D.openconfig-netmod-model-catalog]
Shaikh, A., Shakir, R., and K. D'Souza, "Catalog and registry for YANG models", draft-openconfig-netmod-model-catalog-02 (work in progress), March 2017.

[I-D.wu-l3sm-rfc8049bis]
Wu, Q., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", draft-wu-l3sm-rfc8049bis-11 (work in progress), December 2017.

[openconfigsemver]
"Semantic Versioning for Openconfig Models", <http://www.openconfig.net/docs/semver/>.

[RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <https://www.rfc-editor.org/info/rfc8049>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

Authors' Addresses

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

Joe Clarke
Cisco Systems, Inc.
7200-12 Kit Creek Rd
Research Triangle Park, North Carolina
United States of America

Phone: +1-919-392-2867
Email: jclarke@cisco.com

Balazs Lengyel
Ericsson
Magyar Tudosok Korutja
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
United States of America

Email: kd6913@att.com

Network Working Group
Internet-Draft
Updates: 7950 (if approved)
Intended status: Standards Track
Expires: January 3, 2019

B. Claise
J. Clarke
Cisco Systems, Inc.
B. Lengyel
Ericsson
K. D'Souza
AT&T

July 2, 2018

New YANG Module Update Procedure
draft-clacla-netmod-yang-model-update-06

Abstract

This document specifies a new YANG module update procedure in case of backward-incompatible changes, as an alternative proposal to the YANG 1.1 specifications. This document updates RFC 7950.

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 January 3, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document puts forth a solution to the problems described in [I-D.verdt-netmod-yang-versioning-reqs] by proposing changes to [RFC7950] to address the various requirements that [I-D.verdt-netmod-yang-versioning-reqs] specifies. At this time, the solution herein addresses requirements 1.1, 1.2, 1.3, 2.1, 4.1, 4.2, 4.3, 5.1, and 5.2. Current gaps are documented in Appendix A.1 below.

2. The Solution

The solution is composed of five parts:

1. A semantic versioning YANG extension, along with an optional additional check that validates the semantic versioning from a syntactic point of view, which can either assist in determining the correct semantic versioning value, or which can help in

determining the values for YANG modules that do not support this extension.

2. An import by semantic version statement

3. Updates to the YANG 1.1 module update rules

4. Updates to ietf-yang-library

5. An introduction of deprecated and obsolete reason clauses

2.1. Semantic Versioning

2.1.1. Semantic Versioning, As Set by the YANG Module Designer

The semantic versioning solution proposed here has already been proposed in [I-D.openconfig-netmod-model-catalog] (included here with the authors' permission) which itself is based on [openconfigsemver]. The goal is to indicate the YANG module backward (in)compatibility, following semver.org semantic versioning [semver]:

"The SEMVER version number for the module is introduced. This is expressed as a semantic version number of the form: x.y.z

- x is the MAJOR version. It is incremented when the new version of the specification is incompatible with previous versions.

- y is the MINOR version. It is incremented when new functionality is added in a manner that is backward-compatible with previous versions.

- z is the PATCH version. It is incremented when bug fixes are made in a backward-compatible manner."

The semantic version value is set by the YANG module developer at the design and implementation times. Along these lines, we propose the following YANG 1.1 extension for a more generic semantic version. The formal definition is found at the end of this document. This semantic version extension and the text below address requirements 1.1, 1.2, 2.1, 5.1 and 5.2 of [I-D.verdt-netmod-yang-versioning-reqs].

```
extension module-version {
    argument semver;
}
```

The extension would typically be used this way:

```
       module yang-module-name {

         namespace "name-space";
         prefix "prefix-name";

         import ietf-semver { prefix "semver"; }

         description
           "to be completed";


         revision 2017-10-30 {
           description
             "Change the module structure";
           semver:module-version "2.0.0";
         }

         revision 2017-07-30 {
           description
             "Added new feature XXX";
           semver:module-version "1.2.0";
         }

         revision 2017-04-03 {
           description
             "Update copyright notice.";
           semver:module-version "1.0.1";
         }

         revision 2017-04-03 {
           description
             "First release version.";
           semver:module-version "1.0.0";
         }

         revision 2017-01-26 {
           description
             "Initial module for inet types";
           semver:module-version "0.1.0";
         }


         //YANG module definition starts here
```

See also "Semantic Versioning and Structure for IETF Specifications" [I-D.claise-semver] for a mechanism to combine the semantic versioning, the GitHub tools, and a potential change to the IETF process.

2.1.2. The Derived Semantic Version

If an explicitly defined semantic version is not available in the YANG module, it is possible to algoritmically calculate a derived semantic version. This can be used for modules not containing a definitive semantic-version as defined in this document or as a starting value when specifying the definitive semantic-version. Be aware that this algorithm may sometimes incorrectly classify changes between the categories non-compatible, compatible or error-correction.

2.1.3. Implementation Experience

[yangcatalog] uses the pyang utility to calculate the derived-semantic-version for all of the modules contained within the catalog. [yangcatalog] contains many revisions of the same module in order to provide its derived-semantic-version for module consumers to know what has changed between revisions of the same module.

Two distinct leafs in the YANG module [I-D.clacla-netmod-model-catalog] contain this semver notation:

- the semantic-version leaf contains the value embedded within a YANG module (if it is available).
- the derived-semantic-version leaf is established by examining the the YANG module themselves. As such derived-semantic-version only takes syntax into account as opposed to the meaning of various elements when it computes the semantic version.
- The algorithm used to produce the derived-semantic-version is as follows:
  1. Order all modules of the same name by revision from oldest to newest. Include module revisions that are not available, but which are defined in the revision statements in one of the available module versions.
  2. If module A, revision N+1 has failed compilation, bump its derived semantic MAJOR version. For unavailable module versions assume non-backward compatible changes were done., thus bump its derived semantic MAJOR version.
  3. Else, run "pyang --check-update-from" on module A, revision N and revision N+1 to see if backward-incompatible changes exist.

4. If backward-incompatible changes exist, bump module A, revision N+1's derived MAJOR semantic version.

5. If no backward-incompatible changes exist, compare the pyang trees of module A, revision N and revision N+1.

6. If there are structural differences (e.g., new nodes), bump module A, revision N+1's derived MINOR semantic version.

7. If no structural differences exist, bump module A, revision N+1's derived PATCH semantic version.

The pyang utility checks many of the points listed in section 11 of [RFC7950] for known module incompatibilities. While this approach is a good way to programmatically obtain a semantic version number, it does not address all cases whereby a major version number might need to be increased. For example, a node may have the same name and same type, but its meaning may change from one revision of a module to another. This represents a semantic change that breaks backward compatibility, but the above algorithm would not find it. Therefore, additional, sometimes manual, rigor must be done to ensure a proper version is chosen for a given module revision.

2.2. Import by Semantic Version

If a module is imported by another one, it is usually not specified which revision of the imported module should be used. However, not all revisions may be acceptable. Today YANG 1.1 allows one to specify the revision date of the imported module, but that is too specific, as even a small spelling correction of the imported module results in a change to its revision date, thus making the module revision ineligible for import.

Using semantic versioning to indicate the acceptable imported module versions is much more flexible. For example:

- Only a module of a specific MAJOR version is acceptable. All MINOR and PATCH versions can also be imported.

- A module at a specific MAJOR version or higher is acceptable.

- A module at a specific MAJOR.MINOR version is acceptable. All PATCH versions can also be imported.

- A module within a certain range of versions are acceptable. For example, in this case, a module between version 1.0.0 (inclusive) and 3.0.0 (exclusive) are acceptable.

The ietf-semver module provides another extension, import-versions that is a child of import and specifies the rules for an acceptable set of versions of the given module. This extension addresses requirement 1.3 of [I-D.verdt-netmod-yang-versioning-reqs]. The structure of this extension is specified as follows:

TODO: How to specify this? One thought is below, not fully formalized as this should be discussed further. Note: while this uses a comma to separate discrete versions, we could instead allow for this to be specified multiple times.

```
[\[(]X[.Y[.Z]][-[X[.Y[.X]]][\])]][,...]
```

Where the first character MAY be a '[' or '(' to indicate at least inclusive and at least exclusive (respectively). If this is omitted, a full semantic version must be specified and the import will only support this one version.

The following version, if specified with a '[' or '(' indicates the lower bound. This can be a full semantic version or a MAJOR only or MAJOR.MINOR only.

The '-', if specified, is a literal hyphen indicating a range will be specified. If the second portion of the import-versions clause is omitted, then there is no upper bound on what will be considered an acceptable imported version.

After the '-' the upper bound semantic version (or part thereof) follows.

After the upper bound version, one of ']' or ')' MUST follow to indicate whether this limit is inclusive or exclusive of the upper bound respectively.

Finally, a literal comma (',') MAY be specified with additional ranges. Each range is taken as a logical OR.

For example:

```
import example-module {
  semver:import-versions "[1.0.0-3.0.0)";
  // All versions between 1.0.0 (inclusive) and 3.0.0 (exclusive) are acceptable
.
}

import example-module {
  semver:import-versions "[2-5]";
  // All versions between 2.0.0 (inclusive) and 5.y.z (inclusive) where y and z
are
  // any value for MINOR and PATCH versions.
}

import example-module {
  semver:import-versions "[1.5-2.0.0),[2.5";
  // All versions between 1.5.0 (inclusive) and 2.0.0 (exclusive) as well as all
 versions
  // greater than 2.5 (inclusive).  In this manner, if 2.0 was branched from 1.4
, and a
  // new feature was added into 1.5, all versions of 1.x.x starting at 1.5 are a
llowed,
  // but the feature was not merged into 2.y.z until 2.5.0.
}

import example-module {
  semver:import-versions "[1";
  // All versions greater than MAJOR version 1 are acceptable.  This includes an
y
  // MINOR or PATCH versions.
}

import example-module {
  semver:import-versions "1.0.0";
  // Only version 1.0.0 is acceptable (this mimics what exists with import by re
vision).
}

import example-module {
  semver:import-versions "[1.1-2)"";
  // All versions greater than 1.1 (inclusive, and including all PATCH versions
off of 1.1)
  // up to MAJOR version 2 (exclusive) are acceptable.
}

import example-module {
  semver:import-versions "[1.1-2),[3";
  // All versions greater than 1.1 (inclusive, and including all PATCH versions
off of 1.1)
  // up to MAJOR version 2 (exclusive), as well as all versions greater than MAJ
OR version 3
  // (inclusive) are acceptable.
}

import example-module {
  semver:import-versions "[1.1-2),[3.0.0";
  // This is equivalent to the example above, simply indicating that a partial s
emantic version
  // assumes all missing components are 0.
}
```

The import statement SHOULD include a semver:import-versions statement and MUST NOT include a revision statement. An import statement MUST NOT contain both a semver:import-versions and a revision substatement. The use of the revision substatement for import should be discouraged.

2.3. Updates to YANG 1.1 Module Update Rules

RFC 7950 section 11, must be updated to allow for non-backward changes provided they follow the semantic versioning guidelines and increase the MAJOR version number when a backward incompatible change is made. This change is in the spirit of requirement 5.1 from [I-D.verdt-netmod-yang-versioning-reqs]. The following is proposed text for this change.

"As experience is gained with a module, it may be desirable to revise that module. Changes to published modules are allowed, even if they have some potential to cause interoperability problems, if the module-version YANG extension is used in the revision statement to clearly indicate the nature of the change."

2.4. Updates to ietf-yang-library

The ietf-semver YANG module also specifies additional ietf-yang-library [RFC7895] [I-D.ietf-netconf-rfc7895bis] leafs to be added at the module and submodule levels. The first is module-version, which augments /yanglib:yang-library/yanglib:module-set/yanglib:module. This specifies the current semantic version of the associated module and revision in a given module-set. The related submodule-version leaf is added at /yanglib:yang-library/yanglib:module-set/yanglib:module/yanglib:submodule to indicate the semantic version of a submodule.

In order to satisfy the requirements 4.1 and 4.3 of [I-D.verdt-netmod-yang-versioning-reqs] that deprecated and obsolete node presence and operation are easily and clearly known to clients, ietf-semver also augments the ietf-yang-library with two additional boolean leafs at /yanglib:yang-library/yanglib:module-set/yanglib:module. A client can make one request of the ietf-yang-library and know whether or a not a module that has deprecated or obsolete has those nodes implemented by the server, as opposed to making multiple requests for each node in question.

deprecated-nodes-present : A boolean that indicates whether or not this server implements deprecated nodes. The value of this leaf SHOULD be true; and if so, the server MUST implement nodes within this module as they are documented. If specific deprecated nodes

are not implemented as documented, then they MUST be listed as deviations. This leaf defaults to true.

obsolete-nodes-present : A boolean that indicates whether or not this server implements obsolete nodes. The value of this leaf SHOULD be false; and if so, the server MUST NOT implement nodes within this module. If this leaf is true, then all nodes in this module MUST be implemented as documented in the module. Any variation of this MUST be listed as deviations. This leaf defaults to false.

If a module does not have any deprecated or obsolete nodes, the server SHOULD set the corresponding leaf above to true. This is helpful to clients, such that if the MAJOR version number has not changed, and these booleans are true, then a client does not have to check the status of any node for the module.

Module compatibility can be affected if values other than the default are used for the leafs described here. For example, if a server does not implemennt deprecated nodes, then a given module revision may be incompatible with a previous revision where the nodes were not deprecated. When calculating backward compatibility, the default values of these leafs MUST be considered. From a client's point of view, if two module revisions have the same MAJOR version but the run-time value of deprecated-nodes-present (as read from the ietf-yang-library) is false, then compatibility MUST NOT be assumed based on the module-version alone.

2.5. Deprecated and Obsolete Reasons

The ietf-semver module specifies an extension, status-description, that is designed to be used as a substatement of the status statement when the status is deprecated or obsolete. The argument to this extension is freeform text that explains why the node was deprecated or made obsolete. It may also point to other schema elements that take the place of the deprecated or obsolete node. This text is designed for human consumption to aid in the migration away from nodes that will one day no longer work. These extensions address requirement 4.2 of [I-D.verdt-netmod-yang-versioning-reqs]. An example is shown below.

```
   leaf imperial-temperature {
     type int64;
     units "degrees Fahrenheit";
     status deprecated {
       semver:status-description
         "Imperial measurements are being phased out in favor
          of their metric equivalents.  Use metric-temperature
          instead.";
     }
     description
       "Temperature in degrees Fahrenheit.";
   }
```

3. Semantic Version Extension YANG Module

The extension and related ietf-yang-library changes described in this module are defined in the YANG module below.

```
<CODE BEGINS> file "ietf-semver@2018-04-05.yang"
  module ietf-semver {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-semver";
    prefix semver;

    import ietf-yang-library {
      prefix yanglib;
    }

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

       Author:   Benoit Claise
                 <mailto:bclaise@cisco.com>

       Author:   Joe Clarke
                 <mailto:jclarke@cisco.com>

       Author:   Kevin D'Souza
                 <mailto:kd6913@att.com>

       Author:   Balazs Lengyel
                 <mailto:balazs.lengyel@ericsson.com>";
    description
      "This module contains a definition for a YANG 1.1 extension to
       express the semantic version of YANG modules.";
```

```
    revision 2018-04-05 {
      description
        "* Properly import ietf-yang-library.
         * Fix the name of module-semver => module-version.
         * Fix regular expression syntax.
         * Augment yang-library with booleans as to whether or not
           deprecated and obsolete nodes are present.
         * Add an extension to enable import by semantic version.
         * Add an extension status-description to track deprecated
           and obsolete reasons.
         * Fix yang-library augments to use 7895bis.";
      reference
        "draft-clacla-netmod-yang-model-update:
         New YANG Module Update Procedure";
      semver:module-version "0.2.1";
    }
    revision 2017-12-15 {
      description
        "Initial revision.";
      reference
        "draft-clacla-netmod-yang-model-update:
         New YANG Module Update Procedure";
      semver:module-version "0.1.1";
    }

    extension module-version {
      argument semver;
      description
        "The version number for the module revision it is used in.
         This is expressed as a semantic version string in the form:
          x.y.z
         where:
          * x corresponds to the major version,
          * y corresponds to a minor version,
          * z corresponds to a patch version.

         A major version number of 0 indicates that this model is still
         in development, and is potentially subject to change.

         Following a release of major version 1, all modules will
         increment major revision number where backward incompatible
         changes to the model are made.

         The minor version is changed when features are added to the
         model that do not impact current clients use of the model.
         When major version is stepped, the minor version is reset to 0.

         The patch-level version is incremented when non-feature changes
```

```
          (such as bugfixes or clarifications to human-readable
          descriptions that do not impact model functionality) are made
          that maintain backward compatibility.
          When major or minor version is stepped, the patch-level is
          reset to 0.

          By comparing the module-version between two revisions of a
          given module, one can know if different revisions are backward
          compatible or not, as well as
          whether or not new features have been added to a newer revision.

          If a module contains this extension it indicates that for this
          module the updated status and update rules as this described in
          RFC XXXX are used.

          The statement MUST only be a substatement of the revision statement.
          Zero or one module-version statement is allowed per parent
          statement. NO substatements are allowed.
        ";
      reference "http://semver.org/ : Semantic Versioning 2.0.0";
    }

    extension import-versions {
      argument version-clause;
      description
        "This extension specifies an acceptable set of semantic versions of a gi
ven module
         that may be imported.  The version-clause argument is specified in the
following
         format

         [\\[(]X[.Y[.Z]][-[X[.Y[.X]]][\\])]][,...]

         Where the first character MAY be a '[' or '(' to indicate at least incl
usive and at least
          exclusive (respectively).  If this is omitted, a full semantic version
 must be specified
          and the import will only support this one version.

         The following version, if specified with a '[' or '(' indicates the low
er bound.  This can
          be a full semantic version or a MAJOR only or MAJOR.MINOR only.

         The '-', if specified, is a literal hyphen indicating a range will be s
pecified.  If the second portion
          of the import-versions clause is omitted, then there is no upper bound
 on what will be considered
          an acceptable imported version.

         After the '-' the upper bound semantic version (or part thereof) follow
s.
         After the upper bound version, one of ']' or ')' MUST follow to indicat
e whether this limit is inclusive
          or exclusive of the upper bound respectively.

         Finally, a literal comma (',') MAY be specified with additional ranges.
  Each range is taken as a logical
          OR.
```

```
         The statement MUST only be a substatement of the import statement.  Zer
o or one
         import-versions statement is allowed per import statement.  NO substate
ments are allowed.";
      reference "I-D.clacla-netmod-yang-model-update : Import By Semantic Versio
n";
    }

    extension status-description {
      argument description;
      description
        "Freeform text that describes why a given node has been deprecated or ma
de obsolete.
         This may point to other schema elements that can be used in lieu of the
 given node.

         This statement MUST only be used as a substatement of the status statem
ent, and MUST
         only be used when the status is deprecated or obsolete.  Zero or more s
tatus-description
         statements are allowed per parent statement.  NO substatements are allo
wed.";
      reference "I-D.clacla-netmod-yang-model-update : Deprecated and Obsolete R
easons";
    }

    augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
      description
        "Augmentations for the ietf-yang-library module to support semantic vers
ioning.";
      leaf module-version {
        type string {
          pattern '[0-9]+\.[0-9]+\.[0-9]+';
        }
        description
          "The semantic version for this module in MAJOR.MINOR.PATCH format.  Th
is version
           must match the semver:module-version value in specific revision of th
e module
           loaded in this module-set.";
      }
      leaf deprecated-nodes-present {
        type boolean;
        default "true";
        description
          "A boolean that indicates whether or not this server implements deprec
ated nodes.
           The value of this leaf SHOULD be true; and if so, the server MUST imp
lement nodes
           within this module as they are documented.  If specific deprecated no
des are not
           implemented as document, then they MUST be listed as deviations.  If
a module does
           not currently contain any deprecated nodes, then this leaf SHOULD be
set to true.";
      }
      leaf obsolete-nodes-present {
        type boolean;
        default "false";
        description
          "A boolean that indicates whether or not this server implements obsole
te nodes.
           The value of this leaf SHOULD be false; and if so, the server MUST NO
```

T implement
          nodes within this module. If this leaf is true, then all nodes in thi
s module MUST
          be implemented as documented in the module. Any variation of this MU
ST be listed as
          deviations. If a module does not currently contain any obsolete node
s, then this

```
            leaf SHOULD be set to true.";
        }
      }
      augment "/yanglib:yang-library/yanglib:module-set/yanglib:module/yanglib:sub
module" {
        description
          "Augmentations for the ietf-yang-library module/submodule to support sem
antic versioning.";
        leaf submodule-version {
          type string {
            pattern '[0-9]+\.[0-9]+\.[0-9]+';
          }
          description
            "The semantic version for this submodule in MAJOR.MINOR.PATCH format.
 This version
             must match the semver:module-version value in specific revision of th
e submodule
             loaded in this module-set.";
        }
      }
    }
<CODE ENDS>
```

4. Contributors

- Anees Shaikh, Google
- Rob Shakir, Google

5. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

6. IANA Considerations

6.1. YANG Module Registrations

The following YANG module is requested to be registred in the "IANA Module Names" registry:

The ietf-semver module:

- Name: ietf-semver
- XML Namespace: urn:ietf:params:xml:ns:yang:ietf-semver
- Prefix: semver
- Reference: [RFCXXXX]

7. References

7.1. Normative References

[I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-00 (work in progress), July 2018.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[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>.

7.2. Informative References

[I-D.clacla-netmod-model-catalog]
Clarke, J. and B. Claise, "YANG module for yangcatalog.org", draft-clacla-netmod-model-catalog-03 (work in progress), April 2018.

[I-D.claise-semver]
Claise, B., Barnes, R., and J. Clarke, "Semantic Versioning and Structure for IETF Specifications", draft-claise-semver-02 (work in progress), January 2018.

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-06 (work in progress), April 2018.

[I-D.openconfig-netmod-model-catalog]
Shaikh, A., Shakir, R., and K. D'Souza, "Catalog and registry for YANG models", draft-openconfig-netmod-model-catalog-02 (work in progress), March 2017.

[openconfigsemver]
"Semantic Versioning for Openconfig Models", <http://www.openconfig.net/docs/semver/>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

[yangcatalog]
"YANG Catalog", <https://yangcatalog.org>.

Appendix A. Appendix

A.1. Open Issues: Requirements to be Addressed

There are a few requirements of [I-D.verdt-netmod-yang-versioning-reqs] still to be addressed. These include the following:

- A solution is required for client compatibility to address requirements 3.1 and 3.2 from [I-D.verdt-netmod-yang-versioning-reqs]. This could include adding "module sets" support to ietf-yang-library where the client can choose one set with which to use.

- A solution for instance data to satisfy requirement 5.3 of [I-D.verdt-netmod-yang-versioning-reqs] is also required.

- While it is believed one could work within this semver scheme to support multiple parallel trains of development within a given YANG module, some thought should be given to how this would work in support of optional requirement 4.4 of [I-D.verdt-netmod-yang-versioning-reqs].

- While not mandatory, requirement 2.2 of [I-D.verdt-netmod-yang-versioning-reqs] looks to provide a way to determine, at the node level, whether or not changes have occurred between revisions of a given YANG module. This may require application of semver at the node level.

A.2. Open Issues

Additionally, there are a few open issues to be discussed and settled. These include the following:

- Do we need a new version of YANG?
  While eventually this will fold into a new version, the belief is this solution can work with extensions alone with an update to the [RFC7950] text concerning module updates.

- Should IETF/IANA officially generate derived semantic versions for their own modules? As they are the owner of the modules it should be their responsibility, but how to document it? Note that next round of funding for the yangcatalog.org could help develop the perfect derived-semantic-version toolset

- We could consider a new naming convention for module files. Today, module files are named using a module@revision.yang notation. We could consider module%semver.yang or

module#version.yang variants. Re-using the '@' for version is not ideal, so another separator character should be used. In this manner, both version and revision could be used.

Authors' Addresses

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

Joe Clarke
Cisco Systems, Inc.
7200-12 Kit Creek Rd
Research Triangle Park, North Carolina
United States of America

Phone: +1-919-392-2867
Email: jclarke@cisco.com

Balazs Lengyel
Ericsson
Magyar Tudosok Korutja
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
United States of America

Email: kd6913@att.com

Network Working Group
Internet-Draft
Updates: rfc6087bis (if approved)
Intended status: Standards Track
Expires: September 7, 2018

C. Hopps
Deutsche Telekom
L. Berger
LabN Consulting, L.L.C.
D. Bogdanovic

March 6, 2018

YANG Module Tags
draft-ietf-netmod-module-tags-01

Abstract

This document provides for the association of tags with YANG modules. The expectation is for such tags to be used to help classify and organize modules. A method for defining, reading and writing a modules tags is provided. Tags may be standardized and assigned during module definition; assigned by implementations; or dynamically defined and set by users. This document provides guidance to future model writers and, as such, this document updates [I-D.ietf-netmod-rfc6087bis].

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 September 7, 2018.

Copyright Notice

Table of Contents

1. Introduction

The use of tags for classification and organization is fairly ubiquitous not only within IETF protocols, but in the internet itself (e.g., #hashtags). Tags can be usefully standardized, but they can also serve as a non-standardized mechanism available for users to define themselves. Our solution provides for both cases allowing for the most flexibility. In particular, tags may be standardized as well as assigned during module definition; assigned by implementations; or dynamically defined and set by users.

This document defines a module which provides a list of module entries to allow for adding or removing of tags as well as viewing the set of tags associated with a module.

This document also defines an IANA registry for tag prefixes as well as a set of globally assigned tags.

Section 7 provides guidelines for authors of YANG data models. This section updates [I-D.ietf-netmod-rfc6087bis].

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Note that lower case versions of these key words are used in section Section 7 where guidance is provided to future document authors.

3. Tag Prefixes

All tags have a prefix indicating who owns their definition. An IANA registry is used to support standardizing tag prefixes. Currently 3 prefixes are defined with all others reserved.

3.1. IETF Standard Tags

An IETF standard tag is a tag that has the prefix "ietf:". All IETF standard tags are registered with IANA in a registry defined later in this document.

3.2. Vendor Tags

A vendor tag is a tag that has the prefix "vendor:". These tags are defined by the vendor that implements the module, and are not standardized; however, it is recommended that the vendor consider including extra identification in the tag name to avoid collisions (e.g., vendor:super-duper-company:...).

3.3. Local Tags

A local tag is any tag that has the prefix "local:". These tags are defined by the local user/administrator and will never be standardized.

3.4. Reserved Tags

Any tag not starting with the prefix "ietf:", "vendor:" or "local:" is reserved for future standardization.

4. Tag Management

Tags can become associated with a module in a number of ways. Tags may be defined and associated at model design time, at implementation time, or via user administrative control. As the main consumer of tags are users, users may also remove any tag, no matter how the tag became associated with a module.

4.1. Module Definition Association

A module definition SHOULD indicate a set of tags to be automatically added by the module implementer. These tags MUST be standard tags (Section 3.1). This does imply that new modules may also drive the addition of new standard tags to the IANA registry.

4.2. Implementation Association

An implementation MAY include additional tags associated with a module. These tags may be standard or vendor specific tags.

4.3. Administrative Tagging

Tags of any kind can be assigned and removed with normal configuration mechanisms.

Implementations MUST ensure that a modules tag list is consistent across any location from which the list is accessible. So if a user adds a tag through configuration that tag should also be seen when using any augmentation that exposes the modules tag list.

5. Tags Module Structure

5.1. Tags Module Tree

The tree associated with the tags module is:

```
module: ietf-module-tags
    +--rw module-tags* [name]
       +--rw name          yang:yang-identifier
       +--rw tag*          string
       +--rw masked-tag*   string
```

5.2. Tags Module

```
<CODE BEGINS> file "ietf-module-tags@2018-03-06.yang"
module ietf-module-tags {
  yang-version "1";
  namespace "urn:ietf:params:xml:ns:yang:ietf-module-tags";
```

```
  prefix "mtags";

  import ietf-yang-types {
    prefix yang;
  }

  organization "IETF NetMod Working Group (NetMod)";

  contact
    "NetMod Working Group - <netmod@ietf.org>";

  description
    "This module describes a tagging mechanism for yang module.
     Tags may be IANA assigned or privately defined types.";

  revision "2018-03-06" {
    description
      "Initial revision.";
    reference "TBD";
  }

  list module-tags {
    key "name";

    description
      "A list of modules and their associated tags";

    leaf name {
        type yang:yang-identifier;
        mandatory true;
        description
            "The YANG module or submodule name.";
    }

    leaf-list tag {
        type string;

        description
            "A tag associated with the module. See the IANA 'YANG Module
             Tag Prefix' registry for reserved prefixes and the IANA
             'YANG Module IETF Tag' registry for IETF standard tags.

             The operational view of this list will contain all
             user-configured tags as well as any predefined tags that
             have not been masked by the user using the masked-tag leaf
             list below.";
    }
```

```
    leaf-list masked-tag {
        type string;

        description
            "The list of tags that should not be associated with this
             module. This user can remove (mask) predefined tags by
             adding them to this list. It is not an error to add tags to
             this list that are not predefined for the module.";
    }
  }
}
<CODE ENDS>
```

6. Other Classifications

It’s worth noting that a different yang module classification document exists [RFC8199]. That document is classifying modules in only a logical manner and does not define tagging or any other mechanisms. It divides yang modules into 2 categories (service or element) and then into one of 3 origins: standard, vendor or user. It does provide a good way to discuss and identify modules in general. This document defines standard tags to support [RFC8199] style classification.

7. Guidelines to Model Writers

This section updates [I-D.ietf-netmod-rfc6087bis].

7.1. Define Standard Tags

A module SHOULD indicate, in the description statement of the module, a set of tags that are to be associated with it. This description should also include the appropriate conformance statement or statements, using [RFC2119] language for each tag.

```
module example-module {
  ...
  description
    "[Text describing the module...]

    RFC<this document> TAGS:
    The following tags MUST be included by an implementation:
         - ietf:some-required-tag:foo
         - ...
    The following tags SHOULD be included by an implementation:
         - ietf:some-recommended-tag:bar
         - ...
    The following tags MAY be included by an implementation:
         - ietf:some-optional-tag:baz
         - ...
      ";
  ...
}
```

One SHOULD only include conformance text if there will be tags listed (i.e., there's no need to indicate an empty set).

The module writer may use existing standard tags, or use new tags defined in the model definition, as appropriate. New tags should be assigned in the IANA registry defined below, see Section 8.2 below.

8. IANA Considerations

8.1. YANG Module Tag Prefix Registry

This registry allocates tag prefixes. All YANG module tags SHOULD begin with one of the prefixes in this registry.

The allocation policy for this registry is Specification Required [RFC5226].

The initial values for this registry are as follows.

prefix	description
ietf:	IETF Standard Tag allocated in the IANA YANG Module IETF Tag Registry.
vendor:	Non-standardized tags allocated by the module implementer.
local:	Non-standardized tags allocated by and for the user.

Other SDOs (standard organizations) wishing to standardize their own set of tags could allocate a top level prefix from this registry.

8.2. YANG Module IETF Tag Registry

This registry allocates prefixes that have the standard prefix "ietf:". New values should be well considered and not achievable through a combination of already existing standard tags.

The allocation policy for this registry is IETF Review [RFC5226].

The initial values for this registry are as follows.

[Editor's note: many of these tags may move to [I-D.ietf-rtgwg-device-model] if/when that document is refactored to use tags.]

Tag	Description	Reference
ietf:rfc8199:element	A module for a network element.	[RFC8199]
ietf:rfc8199:service	A module for a network service.	[RFC8199]
ietf:rfc8199:standard	A module defined by a standards organization.	[RFC8199]
ietf:rfc8199:vendor	A module defined by a vendor.	[RFC8199]
ietf:rfc8199:user	A module defined by the user.	[RFC8199]
ietf:device:hardware	A module relating to device hardware (e.g., inventory).	[This document]
ietf:device:software	A module relating to device software (e.g., installed OS).	[This document]
ietf:device:qos	A module for managing quality of service.	[This document]
ietf:protocol	A module representing a protocol.	[This document]
ietf:system-management	A module relating to system management (e.g., a system management protocol).	[This document]

ietf:network-service	A module relating to network service (e.g., a network service protocol).	[This document]
ietf:oam	A module representing Operations, Administration, and Maintenance.	[This document]
ietf:routing	A module related to routing.	[This document]
ietf:routing:rib	A module related to routing information bases.	[This document]
ietf:routing:igp	An interior gateway protocol module.	[This document]
ietf:routing:egp	An exterior gateway protocol module.	[This document]
ietf:signaling	A module representing control plane signaling.	[This document]
ietf:lmp	A module representing a link management protocol.	[This document]

Table 1: IETF Module Tag Registry

9. References

9.1. Normative References

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-18 (work in progress), February 2018.

[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>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

9.2. Informative References

[I-D.ietf-rtgwg-device-model] Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Logical Organization", draft-ietf-rtgwg-device-model-02 (work in progress), March 2017.

Authors' Addresses

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Dean Bogdanovic

Email: ivandean@gmail.com

Network Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: September 1, 2020

C. Hopps
LabN Consulting, L.L.C.
L. Berger
LabN Consulting, LLC.
D. Bogdanovic
Volta Networks

February 29, 2020

YANG Module Tags
draft-ietf-netmod-module-tags-10

Abstract

This document provides for the association of tags with YANG modules. The expectation is for such tags to be used to help classify and organize modules. A method for defining, reading and writing a modules tags is provided. Tags may be registered and assigned during module definition; assigned by implementations; or dynamically defined and set by users. This document also provides guidance to future model writers; as such, this document updates RFC8407.

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 September 1, 2020.

Copyright Notice

Table of Contents

1. Introduction

The use of tags for classification and organization is fairly ubiquitous not only within IETF protocols, but in the internet itself (e.g., "#hashtags"). One benefit of using tags for organization over a rigid structure is that it is more flexible and can more easily adapt over time as technologies evolve. Tags can be usefully registered, but they can also serve as a non-registered mechanism

available for users to define themselves. This document provides a mechanism to define tags and associate them with YANG modules in a flexible manner. In particular, tags may be registered as well as assigned during module definition; assigned by implementations; or dynamically defined and set by users.

This document defines a YANG module [RFC7950] which provides a list of module entries to allow for adding or removing of tags as well as viewing the set of tags associated with a module.

This document defines an extension statement to be used to indicate tags that SHOULD be added by the module implementation automatically (i.e., outside of configuration).

This document also defines an IANA registry for tag prefixes as well as a set of globally assigned tags.

Section 6 provides guidelines for authors of YANG data models.

This document updates [RFC8407].

The YANG data model in this document conforms to the Network Management Datastore Architecture defined in [RFC8342].

1.1. Some possible use cases for YANG module tags

During this documents's development there were requests for example uses of module tags. The following are a few example use cases for tags. This list is certainly not exhaustive.

One example use of tags would be to help filter different discrete categories of YANG modules supported by a device. For example, if modules are suitably tagged, then an XPath query can be used to list all of the vendor modules supported by a device.

Tags can also be used to help coordination when multiple semi-independent clients are interacting with the same devices. For example, one management client could mark that some modules should not be used because they have not been verified to behave correctly, so that other management clients avoid querying the data associated with those modules.

Tag classification is useful for users searching module repositories (e.g., YANG catalog). A query restricted to the 'ietf:routing' module tag could be used to return only the IETF YANG modules associated with routing. Without tags, a user would need to know the name of all the IETF routing protocol YANG modules.

Future management protocol extensions could allow for filtering queries of configuration or operational state on a server based on tags. For example, return all operational state related to system-management.

1.2. Conventions Used in This Document

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 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Tag Values

All tags SHOULD begin with a prefix indicating who owns their definition. An IANA registry (Section 7.1) is used to support registering tag prefixes. Currently 3 prefixes are defined. No further structure is imposed by this document on the value following the registered prefix, and the value can contain any YANG type 'string' characters except carriage-returns, newlines and tabs.

Again, except for the conflict-avoiding prefix, this document is not specifying any structure on (i.e., restricting) the tag values on purpose. The intent is to avoid arbitrarily restricting the values that designers, implementers and users can use. As a result of this choice, designers, implementers, and users are free to add or not add any structure they may require to their own tag values.

2.1. IETF Tags

An IETF tag is a tag that has the prefix "ietf:". All IETF tags are registered with IANA in a registry defined later in this document (Section 7.2).

2.2. Vendor Tags

A vendor tag is a tag that has the prefix "vendor:". These tags are defined by the vendor that implements the module, and are not registered; however, it is RECOMMENDED that the vendor include extra identification in the tag to avoid collisions such as using the enterpise or organization name following the "vendor:" prefix (e.g., vendor:example.com:vendor-defined-classifier).

2.3. User Tags

A user tag is any tag that has the prefix "user:". These tags are defined by the user/administrator and are not meant to be registered. Users are not required to use the "user:" prefix; however, doing so is RECOMMENDED as it helps avoid collisions.

2.4. Reserved Tags

Any tag not starting with the prefix "ietf:", "vendor:" or "user:" is reserved for future use. These tag values are not invalid, but simply reserved in the context of specifications (e.g., RFCs).

3. Tag Management

Tags can become associated with a module in a number of ways. Tags may be defined and associated at module design time, at implementation time, or via user administrative control. As the main consumer of tags are users, users may also remove any tag, no matter how the tag became associated with a module.

3.1. Module Definition Tagging

A module definition MAY indicate a set of tags to be added by the module implementer. These design time tags are indicated using the module-tag extension statement.

If the module is defined in an IETF standards track document, the tags MUST be IETF Tags (2.1). Thus, new modules can drive the addition of new IETF tags to the IANA registry defined in Section 7.2, and the IANA registry can serve as a check against duplication.

3.2. Implementation Tagging

An implementation MAY include additional tags associated with a module. These tags SHOULD be IETF Tags (i.e., registered) or vendor specific tags.

3.3. User Tagging

Tags of any kind, with or without a prefix, can be assigned and removed by the user using normal configuration mechanisms. In order to remove a tag from the operational datastore the user adds a matching "masked-tag" entry for a given module.

4. Tags Module Structure

4.1. Tags Module Tree

The tree associated with the "ietf-module-tags" module follows. The meaning of the symbols can be found in [RFC8340].

```
module: ietf-module-tags
  +--rw module-tags
     +--rw module* [name]
        +--rw name          yang:yang-identifier
        +--rw tag*          tag
        +--rw masked-tag*   tag
```

Figure 1: YANG Module Tags Tree Diagram

4.2. YANG Module

```
<CODE BEGINS> file "ietf-module-tags@2020-02-29.yang"
module ietf-module-tags {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-module-tags";
  prefix tags;

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NetMod Working Group (NetMod)";
  contact
    "WG Web:  <https://tools.ietf.org/wg/netmod/>
     WG List: <mailto:netmod@ietf.org>

     Author: Christian Hopps
             <mailto:chopps@chopps.org>

     Author: Lou Berger
             <mailto:lberger@labn.net>

     Author: Dean Bogdanovic
             <ivandean@gmail.com>";

  // RFC Ed.: replace XXXX with actual RFC number and
  // remove this note.

  description
    "This module describes a mechanism associating tags with YANG
```

```
        modules. Tags may be IANA assigned or privately defined.

        Copyright (c) 2019 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 Simplified 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
        (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
        for full legal notices.

        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.";

      // RFC Ed.: update the date below with the date of RFC publication
      // and RFC number and remove this note.

      revision 2020-02-29 {
        description
          "Initial revision.";
        reference "RFC XXXX: YANG Module Tags";
      }

      typedef tag {
        type string {
          length "1..max";
          pattern '[\S ]+';
        }
        description
          "A tag is a type 'string' value that does not include carriage
           return, newline or tab characters. It SHOULD begin with a
           registered prefix; however, tags without a registered prefix
           SHOULD NOT be treated as invalid.";
      }

      extension module-tag {
        argument tag;
        description
          "The argument 'tag' is of type 'tag'. This extension statement
           is used by module authors to indicate the tags that SHOULD be
```

```
          added automatically by the system. As such the origin of the
          value for the pre-defined tags should be set to 'system'
          [RFC8342].";
      }

      container module-tags {
        description
          "Contains the list of modules and their associated tags";
        list module {
          key "name";
          description
            "A list of modules and their associated tags";
          leaf name {
            type yang:yang-identifier;
            mandatory true;
            description
              "The YANG module name.";
          }
          leaf-list tag {
            type tag;
            description
              "Tags associated with the module. See the IANA 'YANG Module
               Tag Prefixes' registry for reserved prefixes and the IANA
               'IETF YANG Module Tags' registry for IETF tags.

               The 'operational' state [RFC8342] view of this list is
               constructed using the following steps:

               1) System tags (i.e., tags of 'system' origin) are added.
               2) User configured tags (i.e., tags of 'intended' origin)
               are added.
               3) Any tag that is equal to a masked-tag is removed.";
          }
          leaf-list masked-tag {
            type tag;
            description
              "The list of tags that should not be associated with this
               module. The user can remove (mask) tags from the
               operational state datastore [RFC8342] by adding them to
               this list. It is not an error to add tags to this list
               that are not associated with the module, but they have no
               operational effect.";
          }
        }
      }
    }
    <CODE ENDS>
```

Figure 2: Module Tags Module

5. Other Classifications

It is worth noting that a different YANG module classification document exists [RFC8199]. That document only classifies modules in a logical manner and does not define tagging or any other mechanisms. It divides YANG modules into two categories (service or element) and then into one of three origins: standard, vendor or user. It does provide a good way to discuss and identify modules in general. This document defines IETF tags to support [RFC8199] style classification.

6. Guidelines to Model Writers

This section updates [RFC8407].

6.1. Define Standard Tags

A module MAY indicate, using module-tag extension statements, a set of tags that are to be automatically associated with it (i.e., not added through configuration).

```
module example-module {
  //...
  import module-tags { prefix tags; }

  tags:module-tag "ietf:some-new-tag";
  tags:module-tag "ietf:some-other-tag";
  // ...
}
```

The module writer can use existing standard tags, or use new tags defined in the model definition, as appropriate. For IETF standardized modules new tags MUST be assigned in the IANA registry defined below, see Section 7.2.

7. IANA Considerations

7.1. YANG Module Tag Prefixes Registry

IANA is asked to create a new registry "YANG Module Tag Prefixes" grouped under a new "Protocol" category named "YANG Module Tags".

This registry allocates tag prefixes. All YANG module tags SHOULD begin with one of the prefixes in this registry.

Prefix entries in this registry should be short strings consisting of lowercase ASCII alpha-numeric characters and a final ":" character.

The allocation policy for this registry is Specification Required [RFC8126]. The Reference and Assignee values should be sufficient to identify and contact the organization that has been allocated the prefix.

The initial values for this registry are as follows.

Prefix	Description	Reference	Assignee
ietf:	IETF Tags allocated in the IANA IETF YANG Module Tags registry.	[This document]	IETF
vendor:	Non-registered tags allocated by the module implementer.	[This document]	IETF
user:	Non-registered tags allocated by and for the user.	[This document]	IETF

Other standards organizations (SDOs) wishing to allocate their own set of tags should allocate a prefix from this registry.

7.2. IETF YANG Module Tags Registry

IANA is asked to create a new registry "IETF YANG Module Tags" grouped under a new "Protocol" category "IETF YANG Module Tags". This registry should be included below "YANG Module Tag Prefixes" when listed on the same page.

This registry allocates tags that have the registered prefix "ietf:". New values should be well considered and not achievable through a combination of already existing IETF tags. IANA assigned tags must conform to Net-Unicode as defined in [RFC5198] and they shall not need normalization.

The allocation policy for this registry is IETF Review [RFC8126].

The initial values for this registry are as follows.

Tag	Description	Reference
ietf:network-element-class	[RFC8199] network element.	[RFC8199]
ietf:network-service-class	[RFC8199] network service.	[RFC8199]

ietf:sdo-defined-class	Module is defined by a standards organization.	[RFC8199]
ietf:vendor-defined-class	Module is defined by a vendor.	[RFC8199]
ietf:user-defined-class	Module is defined by the user.	[RFC8199]
ietf:hardware	Relates to hardware (e.g., inventory).	[This document]
ietf:software	Relates to software (e.g., installed OS).	[This document]
ietf:protocol	Represents a protocol (often combined with another tag to refine).	[This document]
ietf:qos	Relates to quality of service.	[This document]
ietf:network-service-app	Relates to a network service application (e.g., an NTP server, DNS server, DHCP server, etc).	[This document]
ietf:system-management	Relates to system management (e.g., a system management protocol such as syslog, TACAC+, SNMP, netconf, ...).	[This document]
ietf:oam	Relates to Operations, Administration, and Maintenance (e.g., BFD).	[This document]
ietf:routing	Relates to routing.	[This document]
ietf:security	Related to security.	[This document]
ietf:signaling	Relates to control plane signaling.	[This document]

ietf:link-management	Relates to link management.	[This document]

7.3. Updates to the IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registrations have been made:

URI:
urn:ietf:params:xml:ns:yang:ietf-module-tags

Registrant Contact:
The IESG.

XML:
N/A; the requested URI is an XML namespace.

URI:
urn:ietf:params:xml:ns:yang:ietf-module-tags-state

Registrant Contact:
The IESG.

XML:
N/A; the requested URI is an XML namespace.

7.4. Updates to the YANG Module Names Registry

This document registers two YANG modules in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration have been made:

name:
ietf-module-tags

namespace:
urn:ietf:params:xml:ns:yang:ietf-module-tags

prefix:
tags

reference:
RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove this note.)

```
   name:
      ietf-module-tags-state

   namespace:
      urn:ietf:params:xml:ns:yang:ietf-module-tags-state

   prefix:
      tags

   reference:
      RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove
      this note.)
```

8. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

This document adds the ability to associate tag meta-data with YANG modules. This document does not define any actions based on these associations, and none are yet defined, and therefore it does not by itself introduce any new security considerations directly.

Users of the tag-meta data may define various actions to be taken based on the tag meta-data. These actions and their definitions are outside the scope of this document. Users will need to consider the security implications of any actions they choose to define, including the potential for a tag to get 'masked' by another user.

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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

9.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <https://www.rfc-editor.org/info/rfc5198>.

[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/info/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/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[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>.

Appendix A. Examples

The following is a fictional NETCONF example result from a query of the module tags list. For the sake of brevity only a few module results are imagined.

```
<ns0:data xmlns:ns0="urn:ietf:params:xml:ns:netconf:base:1.0">
  <t:module-tags xmlns:t="urn:ietf:params:xml:ns:yang:ietf-module-tags">
    <t:module>
      <t:name>ietf-bfd</t:name>
      <t:tag>ietf:network-element-class</t:tag>
      <t:tag>ietf:oam</t:tag>
      <t:tag>ietf:protocol</t:tag>
      <t:tag>ietf:sdo-defined-class</t:tag>
    </t:module>
    <t:module>
      <t:name>ietf-isis</t:name>
      <t:tag>ietf:network-element-class</t:tag>
      <t:tag>ietf:protocol</t:tag>
      <t:tag>ietf:sdo-defined-class</t:tag>
      <t:tag>ietf:routing</t:tag>
    </t:module>
    <t:module>
      <t:name>ietf-ssh-server</t:name>
      <t:tag>ietf:network-element-class</t:tag>
      <t:tag>ietf:protocol</t:tag>
      <t:tag>ietf:sdo-defined-class</t:tag>
      <t:tag>ietf:system-management</t:tag>
    </t:module>
  </t:module-tags>
</ns0:data>
```

Figure 3: Example NETCONF Query Output

Appendix B. Acknowledgements

Special thanks to Robert Wilton for his help improving the introduction and providing the example use cases, as well as generating the non-NMDA module.

Appendix C. Non-NMDA State Module.

As per [RFC8407] the following is a non-NMDA module to support viewing the operational state for non-NMDA compliant servers.

```
<CODE BEGINS> file "ietf-module-tags-state@2020-02-29.yang"
module ietf-module-tags-state {
  yang-version 1.1;
```

```
  namespace "urn:ietf:params:xml:ns:yang:ietf-module-tags-state";
  prefix tags-s;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-module-tags {
    prefix tags;
  }

  organization
    "IETF NetMod Working Group (NetMod)";
  contact
    "WG Web:  <https://tools.ietf.org/wg/netmod/>
     WG List: <mailto:netmod@ietf.org>

     Author: Christian Hopps
             <mailto:chopps@chopps.org>

     Author: Lou Berger
             <mailto:lberger@labn.net>

     Author: Dean Bogdanovic
             <ivandean@gmail.com>";

  // RFC Ed.: replace XXXX with actual RFC number and
  // remove this note.

  description
    "This module describes a mechanism associating tags with YANG
     modules. Tags may be IANA assigned or privately defined.

     This is a temporary non-NMDA module that is for use by
     implementations that don't yet support NMDA.

     Copyright (c) 2019 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 Simplified 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
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.
```

```
      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.";

  // RFC Ed.: update the date below with the date of RFC publication
  // and RFC number and remove this note.

  revision 2020-02-29 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: YANG Module Tags";
  }

  container module-tags-state {
    config false;
    status deprecated;
    description
      "Contains the list of modules and their associated tags";
    list module {
      key "name";
      status deprecated;
      description
        "A list of modules and their associated tags";
      leaf name {
        type yang:yang-identifier;
        mandatory true;
        status deprecated;
        description
          "The YANG module name.";
      }
      leaf-list tag {
        type tags:tag;
        status deprecated;
        description
          "Tags associated with the module. See the IANA 'YANG Module
           Tag Prefixes' registry for reserved prefixes and the IANA
           'IETF YANG Module Tags' registry for IETF tags.

           The contents of this list is constructed using the
           following steps:

           1) System tags (i.e., tags of added by the system) are added.
           2) User configured tags (i.e., tags added by configuration)
           are added.
           3) Any tag that is equal to a masked-tag present in the
```

```
            corresponding ietf-module-tags:module-tags:module-tag leaf
            list for this module is removed.";
        }
      }
    }
  }
<CODE ENDS>
```

Figure 4: Non-NMDA Module Tags State Module

Authors' Addresses

Christian Hopps
LabN Consulting, L.L.C.

Email: chopps@chopps.org

Lou Berger
LabN Consulting, LLC.

Email: lberger@labn.net

Dean Bogdanovic
Volta Networks

Email: ivandean@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2019

M. Bjorklund
Tail-f Systems
L. Lhotka
CZ.NIC

October 17, 2018

YANG Schema Mount
draft-ietf-netmod-schema-mount-12

Abstract

This document defines a mechanism to add the schema trees defined by a set of YANG modules onto a mount point defined in the schema tree in some YANG module.

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 http://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 April 20, 2019.

Copyright Notice

Table of Contents

1. Introduction

Modularity and extensibility were among the leading design principles of the YANG data modeling language. As a result, the same YANG module can be combined with various sets of other modules and thus form a data model that is tailored to meet the requirements of a specific use case. Server implementors are only required to specify all YANG modules comprising the data model (together with their revisions and other optional choices) in the YANG library data ([RFC7895], [I-D.ietf-netconf-rfc7895bis] and Section 5.6.4 of [RFC7950]) implemented by the server. Such YANG modules appear in the data model "side by side", i.e., top-level data nodes of each module - if there are any - are also top-level nodes of the overall data model.

YANG has two mechanisms for contributing a schema hierarchy defined elsewhere to the contents of an internal node of the schema tree; these mechanisms are realized through the following YANG statements:

- The "uses" statement explicitly incorporates the contents of a grouping defined in the same or another module. See Section 4.2.6 of [RFC7950] for more details.

- The "augment" statement explicitly adds contents to a target node defined in the same or another module. See Section 4.2.8 of [RFC7950] for more details.

With both mechanisms, the YANG module with the "uses" or "augment" statement explicitly defines the exact location in the schema tree where the new nodes are placed.

In some cases these mechanisms are not sufficient; it is sometimes necessary that an existing module (or a set of modules) is added to the data model starting at locations other than the root. For example, YANG modules such as "ietf-interfaces" [RFC8343] are defined so as to be used in a data model of a physical device. Now suppose we want to model a device that supports multiple logical devices [I-D.ietf-rtgwg-lne-model], each of which has its own instantiation of "ietf-interfaces", and possibly other modules, but, at the same time, we want to be able to manage all these logical devices from the master device. Hence, we would like to have a schema tree like this:

```
  +--rw interfaces
  |  +--rw interface* [name]
  |     ...
  +--rw logical-network-element* [name]
     +--rw name
     |   ...
     +--rw interfaces
       +--rw interface* [name]
          ...
```

With the "uses" approach, the complete schema tree of "ietf-interfaces" would have to be wrapped in a grouping, and then this grouping would have to be used at the top level (for the master device) and then also in the "logical-network-element" list (for the logical devices). This approach has several disadvantages:

- It is not scalable because every time there is a new YANG module that needs to be added to the logical device model, we have to update the model for logical devices with another "uses" statement pulling in contents of the new module.

- Absolute references to nodes defined inside a grouping may break if the grouping is used in different locations.

- Nodes defined inside a grouping belong to the namespace of the module where it is used, which makes references to such nodes from other modules difficult or even impossible.

- It would be difficult for vendors to add proprietary modules when the "uses" statements are defined in a standard module.

With the "augment" approach, "ietf-interfaces" would have to augment the "logical-network-element" list with all its nodes, and at the same time define all its nodes at the top level. The same hierarchy of nodes would thus have to be defined twice, which is clearly not scalable either.

This document introduces a new mechanism, denoted as schema mount, that allows for mounting one data model consisting of any number of YANG modules at a specified location of another (parent) schema. Unlike the "uses" and "augment" approaches discussed above, the mounted modules needn't be specially prepared for mounting and, consequently, existing modules such as "ietf-interfaces" can be mounted without any modifications.

The basic idea of schema mount is to label a data node in the parent schema as the mount point, and then define a complete data model to be attached to the mount point so that the labeled data node effectively becomes the root node of the mounted data model.

In principle, the mounted schema can be specified at three different phases of the data model life cycle:

1. Design-time: the mounted schema is defined along with the mount point in the parent YANG module. In this case, the mounted schema has to be the same for every implementation of the parent module.

2. Implementation-time: the mounted schema is defined by a server implementor and is as stable as the YANG library information of the server.

3. Run-time: the mounted schema is defined by instance data that is part of the mounted data model. If there are multiple instances of the same mount point (e.g., in multiple entries of a list), the mounted data model may be different for each instance.

The schema mount mechanism defined in this document provides support only for the latter two cases. Design-time mounts are outside the

scope of this document, and could be possibly dealt with in a future revision of the YANG data modeling language.

Schema mount applies to the data model, and specifically does not assume anything about the source of instance data for the mounted schemas. It may be implemented using the same instrumentation as the rest of the system, or it may be implemented by querying some other system. Future specifications may define mechanisms to control or monitor the implementation of specific mount points.

How and when specific mount points are instantiated by the server is out of scope for this document. Such mechanisms may be defined in future specifications.

This document allows mounting of complete data models only. Other specifications may extend this model by defining additional mechanisms such as mounting sub-hierarchies of a module.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Terminology and Notation

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] and are not redefined here:

- action
- container
- data node
- list
- RPC operation
- schema node
- schema tree

The following terms are defined in [RFC8342] and are not redefined here:

- client
- notification
- operational state
- server

The following term is defined in [RFC8343] and is not redefined here:

- system-controlled interface

The following term is defined in [I-D.ietf-netconf-rfc7895bis] is not redefined here:

- YANG library content identifier

The following additional terms are used within this document:

- mount point: A container or a list node whose definition contains the "mount-point" extension statement. The argument of the "mount-point" statement defines a label for the mount point.
- schema: A collection of schema trees with a common root.
- top-level schema: A schema rooted at the root node.
- mounted schema: A schema rooted at a mount point.
- parent schema (of a mounted schema): A schema containing the mount point.
- schema mount: The mechanism to combine data models defined in this document.

2.1. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340]

2.2. Namespace Prefixes

In this document, names of data nodes, YANG extensions, actions and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
yangmnt	ietf-yang-schema-mount	Section 9
inet	ietf-inet-types	[RFC6991]
yang	ietf-yang-types	[RFC6991]
yanglib	ietf-yang-library	[RFC7895], [I-D.ietf-netconf-rfc7895bis]

Table 1: Namespace Prefixes

3. Schema Mount

The schema mount mechanism defined in this document provides a new extensibility mechanism for use with YANG 1.1. In contrast to the existing mechanisms described in Section 1, schema mount defines the relationship between the source and target YANG modules outside these modules. The procedure consists of two separate steps that are described in the following subsections.

3.1. Mount Point Definition

A "container" or "list" node becomes a mount point if the "mount-point" extension (defined in the "ietf-yang-schema-mount" module) is used in its definition. This extension can appear only as a substatement of "container" and "list" statements.

The argument of the "mount-point" extension is a YANG identifier that defines a label for the mount point. A module MAY contain multiple "mount-point" statements having the same argument.

It is therefore up to the designer of the parent schema to decide about the placement of mount points. A mount point can also be made conditional by placing "if-feature" and/or "when" as substatements of the "container" or "list" statement that represents the mount point.

The "mount-point" statement MUST NOT be used in a YANG version 1 module [RFC6020]. The reason for this is that otherwise it is not possible to invoke mounted RPC operations, and receive mounted notifications. See Section 5 for details. Note, however, that modules written in any YANG version, including version 1, can be mounted under a mount point.

Note that the "mount-point" statement does not define a new data node.

3.2. Data Model

This document defines the YANG 1.1 module [RFC7950] "ietf-yang-schema-mount", which has the following structure:

```
module: ietf-yang-schema-mount
  +--ro schema-mounts
     +--ro namespace* [prefix]
     |  +--ro prefix    yang:yang-identifier
     |  +--ro uri?      inet:uri
     +--ro mount-point* [module label]
        +--ro module                yang:yang-identifier
        +--ro label                 yang:yang-identifier
        +--ro config?               boolean
        +--ro (schema-ref)
           +--:(inline)
           |  +--ro inline!
           +--:(shared-schema)
              +--ro shared-schema!
                 +--ro parent-reference*   yang:xpath1.0
```

3.3. Specification of the Mounted Schema

Mounted schemas for all mount points in the parent schema are determined from state data in the "/schema-mounts" container.

Generally, the modules that are mounted under a mount point have no relation to the modules in the parent schema; specifically, if a module is mounted it may or may not be present in the parent schema and, if present, its data will generally have no relationship to the data of the parent. Exceptions are possible and such needs to be defined in the model defining the exception. For example, [I-D.ietf-rtgwg-lne-model] defines a mechanism to bind interfaces to mounted logical network elements.

The "/schema-mounts" container has the "mount-point" list as one of its children. Every entry of this list refers through its key to a mount point and specifies the mounted schema.

If a mount point is defined in the parent schema but does not have an entry in the "mount-point" list, then the mounted schema is void, i.e., instances of that mount point MUST NOT contain any data except those that are defined in the parent schema.

If multiple mount points with the same name are defined in the same module - either directly or because the mount point is defined in a grouping and the grouping is used multiple times - then the

corresponding "mount-point" entry applies equally to all such mount points.

The "config" property of mounted schema nodes is overridden and all nodes in the mounted schema are read-only ("config false") if at least one of the following conditions is satisfied for a mount point:

- the mount point is itself defined as "config false"
- the "config" leaf in the corresponding entry of the "mount-point" list is set to "false".

An entry of the "mount-point" list can specify the mounted schema in two different ways, "inline" or "shared-schema".

The mounted schema is determined at run time: every instance of the mount point that exists in the operational state MUST contain a copy of YANG library data that defines the mounted schema exactly as for a top-level schema. A client is expected to retrieve this data from the instance tree. In the "inline" case, instances of the same mount point MAY use different mounted schemas, whereas in the "shared-schema" case, all instances MUST use the same mounted schema. This means that in the "shared-schema" case, all instances of the same mount point MUST have the same YANG library content identifier. In the "inline" case, if two instances have the same YANG library content identifier it is not guaranteed that the YANG library contents are equal for these instances.

Examples of "inline" and "shared-schema" can be found in Appendix A.2 and Appendix A.3, respectively.

3.4. Multiple Levels of Schema Mount

YANG modules in a mounted schema MAY again contain mount points under which other schemas can be mounted. Consequently, it is possible to construct data models with an arbitrary number of mounted schemas. A schema for a mount point contained in a mounted module can be specified by implementing "ietf-yang-library" and "ietf-yang-schema-mount" modules in the mounted schema, and specifying the schemas exactly as it is done in the top-level schema.

4. Referring to Data Nodes in the Parent Schema

A fundamental design principle of schema mount is that the mounted schema works exactly as a top-level schema, i.e., it is confined to the "mount jail". This means that all paths in the mounted schema (in leafrefs, instance-identifiers, XPath [XPATH] expressions, and target nodes of augments) are interpreted with the mount point as the

root node. YANG modules of the mounted schema as well as corresponding instance data thus cannot refer to schema nodes or instance data outside the mount jail.

However, this restriction is sometimes too severe. A typical example is network instances (NI) [I-D.ietf-rtgwg-ni-model], where each NI has its own routing engine but the list of interfaces is global and shared by all NIs. If we want to model this organization with the NI schema mounted using schema mount, the overall schema tree would look schematically as follows:

```
  +--rw interfaces
  |  +--rw interface* [name]
  |     ...
  +--rw network-instances
     +--rw network-instance* [name]
        +--rw name
        +--rw root
           +--rw routing
              ...
```

Here, the "root" node is the mount point for the NI schema. Routing configuration inside an NI often needs to refer to interfaces (at least those that are assigned to the NI), which is impossible unless such a reference can point to a node in the parent schema (interface name).

Therefore, schema mount also allows for such references. For every mount point in the "shared-schema" case, it is possible to specify a leaf-list named "parent-reference" that contains zero or more XPath 1.0 expressions. Each expression is evaluated with the node in the parent data tree where the mount point is defined as the context node. The result of this evaluation MUST be a nodeset (see the description of the "parent-reference" node for a complete definition of the evaluation context). For the purposes of evaluating XPath expressions within the mounted data tree, the union of all such nodesets is added to the accessible data tree.

It is worth emphasizing that the nodes specified in "parent-reference" leaf-list are available in the mounted schema only for XPath evaluations. In particular, they cannot be accessed there via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040].

5. RPC operations and Notifications

If a mounted YANG module defines an RPC operation, clients can invoke this operation as if it were defined as an action for the corresponding mount point, see Section 7.15 of [RFC7950]. An example of this is given in Appendix A.4.

Similarly, if the server emits a notification defined at the top level of any mounted module, it MUST be represented as if the notification was connected to the mount point, see Section 7.16 of [RFC7950].

Note, inline actions and notifications will not work when they are contained within a list node without a "key" statement (see section 7.15 and 7.16 of [RFC7950]). Therefore, to be useful, mount points that contain modules with RPCs, actions, and notifications SHOULD NOT have any ancestor node that is a list node without a "key" statement. This requirement applies to the definition of modules using the "mount-point" extension statement.

6. Network Management Datastore Architecture (NMDA) Considerations

The schema mount solution presented in this document is designed to work both with servers that implement the NMDA [RFC8342], and old servers that don’t implement the NMDA.

Note to RFC Editor: please update the date YYYY-MM-DD below with the revision of the ietf-yang-library in the published version of draft-ietf-netconf-rfc7895bis, and remove this note.

Specifically, a server that doesn’t support the NMDA, MAY implement revision 2016-06-21 of "ietf-yang-library" [RFC7895] under a mount point. A server that supports the NMDA, MUST implement at least revision YYYY-MM-DD of "ietf-yang-library" [I-D.ietf-netconf-rfc7895bis] under the mount points.

7. Interaction with the Network Configuration Access Control Model (NACM)

If NACM [RFC8341] is implemented on a server, it is used to control access to nodes defined by the mounted schema in the same way as for nodes defined by the top-level schema.

For example, suppose the module "ietf-interfaces" is mounted in the "root" container in the "logical-network-element" list defined in [I-D.ietf-rtgwg-lne-model]. Then the following NACM path can be used to control access to the "interfaces" container (where the character

'\' is used where a line break has been inserted for formatting reasons):

```
  <path xmlns:lne=
          "urn:ietf:params:xml:ns:yang:ietf-logical-network-element"
        xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    /lne:logical-network-elements\
      /lne:logical-network-element/lne:root/if:interfaces
  </path>
```

8. Implementation Notes

Network management of devices that use a data model with schema mount can be implemented in different ways. However, the following implementations options are envisioned as typical:

- shared management: instance data of both parent and mounted schemas are accessible within the same management session.

- split management: one (master) management session has access to instance data of both parent and mounted schemas but, in addition, an extra session exists for every instance of the mount point, having access only to the mounted data tree.

9. Schema Mount YANG Module

This module references [RFC6991].

<CODE BEGINS> file "ietf-yang-schema-mount@2018-10-16"

```
module ietf-yang-schema-mount {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount";
  prefix yangmnt;

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

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

  organization
```

```
      "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

    contact
      "WG Web:   <https://tools.ietf.org/wg/netmod/>
       WG List:  <mailto:netmod@ietf.org>

       Editor:   Martin Bjorklund
                 <mailto:mbj@tail-f.com>

       Editor:   Ladislav Lhotka
                 <mailto:lhotka@nic.cz>";

    // RFC Ed.: replace XXXX with actual RFC number and
    // remove this note.
    description
      "This module defines a YANG extension statement that can be used
       to incorporate data models defined in other YANG modules in a
       module. It also defines operational state data that specify the
       overall structure of the data model.

       Copyright (c) 2018 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 Simplified 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).

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
       NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
       'MAY', and 'OPTIONAL' in the module text are to be interpreted
       as described in BCP 14 [RFC 2119] [RFC8174] when, and only when,
       they appear in all capitals, as shown here.

       This version of this YANG module is part of RFC XXXX
       (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for
       full legal notices.";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    revision 2018-10-16 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: YANG Schema Mount";
    }
```

```
     /*
      * Extensions
      */

     extension mount-point {
       argument label;
       description
         "The argument 'label' is a YANG identifier, i.e., it is of the
          type 'yang:yang-identifier'.

          The 'mount-point' statement MUST NOT be used in a YANG
          version 1 module, neither explicitly nor via a 'uses'
          statement.

          The 'mount-point' statement MAY be present as a substatement
          of 'container' and 'list', and MUST NOT be present elsewhere.
          There MUST NOT be more than one 'mount-point' statement in a
          given 'container' or 'list' statement.

          If a mount point is defined within a grouping, its label is
          bound to the module where the grouping is used.

          A mount point defines a place in the node hierarchy where
          other data models may be attached. A server that implements a
          module with a mount point populates the
          /schema-mounts/mount-point list with detailed information on
          which data models are mounted at each mount point.

          Note that the 'mount-point' statement does not define a new
          data node.";
     }

     /*
      * State data nodes
      */

     container schema-mounts {
       config false;
       description
         "Contains information about the structure of the overall
          mounted data model implemented in the server.";
       list namespace {
         key "prefix";
         description
           "This list provides a mapping of namespace prefixes that are
            used in XPath expressions of 'parent-reference' leafs to the
            corresponding namespace URI references.";
         leaf prefix {
```

```
          type yang:yang-identifier;
          description
            "Namespace prefix.";
        }
        leaf uri {
          type inet:uri;
          description
            "Namespace URI reference.";
        }
      }
      list mount-point {
        key "module label";
        description
          "Each entry of this list specifies a schema for a particular
           mount point.

           Each mount point MUST be defined using the 'mount-point'
           extension in one of the modules listed in the server's
           YANG library instance with conformance type 'implement'.";
        leaf module {
          type yang:yang-identifier;
          description
            "Name of a module containing the mount point.";
        }
        leaf label {
          type yang:yang-identifier;
          description
            "Label of the mount point defined using the 'mount-point'
             extension.";
        }
        leaf config {
          type boolean;
          default "true";
          description
            "If this leaf is set to 'false', then all data nodes in the
             mounted schema are read-only (config false), regardless of
             their 'config' property.";
        }
        choice schema-ref {
          mandatory true;
          description
            "Alternatives for specifying the schema.";
          container inline {
            presence
              "A complete self-contained schema is mounted at the
               mount point.";
            description
              "This node indicates that the server has mounted at least
```

```
            the module 'ietf-yang-library' at the mount point, and
            its instantiation provides the information about the
            mounted schema.

            Different instances of the mount point may have
            different schemas mounted.";
        }
        container shared-schema {
          presence
            "The mounted schema together with the 'parent-reference'
             make up the schema for this mount point.";
          description
            "This node indicates that the server has mounted at least
             the module 'ietf-yang-library' at the mount point, and
             its instantiation provides the information about the
             mounted schema.  When XPath expressions in the mounted
             schema are evaluated, the 'parent-reference' leaf-list
             is taken into account.

             Different instances of the mount point MUST have the
             same schema mounted.";
          leaf-list parent-reference {
            type yang:xpath1.0;
            description
              "Entries of this leaf-list are XPath 1.0 expressions
               that are evaluated in the following context:

                - The context node is the node in the parent data tree
                  where the mount-point is defined.

                - The accessible tree is the parent data tree
                  *without* any nodes defined in modules that are
                  mounted inside the parent schema.

                - The context position and context size are both equal
                  to 1.

                - The set of variable bindings is empty.

                - The function library is the core function library
                  defined in [XPath] and the functions defined in
                  Section 10 of [RFC7950].

                - The set of namespace declarations is defined by the
                  'namespace' list under 'schema-mounts'.

               Each XPath expression MUST evaluate to a nodeset
               (possibly empty). For the purposes of evaluating XPath
```

```
                    expressions whose context nodes are defined in the
                    mounted schema, the union of all these nodesets
                    together with ancestor nodes are added to the
                    accessible data tree.

                    Note that in the case 'ietf-yang-schema-mount' is
                    itself mounted, a 'parent-reference' in the mounted
                    module may refer to nodes that were brought into the
                    accessible tree through a 'parent-reference' in the
                    parent schema.";
              }
            }
          }
        }
      }
    }
```

<CODE ENDS>

10. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
  name:         ietf-yang-schema-mount
  namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount
  prefix:       yangmnt
  reference:    RFC XXXX
```

11. Security Considerations

YANG module "ietf-yang-schema-mount" specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The network configuration access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /schema-mounts: The schema defined by this state data provides detailed information about a server implementation may help an attacker identify the server capabilities and server implementations with known bugs. Server vulnerabilities may be specific to particular modules included in the schema, module revisions, module features, or even module deviations. For example, if a particular operation on a particular data node is known to cause a server to crash or significantly degrade device performance, then the schema information will help an attacker identify server implementations with such a defect, in order to launch a denial-of-service attack on the device.

It is important to take the security considerations for all nodes in the mounted schemas into account, and control access to these nodes by using the mechanism described in Section 7.

Care must be taken when the "parent-reference" XPath expressions are constructed, since the result of the evaluation of these expressions is added to the accessible tree for any XPath expression found in the mounted schema.

12. Contributors

The idea of having some way to combine schemas from different YANG modules into one has been proposed independently by several groups of people: Alexander Clemm, Jan Medved, and Eric Voit ([I-D.clemm-netmod-mount]); and Lou Berger and Christian Hopps:

- Lou Berger, LabN Consulting, L.L.C., <lberger@labn.net>
- Alexander Clemm, Huawei, <alexander.clemm@huawei.com>
- Christian Hopps, Deutsche Telekom, <chopps@chopps.org>
- Jan Medved, Cisco, <jmedved@cisco.com>
- Eric Voit, Cisco, <evoit@cisco.com>

13. References

13.1. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-06 (work in progress), April 2018.

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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/info/rfc6020>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[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>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[XPATH] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

13.2. Informative References

[I-D.clemm-netmod-mount] Clemm, A., Voit, E., and J. Medved, "Mounting YANG-Defined Information from Remote Datastores", draft-clemm-netmod-mount-06 (work in progress), March 2017.

[I-D.ietf-isis-yang-isis-cfg] Litkowski, S., Yeung, D., Lindem, A., Zhang, Z., and L. Lhotka, "YANG Data Model for IS-IS protocol", draft-ietf-isis-yang-isis-cfg-24 (work in progress), August 2018.

[I-D.ietf-rtgwg-device-model] Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Logical Organization", draft-ietf-rtgwg-device-model-02 (work in progress), March 2017.

[I-D.ietf-rtgwg-lne-model] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", draft-ietf-rtgwg-lne-model-10 (work in progress), March 2018.

[I-D.ietf-rtgwg-ni-model] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Network Instances", draft-ietf-rtgwg-ni-model-12 (work in progress), March 2018.

[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>.

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

[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>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

Appendix A. Example: Device Model with LNEs and NIs

This non-normative example demonstrates an implementation of the device model as specified in Section 2 of [I-D.ietf-rtgwg-device-model], using both logical network elements (LNE) and network instances (NI).

In these examples, the character '\' is used where a line break has been inserted for formatting reasons.

A.1. Physical Device

The data model for the physical device may be described by this YANG library content, assuming the server supports the NMDA:

```
{
   "ietf-yang-library:yang-library": {
     "content-id": "14e2ab5dc325f6d86f743e8d3ade233f1a61a899",
     "module-set": [
       {
         "name": "physical-device-modules",
         "module": [
           {
             "name": "ietf-datastores",
             "revision": "2018-02-14",
             "namespace":
               "urn:ietf:params:xml:ns:yang:ietf-datastores"
           },
           {
             "name": "iana-if-type",
             "revision": "2015-06-12",
             "namespace": "urn:ietf:params:xml:ns:yang:iana-if-type"
           },
           {
             "name": "ietf-interfaces",
             "revision": "2018-02-20",
```

```
                "feature": ["arbitrary-names", "pre-provisioning" ],
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-interfaces"
              },
              {
                "name": "ietf-ip",
                "revision": "2018-02-22",
                "namespace": "urn:ietf:params:xml:ns:yang:ietf-ip"
              },
              {
                "name": "ietf-logical-network-element",
                "revision": "2016-10-21",
                "feature": [ "bind-lne-name" ],
                "namespace":
                  "urn:ietf:params:xml:ns:yang:\
                  ietf-logical-network-element"
              },
              {
                "name": "ietf-yang-library",
                "revision": "2018-02-21",
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-yang-library"
              },
              {
                "name": "ietf-yang-schema-mount",
                "revision": "2018-03-20",
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount"
              }
            ],
            "import-only-module": [
              {
                "name": "ietf-inet-types",
                "revision": "2013-07-15",
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-inet-types"
              },
              {
                "name": "ietf-yang-types",
                "revision": "2013-07-15",
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-yang-types"
              }
            ]
          }
        ],
        "schema": [
          {
```

```
            "name": "physical-device-schema",
            "module-set": [ "physical-device-modules" ]
          }
        ],
        "datastore": [
          {
            "name": "ietf-datastores:running",
            "schema": "physical-device-schema"
          },
          {
            "name": "ietf-datastores:operational",
            "schema": "physical-device-schema"
          }
        ]
      }
    }
```

A.2. Logical Network Elements

Each LNE can have a specific data model that is determined at run time, so it is appropriate to mount it using the "inline" method, hence the following "schema-mounts" data:

```
{
  "ietf-yang-schema-mount:schema-mounts": {
    "mount-point": [
      {
        "module": "ietf-logical-network-element",
        "label": "root",
        "inline": {}
      }
    ]
  }
}
```

An administrator of the host device has to configure an entry for each LNE instance, for example,

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": true,
        "ietf-logical-network-element:bind-lne-name": "eth0"
      }
    ]
  },
  "ietf-logical-network-element:logical-network-elements": {
    "logical-network-element": [
      {
        "name": "lne-1",
        "managed": true,
        "description": "LNE with NIs",
        "root": {
          ...
        }
      }
      ...
    ]
  }
}
```

and then also place necessary state data as the contents of the "root" instance, which should include at least

- YANG library data specifying the LNE's data model, for example, assuming the server does not implement the NMDA:

```
{
  "ietf-yang-library:modules-state": {
    "module-set-id": "9358e11874068c8be06562089e94a89e0a392019",
    "module": [
      {
        "name": "iana-if-type",
        "revision": "2014-05-08",
        "namespace": "urn:ietf:params:xml:ns:yang:iana-if-type",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-inet-types",
        "revision": "2013-07-15",
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types",
        "conformance-type": "import"
      },
```

```
        {
          "name": "ietf-interfaces",
          "revision": "2014-05-08",
          "feature": [
            "arbitrary-names",
            "pre-provisioning"
          ],
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-interfaces",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ip",
          "revision": "2014-06-16",
          "feature": [
            "ipv6-privacy-autoconf"
          ],
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-ip",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-network-instance",
          "revision": "2016-10-27",
          "feature": [
            "bind-network-instance-name"
          ],
          "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-network-instance",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-schema-mount",
          "revision": "2017-05-16",
          "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
```

```
      ]
    }
  }
```

o state data for interfaces assigned to the LNE instance (that effectively become system-controlled interfaces for the LNE), for example:

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "oper-status": "up",
        "statistics": {
          "discontinuity-time": "2016-12-16T17:11:27+02:00"
        },
        "ietf-ip:ipv6": {
          "address": [
            {
              "ip": "fe80::42a8:f0ff:fea8:24fe",
              "origin": "link-layer",
              "prefix-length": 64
            }
          ]
        }
      }
    ]
  }
}
```

A.3. Network Instances

Assuming that network instances share the same data model, it can be mounted using the "shared-schema" method as follows:

```
{
  "ietf-yang-schema-mount:schema-mounts": {
    "namespace": [
      {
          "prefix": "if",
          "uri": "urn:ietf:params:xml:ns:yang:ietf-interfaces"
      },
      {
          "prefix": "ni",
          "uri": "urn:ietf:params:xml:ns:yang:ietf-network-instance"
      }
    ],
    "mount-point": [
      {
        "module": "ietf-network-instance",
        "label": "root",
          "shared-schema": {
            "parent-reference": [
              "/if:interfaces/if:interface[\
              ni:bind-network-instance-name = current()/../ni:name]"
            ]
          }
      }
    ]
  }
}
```

Note also that the "ietf-interfaces" module appears in the "parent-reference" leaf-list for the mounted NI schema. This means that references to LNE interfaces, such as "outgoing-interface" in static routes, are valid despite the fact that "ietf-interfaces" isn’t part of the NI schema.

A.4. Invoking an RPC Operation

Assume that the mounted NI data model also implements the "ietf-isis" module [I-D.ietf-isis-yang-isis-cfg]. An RPC operation defined in this module, such as "clear-adjacency", can be invoked by a client session of a LNE’s RESTCONF server as an action tied to a the mount point of a particular network instance using a request URI like this (all on one line):

```
POST /restconf/data/ietf-network-instance:network-instances/
    network-instance=rtrA/root/ietf-isis:clear-adjacency HTTP/1.1
```

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
K. Watsen
Juniper Networks
March 5, 2018

YANG Data Extensions
draft-ietf-netmod-yang-data-ext-01

Abstract

This document describes YANG mechanisms for defining abstract data structures with YANG. It is intended to replace and extend the "yang-data" extension statement defined in RFC 8040.

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 http://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 September 6, 2018.

Copyright Notice

Table of Contents

1. Introduction

There is a need for standard mechanisms to allow the definition of abstract data that is not intended to be implemented as configuration or operational state. The "yang-data" extension statement from RFC 8040 [RFC8040] is defined for this purpose, however it is limited in its functionality.

The intended use of the "yang-data" extension is to model all or part of a protocol message, such as the "errors" definition in ietf-restconf.yang [RFC8040], or the contents of a file. However, protocols are often layered such that the header or payload portions of the message can be extended by external documents. The YANG statements that model a protocol need to support this extensibility that is already found in that protocol.

This document defines a new YANG extension statement called "augment-yang-data", which allows abstract data structures to be augmented from external modules, similar to the existing YANG "augment" statement. Note that "augment" cannot be used to augment a yang data structure since a YANG compiler or other tool is not required to understand the "yang-data" extension.

The "yang-data" extension from [RFC8040] has been copied here and updated to be more flexible. There is no longer a requirement for the "yang-data" statement to result in exactly one container object. There is no longer an assumption that a yang data structure can only be used as a top-level abstraction, instead of nested within some other data structure.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The following terms are used within this document:

- yang data structure: A data structure defined with the "yang-data" statement.

1.1.1. NMDA

The following terms are defined in the Network Management Datastore Architecture (NMDA) [I-D.ietf-netmod-revised-datastores]. and are not redefined here:

- configuration
- operational state

1.1.2. YANG

The following terms are defined in [RFC7950]:

- absolute-schema-nodeid
- container
- data definition statement
- data node
- leaf
- leaf-list
- list

2. Definitions

2.1. Restrictions on Conceptual YANG Data

This document places restrictions on the "yang-data" external statements that can be used with the "yang-data" and "augment-yang-data" extensions. The conceptual data definitions are considered to be in the same identifier namespace as defined in section 6.2.1 of [RFC7950]. In particular, bullet 7:

> All leafs, leaf-lists, lists, containers, choices, rpcs, actions, notifications, anydatas, and anyxmls defined (directly or through a "uses" statement) within a parent node or at the top level of the module or its submodules share the same identifier namespace.

This means that conceptual data defined with the "yang-data" or "augment-yang-data" statements cannot have the same local-name as sibling nodes from regular YANG data definition statements or other "yang-data" or "augment-yang-data" statements.

This does not mean a yang data structure has to be used as a top-level protocol message or other top-level data structure. A yang data structure does not have to result in a single container.

2.2. YANG Data Extensions Module

The "ietf-yang-data-ext" module defines the "augment-yang-data" extension to augment conceptual data already defined with the "yang-data" extension. The RESTCONF "yang-data" extension has been moved to this document and updated.

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-yang-data-ext@2018-03-05.yang"

```
module ietf-yang-data-ext {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-data-ext";
  prefix "yd";

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>
```

```
       Author:   Andy Bierman
                 <mailto:andy@yumaworks.com>

       Author:   Martin Bjorklund
                 <mailto:mbj@tail-f.com>

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.net>";


    description
      "This module contains conceptual YANG specifications
       for defining abstract 'yang-data' data structures.

       Copyright (c) 2017 - 2018 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 Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).";

    revision 2018-03-05 {
      description
        "Initial revision.";
      reference
         // RFC Ed.: replace XXXX with RFC number and remove this note
        "RFC XXXX: YANG Data Extensions.";
    }


    extension yang-data {
      argument name {
        yin-element true;
      }
      description
        "This extension is used to specify a YANG data template which
         represents conceptual data defined in YANG. It is
         intended to describe hierarchical data independent of
         protocol context or specific message encoding format.
         Data definition statements within a yang-data extension
         specify the generic syntax for the specific YANG data
         template, whose name is the argument of the yang-data
         extension statement.

         Note that this extension does not define a media-type.
```

A specification using this extension MUST specify the message encoding rules, including the content media type.

The mandatory 'name' parameter value identifies the YANG data template that is being defined. It contains the template name. This parameter is only used for readability purposes. There are no mechanisms to reuse yang-data by its template name value.

This extension is ignored unless it appears as a top-level statement. It MUST contain data definition statements that result in a set of data definition statements.

If the yang data template is intended to be used as a top-level structure, then the yang data template needs to result in a single container, so an instance of the YANG data template can thus be translated into an XML instance document, whose top-level element corresponds to the top-level container.

The module name and namespace value for the YANG module using the extension statement is assigned to each of the data definition statements resulting from the yang data template. The name of each data definition statement resulting from a yang data template is assigned to a top-level identifier name in the data node identifier namespace, according to RFC 7950, section 6.2.1.

The sub-statements of this extension MUST follow the 'data-def-stmt' rule in the YANG ABNF.

The XPath document root is the extension statement itself, such that the child nodes of the document root are represented by the data-def-stmt sub-statements within this extension. This conceptual document is the context for the following YANG statements:

- must-stmt
- when-stmt
- path-stmt
- min-elements-stmt
- max-elements-stmt
- mandatory-stmt
- unique-stmt
- ordered-by
- instance-identifier data type

The following data-def-stmt sub-statements are constrained when used within a yang-data-resource extension statement.

```
          - The list-stmt is not required to have a key-stmt defined.
          - The if-feature-stmt is ignored if present.
          - The config-stmt is ignored if present.
          - The available identity values for any 'identityref'
            leaf or leaf-list nodes is limited to the module
            containing this extension statement, and the modules
            imported into that module.
      ";
  }


  extension augment-yang-data {
    argument path {
      yin-element true;
    }
    description
      "This extension is used to specify an augmentation to
       conceptual data defined with the 'yang-data' statement.
       It is intended to describe hierarchical data independent
       of protocol context or specific message encoding format.

       This statement has almost the same structure as the
       'augment-stmt'. Data definition statements within this
       statement specify the semantics and generic syntax for the
       additional data to be added to the specific YANG data template,
       identified by the 'path' argument.

       The mandatory 'path' parameter value identifies the YANG
       conceptual data node that is being augmented, represented
       as an absolute-schema-nodeid string.

       This extension is ignored unless it appears as a top-level
       statement. The sub-statements of this extension MUST follow
       the 'data-def-stmt' rule in the YANG ABNF.

       The module name and namespace value for the YANG module using
       the extension statement is assigned to instance document data
       conforming to the data definition statements within
       this extension.

       The XPath document root is the augmented extension statement
       itself, such that the child nodes of the document root are
       represented by the data-def-stmt sub-statements within
       the augmented yang-data statement.

       The context node of the augment-yang-data statement is derived
       in the same way as the 'augment' statement, as defined in
       section 6.4.1 of [RFC7950]. This conceptual node is
```

considered the context node for the following YANG statements:

- must-stmt
- when-stmt
- path-stmt
- min-elements-stmt
- max-elements-stmt
- mandatory-stmt
- unique-stmt
- ordered-by
- instance-identifier data type

The following data-def-stmt sub-statements are constrained when used within a augment-yang-data extension statement.

- The list-stmt is not required to have a key-stmt defined.
- The if-feature-stmt is ignored if present.
- The config-stmt is ignored if present.
- The available identity values for any 'identityref' leaf or leaf-list nodes is limited to the module containing this extension statement, and the modules imported into that module.

Example:

```
   foo.yang {
      import yang-data-ext { prefix yd; }

      yd:yang-data foo-data {
        container foo-con { }
      }
   }

   bar.yang {
      import yang-data-ext { prefix yd; }
      import foo { prefix foo; }

      yd:augment-yang-data /foo:foo-con {
        leaf add-leaf1 { type int32; }
        leaf add-leaf2 { type string; }
      }
   }
";
  }

 }
```

<CODE ENDS>

3. IANA Considerations

3.1. YANG Module Registry

This document registers one URI as a namespace in the "IETF XML Registry" [RFC3688]:

```
URI: urn:ietf:params:xml:ns:yang:ietf-yang-data-ext
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]:

```
name:         ietf-yang-data-ext
namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-data-ext
prefix:       yd
// RFC Ed.: replace XXXX with RFC number and remove this note
reference:    RFC XXXX
```

4. Security Considerations

This document defines YANG extensions that are used to define conceptual YANG data. It does not introduce any new vulnerabilities beyond those specified in YANG 1.1 [RFC7950].

5. Normative References

[I-D.ietf-netmod-revised-datastores] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-10 (work in progress), January 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

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

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

Appendix A. Examples

A.1. yang-data Example

This example shows a simple address book that could be stored as an artifact.

```
yd:yang-data example-address-book {
  container address-book {
    list address {
      key "last first";
      leaf last {
        type string;
        description "Last name";
      }
      leaf first {
        type string;
        description "First name";
      }
      leaf street {
        type string;
        description "Street name";
      }
      leaf city {
        type string;
        description "City name";
      }
      leaf state {
        type string;
        description "State name";
      }
    }
  }
}
```

A.2. augment-yang-data Example

This example adds "county" and "zipcode" leafs to the address book:

```
      yd:augment-yang-data /address-book/address {
        leaf county {
          type string;
          description "County name";
        }
        leaf zipcode {
          type string;
          description "Postal zipcode";
        }
      }
```

Appendix B. Change Log

B.1. v00 to v01

- moved open issues to github
- added examples section
- filled in IANA considerations

Appendix C. Open Issues

The YANG Data Extensions issues are tracked on github.com:

https://github.com/netmod-wg/yang-data-ext/issues

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net

Network Working Group
Internet-Draft
Updates: 8340 (if approved)
Intended status: Standards Track
Expires: June 12, 2020

A. Bierman
YumaWorks
M. Bjorklund
Cisco
K. Watsen
Watsen Networks

December 10, 2019

YANG Data Structure Extensions
draft-ietf-netmod-yang-data-ext-05

Abstract

This document describes YANG mechanisms for defining abstract data structures with YANG.

Status of This Memo

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

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

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 June 12, 2020.

Copyright Notice

Table of Contents

1. Introduction

There is a need for standard mechanisms to allow the definition of abstract data that is not intended to be implemented as configuration or operational state. The "yang-data" extension statement from RFC 8040 [RFC8040] was defined for this purpose but it is limited in its functionality.

The intended use of the "yang-data" extension was to model all or part of a protocol message, such as the "errors" definition in the YANG module "ietf-restconf" [RFC8040], or the contents of a file. However, protocols are often layered such that the header or payload portions of the message can be extended by external documents. The YANG statements that model a protocol need to support this extensibility that is already found in that protocol.

This document defines a new YANG extension statement called "structure", which is similar to but more flexible than the "yang-data" extension from [RFC8040]. There is no assumption that a YANG data structure can only be used as a top-level abstraction, and it may also be nested within some other data structure.

This document also defines a new YANG extension statement called "augment-structure", which allows abstract data structures to be

augmented from external modules, similarly to the existing YANG "augment" statement. Note that "augment" cannot be used to augment a YANG data structure since a YANG compiler or other tool is not required to understand the "structure" extension.

1.1. 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 used within this document:

- YANG data structure: A data structure defined with the "structure" statement.

1.1.1. NMDA

The following terms are defined in the Network Management Datastore Architecture (NMDA) [RFC8342], and are not redefined here:

- configuration
- operational state

1.1.2. YANG

The following terms are defined in [RFC7950]:

- absolute-schema-nodeid
- container
- data definition statement
- data node
- leaf
- leaf-list
- list

2. Definitions

A YANG data structure is defined with the "structure" extension statement, defined in the YANG module "ietf-yang-structure-ext". The argument to the "structure" extension statement is the name of the data structure. The data structures are considered to be in the same identifier namespace as defined in section 6.2.1 of [RFC7950]. In particular, bullet 7:

> All leafs, leaf-lists, lists, containers, choices, rpcs, actions, notifications, anydatas, and anyxmls defined (directly or through a "uses" statement) within a parent node or at the top level of the module or its submodules share the same identifier namespace.

This means that data structures defined with the "structure" statement cannot have the same name as sibling nodes from regular YANG data definition statements or other "structure" statements in the same YANG module.

This does not mean a YANG data structure, once defined, has to be used as a top-level protocol message or other top-level data structure.

A YANG data structure is encoded in the same way as an "anydata" node. This means that the name of the structure is encoded as a "container", with the instantiated children encoded as child nodes to this node. For example, this structure:

```
module example-errors {
  ...

  sx:structure my-error {
    leaf error-number {
      type int;
    }
  }
}
```

can be encoded in JSON as:

```
"example-errors:my-error": {
  "error-number": 131
}
```

3. YANG Data Structures in YANG Tree Diagrams

A YANG data structure can be printed in a YANG Tree Diagram [RFC8340]. This document updates RFC 8340 by defining two new sections in the tree diagram for a module:

1. YANG data structures, offset by two spaces and identified by the keyword "structure" followed by the name of the YANG data structure and a colon (":") character.

2. YANG data structure augmentations, offset by 2 spaces and identified by the keyword "augment-structure" followed by the augment target structure name and a colon (":") character.

The new sections, including spaces conventions is:

```
  structure <structure-name>:
    +--<node>
       +--<node>
       |  +--<node>
       +--<node>
  structure <structure-name>:
    +--<node>

  augment-structure <structure-name>:
    +--<node>
       +--<node>
       |  +--<node>
       +--<node>
  augment-structure <structure-name>:
    +--<node>
```

Nodes in YANG data structures are printed according to the rules defined in section 2.6 in [RFC8340].

4. YANG Data Structure Extensions Module

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-yang-structure-ext@2019-03-07.yang"

```
module ietf-yang-structure-ext {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-structure-ext";
  prefix sx;

  organization
```

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

       Author:   Andy Bierman
                 <mailto:andy@yumaworks.com>

       Author:   Martin Bjorklund
                 <mailto:mbj@tail-f.com>

       Author:   Kent Watsen
                 <mailto:kent+ietf@watsen.net>";
    description
      "This module contains conceptual YANG specifications for defining
       abstract data structures.

       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) 2019 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 Simplified BSD License set
       forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX
       (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
       for full legal notices.";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.

    revision 2019-03-07 {
      description
        "Initial revision.";
      // RFC Ed.: replace XXXX with RFC number and remove this note
      reference
        "RFC XXXX: YANG Structure Extensions.";
    }
```

```
extension structure {
  argument name {
    yin-element true;
  }
  description
    "This extension is used to specify a YANG data structure that
     represents conceptual data defined in YANG.  It is intended to
     describe hierarchical data independent of protocol context or
     specific message encoding format.  Data definition statements
     within a 'structure' extension statement specify the generic
     syntax for the specific YANG data structure, whose name is the
     argument of the 'structure' extension statement.

     Note that this extension does not define a media-type.  A
     specification using this extension MUST specify the message
     encoding rules, including the content media type, if
     applicable.

     The mandatory 'name' parameter value identifies the YANG data
     structure that is being defined.

     This extension is only valid as a top-level statement, i.e.,
     given as a sub-statement to 'module' or 'submodule'.

     The sub-statements of this extension MUST follow the ABNF
     rules below, where the rules are defined in RFC 7950:

       *must-stmt
       [status-stmt]
       [description-stmt]
       [reference-stmt]
       *(typedef-stmt / grouping-stmt)
       *data-def-stmt

     A YANG data structure defined with this extension statement is
     encoded in the same way as an 'anydata' node.  This means
     that the name of the structure is encoded as a 'container',
     with the instantiated child statements encoded as child nodes
     to this node.

     The module name and namespace value for the YANG module using
     the extension statement is assigned to each of the data
     definition statements resulting from the YANG data structure.

     The XPath document element is the extension statement itself,
     such that the child nodes of the document element are
     represented by the data-def-stmt sub-statements within this
     extension.  This conceptual document is the context for the
```

```
          following YANG statements:

            - must-stmt
            - when-stmt
            - path-stmt
            - min-elements-stmt
            - max-elements-stmt
            - mandatory-stmt
            - unique-stmt
            - ordered-by
            - instance-identifier data type

          The following data-def-stmt sub-statements are constrained
          when used within a 'structure' extension statement.

            - The list-stmt is not required to have a key-stmt defined.
            - The config-stmt is ignored if present.
        ";
      }

      extension augment-structure {
        argument path {
          yin-element true;
        }
        description
          "This extension is used to specify an augmentation to YANG data
           structure defined with the 'structure' statement.  It is
           intended to describe hierarchical data independent of protocol
           context or specific message encoding format.

           This statement has almost the same structure as the
           'augment-stmt'.  Data definition statements within this
           statement specify the semantics and generic syntax for the
           additional data to be added to the specific YANG data
           structure, identified by the 'path' argument.

           The mandatory 'path' parameter value identifies the YANG
           conceptual data node that is being augmented, represented as
           an absolute-schema-nodeid string, where the first node in the
           absolute-schema-nodeid string identifies the YANG data
           structure to augment, and the rest of the nodes in the string
           identifies the node within the YANG structure to augment.

           This extension is only valid as a top-level statement, i.e.,
           given as a sub-statement to 'module' or 'submodule'.

           The sub-statements of this extension MUST follow the ABNF
           rules below, where the rules are defined in RFC 7950:
```

```
  [status-stmt]
  [description-stmt]
  [reference-stmt]
  1*(data-def-stmt / case-stmt)
```

The module name and namespace value for the YANG module using the extension statement is assigned to instance document data conforming to the data definition statements within this extension.

The XPath document element is the augmented extension statement itself, such that the child nodes of the document element are represented by the data-def-stmt sub-statements within the augmented 'structure' statement.

The context node of the 'augment-structure' statement is derived in the same way as the 'augment' statement, as defined in section 6.4.1 of [RFC7950]. This conceptual node is considered the context node for the following YANG statements:

- must-stmt
- when-stmt
- path-stmt
- min-elements-stmt
- max-elements-stmt
- mandatory-stmt
- unique-stmt
- ordered-by
- instance-identifier data type

The following data-def-stmt sub-statements are constrained when used within an 'augment-structure' extension statement.

- The list-stmt is not required to have a key-stmt defined.
- The config-stmt is ignored if present.

Example:

```
   module foo {
      import ietf-yang-structure-ext { prefix sx; }

      sx:structure foo-data {
        container foo-con { }
      }
   }

   module bar {
      import ietf-yang-structure-ext { prefix sx; }
```

```
                import foo { prefix foo; }

                sx:augment-structure /foo:foo-data/foo:foo-con {
                  leaf add-leaf1 { type int32; }
                  leaf add-leaf2 { type string; }
                }
              }
        ";
      }
    }
```

<CODE ENDS>

5. IANA Considerations

5.1. YANG Module Registry

This document registers one URI as a namespace in the "IETF XML Registry" [RFC3688]:

```
URI: urn:ietf:params:xml:ns:yang:ietf-yang-structure-ext
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]:

```
name:         ietf-yang-structure-ext
namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-structure-ext
prefix:       sx
// RFC Ed.: replace XXXX with RFC number and remove this note
reference:    RFC XXXX
```

6. Security Considerations

This document defines YANG extensions that are used to define conceptual YANG data structures. It does not introduce any new vulnerabilities beyond those specified in YANG 1.1 [RFC7950].

7. References

7.1. 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>.

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

[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>.

[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>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

7.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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/info/rfc6020>.

Appendix A. Examples

A.1. "structure" Example

This example shows a simple address book that could be stored as an artifact.

```
module example-module {
  yang-version 1.1;
  namespace "urn:example:example-module";
  prefix exm;

  import ietf-yang-structure-ext {
    prefix sx;
  }

  sx:structure address-book {
    list address {
      key "last first";
      leaf last {
        type string;
        description "Last name";
      }
      leaf first {
        type string;
        description "First name";
      }
      leaf street {
        type string;
        description "Street name";
      }
      leaf city {
        type string;
        description "City name";
      }
      leaf state {
        type string;
        description "State name";
      }
    }
  }
}
```

Below is the tree diagram of this module.

```
module: example-module

  structure address-book:
    +-- address* [last first]
       +-- last      string
       +-- first     string
       +-- street?   string
       +-- city?     string
       +-- state?    string
```

A.2. "augment-structure" Example

This example adds "county" and "zipcode" leafs to the address book:

```
module example-module-aug {
  yang-version 1.1;
  namespace "urn:example:example-module-aug";
  prefix exma;

  import ietf-yang-structure-ext {
    prefix sx;
  }
  import example-module {
    prefix exm;
  }

  sx:augment-structure "/exm:address-book/exm:address" {
    leaf county {
      type string;
      description "County name";
    }
    leaf zipcode {
      type string;
      description "Postal zipcode";
    }
  }
}
```

Below is the tree diagram of this module.

```
module: example-module-aug

  augment-structure /exm:address-book/exm:address:
    +-- county?    string
    +-- zipcode?   string
```

A.3. XML Encoding Example

This example shows how an address book can be encoded in XML:

```
<address-book xmlns="urn:example:example-module">
  <address>
    <last>Flintstone</last>
    <first>Fred</first>
    <street>301 Cobblestone Way</street>
    <city>Bedrock</city>
    <zipcode xmlns="urn:example:example-module-aug">70777</zipcode>
  </address>
  <address>
    <last>Root</last>
    <first>Charlie</first>
    <street>4711 Cobblestone Way</street>
    <city>Bedrock</city>
    <zipcode xmlns="urn:example:example-module-aug">70777</zipcode>
  </address>
</address-book>
```

A.4. JSON Encoding Example

This example shows how an address book can be encoded in JSON:

```
"example-module:address-book": {
  "address": [
    {
      "city": "Bedrock",
      "example-module-aug:zipcode": "70777",
      "first": "Fred",
      "last": "Flintstone",
      "street": "301 Cobblestone Way"
    },
    {
      "city": "Bedrock",
      "example-module-aug:zipcode": "70777",
      "first": "Charlie",
      "last": "Root",
      "street": "4711 Cobblestone Way"
    }
  ]
}
```

A.5. "structure" example that defines a non-top-level structure

The following example defines a data structure with error information, that can be included in an <error-info> element in an <rpc-error>.

```
   module example-error-info {
     yang-version 1.1;
     namespace "urn:example:example-error-info";
     prefix exei;

     import ietf-yang-structure-ext {
       prefix sx;
     }

     sx:structure my-example-error-info {
       leaf error-code {
         type uint32;
       }
     }

   }
```

The example below shows how this structure can be used in an <rpc-error>.

```
   <rpc-reply message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <rpc-error>
       <error-type>protocol</error-type>
       <error-tag>operation-failed</error-tag>
       <error-severity>error</error-severity>
       <error-info>
         <my-example-error-info
             xmlns="urn:example:example-error-info">
           <error-code>42</error-code>
         </my-example-error-info>
       </error-info>
     </rpc-error>
   </rpc-reply>
```

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Cisco

Email: mbj@tail-f.com

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 10, 2018

B. Lengyel
Ericsson
B. Claise
Cisco Systems, Inc.

February 6, 2018

YANG Instance Data Files and their use for Documenting Server Capabilities
draft-lengyel-netmod-yang-instance-data-00

Abstract

This document specifies a standard file format for YANG instance data, that is data that could be stored in a datastore and whose syntax and semantics is defined by YANG models. Instance data files can be used to provide information that is defined in design time. There is a need to document Server capabilities (which are often specified in design time), which should be done using instance data files.

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 August 10, 2018.

Copyright Notice

1. Terminology

Instance Data Set: A named set of data items that can be used as instance data in a YANG data tree.

Instance Data File: A file containing an instance data set formatted according to the rules described in this document.

2. Introduction

A YANG server has a number of server-capabilities that can be retrieved from the server using protocols like NETCONF or RESTCONF. YANG server capabilities include among other things

- data defined in ietf-yang-library (YANG modules, submodules, features, deviations, schema-mounts)
- datastores supported
- alarms supported (draft-vallin-ccamp-alarm-module)
- data nodes, subtrees that support or do not support on-change notifications (draft-ietf-netconf-yang-push)-
- netconf-capabilities
- etc.

While it is good practice to allow a client to query these capabilites from the live YANG server, that is often not enough. Many of these server-capabilities are relatively stable. They may change

1. only at upgrade, or
2. rarely (e.g. due to licensing), or
3. more frequently

Most capabilities belong to type 1), some to type 2) and a relatively small set to type 3). Many network nodes only have type 1) or type 1+2) capabilities. Stable capabilities are usually defined by a vendor in design time, before the product is released. While these

capabilities can be retrieved from the live server in run-time, there is a strong need to provide the same data already during design time. (Often only a part of all the server capabilities can be made available.)

Often when a network node is released an associated NMS (network management system) is also released with it. The NMS depends on the capabilities of the YANG server. During NMS implementation the information about server capabilities is needed. If the information is not available early in some off-line document, but only as instance data from the network node, the NMS implementation will be delayed, because it has to wait for the network node to be ready. Also assuming that all NMS implementors will have a correctly configured network node available to retrieve data from, is a very expensive proposition. (An NMS may handle dozens of node types.)

Beside NMS implementors, system integrators and many others also need the same information early. Examples could be model driven testing, generating documentation, etc.

This document specifies a file format for YANG instance data and proposes to use it to provide server capability information, allowing vendors to specify capabilities together with the YANG modules.

The same instance data file format can be used for other purposes, like providing initial data for any YANG module. E.g. a basic set of access control groups can be provided either by a device vendor or an operator using a network device.

2.1. Data Life cycle

Data defined or documented in YANG Instance Data Sets may be used for preloading a YANG server with this data, but the server may populate the data without using the actual file in which case the Instance Data File is only used as documentation.

While such data will usually not change, a data documented by Instance Data Files MAY be changed by the YANG server itself or by management operations. It is out of scope for this document to specify a method to prevent this.

Notifications about the change of data documented by Instance Data Sets may be supplied by e.g. the Yang-Push mechanism, but it is out of scope for this document.

2.2. Use Case 1: Early Documentation of Server Capabilites

An operator wants to integrate his own in-house built management system with the network node from ACME Systems. The management integration must be ready by the time the first AcmeRouter 9000 is installed in the network. To do the integration the operator needs the the list of supported YANG modules and features. While this list could be read from the ietf-yang-library via Netconf, in order to allow time for developing the management integration, the operator demands this information early. The operator will value that this information is available in a standard format, that is actually the same format he can later read from the node via Netconf.

2.3. Use Case 2: Preloading Data

Defining Access control data is a complex task. To help with this the device vendor pre-defines some of the data. Among others a set of default groups (/nacm:nacm/nacm:groups) are defined e.g. "read-only", "operator", "sys-admin". The operator will often use these default groups, but is also free to completely remove them and define his own set of groups.

3. Instance Data File Format

Two standard formats to represent YANG Instance Data are specified based on the XML and JSON serialization. The XML format is defined in [RFC7950] while the JSON format is defined in [RFC7951] with the additions below.

For both formats data is placed in a top level auxiliary container named "instance-data". The purpose of the container, which is not part of the real data itself, is to contain the potentially multiple top level XML elements and to carry meta-data for the complete instance-data-set.

The XML format SHALL follow the format returned for a NETCONF GET operation. The <instance-data> container SHALL contain all data that would be inside the <data> wrapper element. XML attributes SHOULD NOT be used, however if a SW receiving a YANG instance data file encounters XML attributes it MUST discard them silently, allowing them to be used later for other purposes.

The JSON format SHALL follow the format of the reply returmed for a RESTCONF GET request directed at the datastore resource: {+restconf}/data. ETags and Timestamps SHOULD NOT be included.

A YANG Instance data file MUST contain a single instance data set. Instance data MUST conform to the corresponding YANG Modules.

Default values SHOULD NOT but MAY be included. Config=true and config=false data may be mixed in the instance data file. Instance data files MAY contain partial data sets. This means mandatory, min-elements or require-instance=true constrains MAY be violated.

Meta data, information about the data set itself SHALL be included in the instance data file. Metadata SHALL be formulated following [RFC7952] using the annotations defined in module ietf-yang-instance-data-annotations. All metadata SHOULD be connected to the top level "instance-data" container. Meta data SHALL include:

- Name of the instance data set
- Revision date of the instance data set (later a semantic version MAY also be included)
- Description of the instance data set. The description SHOULD contain information whether and how the data can change during the lifetime of the network element.

Any other metadata may also be included after these items.

```
<instance-data xmlns:ida=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data-annotations"
    ida:instance-data-set="acme-router-modules"
    ida:revision="2108-01-25"
    ida:description="Defines the minimal set of modules that any acme
        will contain. These modules will always be present."
    ida:contact="info@acme.com">
  <modules xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
    <module>
      <name>ietf-system</>
      <revision>2014-08-06</revision>
      <!-- description "A later revision may be used."; -->
      <namespace>urn:ietf:params:xml:ns:yang:ietf-system</namespace>
      <conformance-type>implement</conformance-type>
      <feature>authentication</feature>
      <feature>radius-authentication</feature>
    </module>
  </modules>
</instance-data>
```

Figure 1: XML Instance Data File example

```
{
  "instance-data": {
      "@": {
        "ietf-yang-instance-data-annotations:instance-data-set":
          "acme-router-modules",
        "ietf-yang-instance-data-annotations:revision": "2108-01-25",
        "ietf-yang-instance-data-annotations:contact":
          "info@acme.com",
        "ietf-yang-instance-data-annotations:description":
          "Defines the set of modules that an acme-router will contain."
      },
    "ietf-yang-library:modules-state": {
      "module": [
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement",
          "feature": ["authentication", "radius-authentication"]
        }
      ]
    }
  }
}
```

Figure 2: JSON Instance Data File example

4. Providing Initial Data

YANG instance data files SHOULD be used to provide design time information about server capabilities. The content of the instance data file SHOULD describe the capabilities of the server, however there is no general guarantee that the capabilities will not change over time. Whether capabilities change and if so, when and how SHOULD be described either in the instance data file description statements or some other implementation specific manner. The set of server capabilities to be documented will be defined by other standards and specifications, and is out of scope for this document. Whether and how the instance data files are used by SW implementing a YANG server is out of scope for this specification.

5. YANG Model

<CODE BEGINS> file "ietf-yang-instance-data-annotations.yang"

```
 module ietf-yang-instance-data-annotations {
    yang-version 1.1;
    namespace
```

```
      "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data-annotations";
    prefix ida ;

    import ietf-yang-types { prefix "yang"; }
    import ietf-yang-metadata { prefix "md"; }

    organization  "IETF NETMOD Working Group";
    contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmodf/>

     WG List:  <mailto:netmod@ietf.org>

     Author:  Balazs Lengyel
        <mailto:balazs.lengyel@ericsson.com>";


    description "The module defines annotations to allow defining
       metadata for YANG Instance Data files.";
    reference  "RFC 7950, RFC 7962";

    revision 2017-08-24 {
       description  "Initial revision.";
       reference "RFC XXXX: YANG Instance Data";
    }

    md:annotation instance-data-set {
        type yang:yang-identifier;
       description "Defines the name of a YANG instance data set.

          The annotation SHALL only be used on the top level
          <data> element in RFC XXXX defined YANG Instance Data files.
          Exactly one instance-data-set annotation SHALL be used per
          <data> element.";
     }

     md:annotation contact {
        type string;
       description "Contains the same information the ontact statement
          carries for a YANG module.

          The annotation SHALL only be used on the top level
          <data> element in RFC XXXX defined YANG Instance Data files.
          Zero or one contact annotation SHALL be used per
          <data> element.";
     }

     md:annotation organization {
        type string;
```

```
       description "Contains the same information the organization
          statement carries for a YANG module.

          The annotation SHALL only be used on the top level
          <data> element in RFC XXXX defined YANG instance data files.
          Zero or one organization annotation SHALL be used per
          <data> element.";
     }

     md:annotation revision {
        type string {
           pattern '\d{4}-\d{2}-\d{2}';
        }
       description "Specifies the data the instance-data-set was
          modified for this release.

          The annotation SHALL only be used on the top level
          <data> element in RFC XXXX defined YANG Instance Data files.
          One or more revision annotations SHALL be used per
          <data> element.
          A separate annotation SHOULD be added each time the
          instance-data-set is released.";
     }

     md:annotation description {
        type string;
       description "Defines the name of a YANG instance data set.

          The annotation SHALL be used on the top level <data> element
          in RFC XXXX defined YANG Instance Data files, and MAY be used
          on other data elements of an instance data file.
          Zero one description annotation SHALL be used per element, but
          exactly one description annotation SHALL be used on the top
          level <data> element.";
     }
 }
```

<CODE ENDS>

6. Security Considerations

To be completed

7. IANA Considerations

No IANA action is requested.

8. References

8.1. 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/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

8.2. Informative References

[I-D.ietf-netconf-rfc7895bis] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-04 (work in progress), January 2018.

[I-D.ietf-netconf-yang-push] Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-13 (work in progress), February 2018.

[I-D.vallin-ccamp-alarm-module] Vallin, S. and M. Bjorklund, "YANG Alarm Module", draft-vallin-ccamp-alarm-module-01 (work in progress), October 2017.

[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>.

[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>.

Authors’ Addresses

Balazs Lengyel
Ericsson
xxx
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

Netconf
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2019

B. Lengyel
Ericsson
B. Claise
Cisco Systems, Inc.

October 19, 2018

YANG Based Instance Data Files Format
draft-lengyel-netmod-yang-instance-data-05

Abstract

There is a need to document data defined in YANG models without the need to fetch it from a live YANG server. Data is often needed already in design time or needed by groups that do not have a live running YANG server available. This document specifies a standard file format for YANG Based Instance data, that is data that could be stored in a datastore and whose syntax and semantics is defined by YANG models. Most important use cases foreseen include documenting server capabilities, factory-default settings, or vendor provided default configurations.

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 April 22, 2019.

Table of Contents

1. 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 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

Design time: A time during which a YANG model and the implementation behind it is created. Sometimes in other documents this period is divided into design and implementation time.

Instance Data Set: A named set of data items that can be used as instance data in a YANG data tree.

Instance Data File: A file containing an instance data set formatted according to the rules described in this document.

Target YANG Module: A YANG module for which the instance data set contains instance data, like ietf-yang-library in the examples.

2. Introduction

There is a need to provide instance data defined in YANG models without the need to fetch it from a live YANG server. Data is often needed already in design time before the YANG server is implemented or needed by groups that do not have a live running YANG server available. To facilitate this off-line delivery of data this document specifies a standard file format for YANG Based Instance data, that is data that could be stored in a datastore and whose syntax and semantics is defined by YANG models.

2.1. Use Cases

We present a number of use cases were Yang based instance data is needed.

2.1.1. Use Case 1: Early Documentation of Server Capabilites

A YANG server has a number of server-capabilities that are defined in YANG modules and can be retrieved from the server using protocols like NETCONF or RESTCONF. YANG server capabilities include

- data defined in ietf-yang-library: YANG modules, submodules, features, deviations, schema-mounts, datastores supported ([I-D.ietf-netconf-rfc7895bis])
- alarms supported ([I-D.ietf-ccamp-alarm-module])
- data nodes, subtrees that support or do not support on-change notifications ([I-D.ietf-netconf-yang-push])
- netconf-capabilities in ietf-netconf-monitoring

While it is good practice to allow a client to query these capabilites from the live YANG server, that is often not enough.

Often when a network node is released an associated NMS (network management system) is also released with it. The NMS depends on the capabilities of the YANG server. During NMS implementation information about server capabilities is needed. If the information is not available early in some off-line document, but only as instance data from the live network node, the NMS implementation will be delayed, because it has to wait for the network node to be ready. Also assuming that all NMS implementors will have a correctly configured network node available to retrieve data from, is a very expensive proposition. (An NMS may handle dozens of node types.)

Network operators often build their own home-grown NMS systems that needs to be integrated with a vendor's network node. The operator needs to know the network node's server capabilities in order to do this. Moreover the network operator's decision to buy a vendor's product may even be influenced by the network node's OAM feature set documented as the Yang server's capabilites.

Beside NMS implementors, system integrators and many others also need the same information early. Examples could be model driven testing, generating documentation, etc.

Most server-capabilities are relatively stable and change only during upgrade or due to licensing or addition or removal of HW. They are usually defined by a vendor in design time, before the product is released. It feasible and advantageous to define/document them early e.g. in a Yang Based Instance Data File.

It is anticipated that a separate IETF document will define in detail how and which set of server capabilites should be documented.

2.1.2. Use Case 2: Preloading Data

There are parts of the configuration that must be fully configurable by the operator, however for which often a simple default configuration will be sufficient.

One example is access control groups/roles and related rules. While a sophisticated operator may define dozens of different groups often a basic (read-only operator, read-write system administrator, security-administrator) triplet will be enough. Vendors will often provide such default configuration data to make device configuration easier for an operator.

Defining Access control data is a complex task. To help the device vendor pre-defines a set of default groups (/nacm:nacm/groups) and rules for these groups to access specific parts of common models (/nacm:nacm/rule-list/rule).

YANG Based Instance data files are used to document and/or preload the default configurationp.

2.1.3. Use Case 3: Dcoumenting Factory Default Settings

Nearly every YANG server has a factory default configuration. If the system is really badly misconfigured or if the current configuration is to be abandoned the system can be reset to this default.

In Netconf the <delete-config> operation can already be used to do this for the startup configuration. There are ongoing efforts to introduce a new, more generic reset operation for the same purpose [I-D.wu-netconf-restconf-factory-restore]

The operator currently has no way to know what the default configuration actually contains. YANG Based Instance data can be used to document the factory default configuration.

3. Instance Data File Format

Two standard formats to represent YANG Based Instance Data are specified based on the XML and JSON encoding. The XML format is based on [RFC7950] while the JSON format is based on [RFC7951]. Later as other YANG encodings (e.g. CBOR) are defined further Instance Data formats may be specified.

For both formats data is placed in a top level auxiliary container named "instance-data-set". The purpose of the container, which is not part of the real data itself, is to carry meta-data for the complete instance-data-set.

The XML format SHALL follow the format returned for a NETCONF GET operation. The <data> anydata (which is not part of the real data itself) SHALL contain all data that would be inside the <data> wrapper element of a reply to the <get> operation. XML attributes SHOULD NOT be present, however if a SW receiving a YANG Based Instance data file encounters XML attributes unknown to it, it MUST ignore them, allowing them to be used later for other purposes.

The JSON format SHALL follow the format of the reply returmed for a RESTCONF GET request directed at the datastore resource: {+restconf}/data. ETags and Timestamps SHOULD NOT be included, but if present SHOULD be ignored.

A YANG Based Instance data file MUST contain a single instance data set. Instance data MUST conform to the corresponding target YANG Modules and follow the XML/JSON encoding rules as defined in [RFC7950] and [RFC7951] and use UTF-8 character encoding. A single instance data set MAY contain data for any number of target YANG modules, if needed it MAY carry the complete configuraton and state data set for a YANG server. Default values SHOULD NOT but MAY be included. Config=true and config=false data MAY be mixed in the instance data file. Instance data files MAY contain partial data sets. This means mandatory, min-elements or require-instance=true constrains MAY be violated.

The name of the file SHOULD be of the form:

instance-data-set-name ['@' revision-date] ('.yid')

E.g. acme-router-modules@2018-01-25.yid

The revision date is optional. It SHOULD NOT be used if the file is stored in a version control system (e.g. git) because the change of file names will break the connection between the different revisions of the file.

Meta data, information about the data set itself SHALL be included in the instance data set. This data will be children of the top level instance-data-set container as defined in the ietf-instance-data YANG module. Meta data SHALL include:

- Name of the instance data set

Meta data SHOULD include:

- Revision date of the instance data set
- Description of the instance data set. The description SHOULD contain information whether and how the data can change during the lifetime of the YANG server.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-modules</name>
  <revision>2108-01-25</revision>
  <description>Defines the minimal set of modules that any acme-router
       will contain. These modules will always be present.</description>
  <contact>info@acme.com</contact>
  <data>
    <yang-library xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module-set>
        <name>basic</name>
        <module>
          <name>ietf-system</>
          <revision>2014-08-06</revision>
          <!-- description "A later revision may be used."; -->
          <namespace>urn:ietf:params:xml:ns:yang:ietf-system</namespace>
          <feature>authentication</feature>
          <feature>radius-authentication</feature>
        </module>
      </module-set>
    </yang-library>
  </data>
</instance-data-set>
```

Figure 1: XML Instance Data File example

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "acme-router-modules",
    "revision": "2108-01-25",
    "contact": "info@acme.com",
    "description":
      "Defines the set of modules that an acme-router will contain.",
    "data": {
      "ietf-yang-library:yang-library": {
        "module-set": [
          "name": "basic",
          "module": [
            {
              "name": "ietf-system",
              "revision": "2014-08-06",
              "namespace": "urn:ietf:params:xml:ns:yang:ietf-system",
              "feature": ["authentication", "radius-authentication"]
            }
          }
        ]
      ]
    }
  }
}
```

Figure 2: JSON Instance Data File example

4. Data Life cycle

Data defined or documented in YANG Based Instance Data Sets may be used for preloading a YANG server with this data, but the server may populate the data without using the actual file in which case the Instance Data File is only used as documentation.

While such data will usually not change, data documented by Instance Data sets MAY be changed by the YANG server itself or by management operations. It is out of scope for this document to specify a method to prevent this. Whether such data changes and if so, when and how, SHOULD be described either in the instance data file description statement or in some other implementation specific manner.

YANG Based Instance data is a snap-shot of information at a specific point of time. If the data changes afterwards this is not represented in the instance data set anymore, the valid values can be retrieved in run-time via Netconf/Restconf

Notifications about the change of data documented by Instance Data Sets may be supplied by e.g. the Yang-Push mechanism, but it is out of scope for this document.

5. Delivery of Instance Data

Instance data files SHOULD be available without the need for a live YANG server e.g. via download from the vendor's website, or any other way together with other product documentation.

6. YANG Model

<CODE BEGINS> file "ietf-yang-instance-data.yang"

```
module ietf-yang-instance-data {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data";
  prefix yid ;

  import ietf-yang-data-ext { prefix yd; }

  import ietf-datastores { prefix ds; }

  organization  "IETF NETMOD Working Group";
  contact
   "WG Web:   <https://datatracker.ietf.org/wg/netmodf/>
    WG List:  <mailto:netmod@ietf.org>

    Author:  Balazs Lengyel
       <mailto:balazs.lengyel@ericsson.com>";

  description "The module defines the structure and content of YANG
       Instance Data Sets.";

  revision 2018-06-30 {
    description  "Initial revision.";
    reference "RFC XXXX: YANG Based Instance Data";
  }

  yd:yang-data instance-data-format {
    container instance-data-set {
      description "Auxiliary container to carry meta-data for
        the complete instance data set.";

      leaf name {
        type string;
        mandatory true;
```

```
          description "Name of a YANG Based Instance data set.";
        }

        leaf description { type string; }

        leaf contact {
          type string;
          description "Contains the same information the contact
              statement carries for a YANG module.";
        }

        leaf organization {
          type string;
          description "Contains the same information the
              organization statement carries for a YANG module.";
        }

        leaf datastore {
          type ds:datastore-ref;
          description  "The identity of the datastore for which
            the instance data is documented for config=true data nodes.
            The leaf MAY be absent in which case the running dtastore or
            if thats not writable, the candidate datastore is implied.

            For config=false data nodes always the operational
            data store is implied.";
        }

        list revision {
          key date;
          description "An instance-data-set SHOULD have at least
              one revision entry.  For every published
              editorial change, a new one SHOULD be added in front
              of the revisions sequence so that all revisions are
              in reverse chronological order.";

          leaf date {
            type string {
              pattern '\d{4}-\d{2}-\d{2}';
            }
            description "Specifies the data the revision
                was last modified. Formated as YYYY-MM-DD";
          }

          leaf description { type  string; }
        }

        anydata data {
```

```
          mandatory true;
          description "Contains the real instance data.
              The data MUST conform to the relevant YANG Modules.";
        }
      }
    }
  }
```

<CODE ENDS>

7. Security Considerations

Depending on the nature of the instance data, instance data files MAY need to be handled in a secure way. The same type of handling should be applied, that would be needed for the result of a <get> operation returning the same data.

8. IANA Considerations

To be completed, all the usual requests for a new YANG module

9. References

9.1. Normative References

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Extensions", draft-ietf-netmod-yang-data-ext-01 (work in progress), March 2018.

[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>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

9.2. Informative References

[I-D.ietf-ccamp-alarm-module]
Vallin, S. and M. Bjorklund, "YANG Alarm Module", draft-ietf-ccamp-alarm-module-04 (work in progress), October 2018.

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-19 (work in progress), September 2018.

[I-D.wu-netconf-restconf-factory-restore]
Wu, Q., Lengyel, B., and Y. Niu, "Factory default Setting", draft-wu-netconf-restconf-factory-restore-03 (work in progress), October 2018.

[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>.

[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>.

Appendix A. Open Issues

- If we define metadata per target module, a list of target YAM could be included in the metadata. This depends on what additional metadata we will include.

- How do we know for which version of the target Yang Module is a data set valid? Proposal: One possibility would be to just indicate for which module version(s) was the data set last updated. This would be a hint about compatibility, but nothing more. Maybe we should wait till the YANG versioning work is complete/stable. Identifying just one version is way to strict, so something enforcing that shall not be used.

- Should we document what YANG features does the instance data set implicitly require? Proposal: that is already a use case, documenting data from the YANG library.

- Augmenting metadata must be possible. As of now it looks like yang-data-ext will solve that. If not, define instance data as regular YANG instead of yd:yang-data.

Appendix B. Changes between revisions

v04 - v05

- Changed title and introduction to clarify that this draft is only about the file format and documenting server capabilities is just a use case.
- Added reference to draft-wu-netconf-restconf-factory-restore
- Added new open issues.

v03 - v04

- Updated changelog for v02-v03

v02 - v03

- Updated the document according to comments received at IETF102
- Added parameter to specify datastore
- Rearranged chapters
- Added new use case: Documenting Factory Default Settings
- Added "Target YANG Module" to terminology
- Clarified that instance data is a snapshot valid at the time of creation, so it does not contain any later changes.
- Removed topics from Open Issues according to comments received at IETF102

v01 - v02

- The recommendation to document server capabilities was changed to be just the primary use-case. (Merged chapter 4 into the use case chapter.)
- Stated that RFC7950/7951 encoding must be followed which also defines (dis)allowed whitespace rules.
- Added UTF-8 encoding as it is not specified in t950 for instance data
- added XML declaration

v00 - v01

- Redefined using yang-data-ext

- Moved meta data into ordinary leafs/leaf-lists

Authors' Addresses

Balazs Lengyel
Ericsson
Magyar Tudosok korutja 11
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2018

N. Sambo
P. Castoldi
Scuola Superiore Sant'Anna
G. Fioccola
Telecom Italia
F. Cugini
CNIT
H. Song
T. Zhou
Huawei

March 2, 2018

YANG model for finite state machine
draft-sambo-netmod-yang-fsm-02

Abstract

Network operators and service providers are facing the challenge of deploying systems from different vendors while looking for a trade-off among transmission performance, network device reuse, and capital expenditure without the need of being tied to single vendor equipment. The deployment and operation of more dynamic and programmable network infrastructures can be driven by adopting model-driven and software-defined control and management paradigms. In this context, YANG enables to compile a set of consistent vendor-neutral data models for networks and components based on actual operational needs emerging from heterogeneous use cases. This document proposes YANG models to describe events, operations, and finite state machine of YANG-defined network elements. The proposed models can be applied in several use cases: i) in the context of optical networks to pre-instruct data plane devices (e.g., an optical transponder) on the actions to be performed (e.g., code adaptation) in case some events, such as physical layer degradations, occur; ii) in general data networks, network telemetry applications can define and embed custom data probes into data plane devices. A probe in many cases can be modeled as an FSM; iii) the monitoring of packet loss and delay through a network clustering approach.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Networks are evolving toward more programmability, flexibility, and multi-vendor interoperability. Multi-vendor interoperability can be applied in the context of nodes, i.e. a node composed of components provided by different vendors (named fully disaggregated white box) is assembled under the same control system. This way, operators can optimize costs and network performance without the need of being tied to single vendor equipment. NETCONF protocol RFC6241 [RFC6241] based on YANG data modeling language RFC6020 [RFC6020] is emerging as a candidate Software Defined Networking (SDN) enabled protocol. First, NETCONF supports both control and management functionalities, thus permits high programmability. Then, YANG enables data modeling in a vendor-neutral way. Some recent works have provided YANG models to describe attributes of links (e.g., identification), nodes (e.g., connectivity matrix), media channels, and transponders (e.g., supported forward error correction - FEC) of networks ([I-D.ietf-i2rs-yang-network-topo] [I-D.vergara-ccamp-flexigrid-yang] [I-D.zhang-ccamp-l1-topo-yang]), also including optical technologies. This document presents YANG models to describe events, operations, and finite state machine of YANG-defined network elements. Such models can be applied to several use cases. In the context of elastic optical networks (EONs), the model enables a centralized remote network controller (managed by a network operator) to instruct a transponder controller about the actions to perform when certain events (e.g., failures) occur. The actions to be taken and the events can be re-programmed on the device. In general data networks, programmable network telemetry is considered a killer SDN application which can help applications gain unprecedented visibility to network data plane. Instead of providing raw data, network devices can be configured to filter and process data directly on the data plane and only hand preprocessed data to the collector, in order to save data bandwidth and reduce reaction delay ([I-D.song-opsawg-dnp4iq]) . Such configurations can be programed as custom probes and dynamically deployed into data plane devices. A probe in many cases can be modeled as an FSM. Another use case is the monitoring of packet loss and delay through a network clustering approach: in this case, each FSM state is determined by a specific subdivision of the network in Clusters ([I-D.fioccola-ippm-multipoint-alt-mark]).

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [RFC2119].

3. Terminology

ABNO: Application-Based Network Operations

BER: Bit Error Rate

EON: Elastic Optical Network

FEC: Forward Error Correction

FSM: Finite State Machine

NETCONF: Network Configuration Protocol

OAM: Operation Administration and Maintenance

SDN: Software Defined Network

YANG: Yet Another Network Generator

DNP: Dynamic Network Probe

AMM: Alternate Marking Method

4. Example of application

4.1. Pre-programming resiliency schemes in EONs

EONs (optical networks based on flexible grid supporting circuits of different bandwidth) are expected to employ flexible transponders, i.e. transponders supporting multiple bit rates, multiple modulation formats, and multiple codes. Such transponders permits the (re-) configuration of the bit rate value based on traffic requirements, as well as the configuration of the modulation format and code based on the physical characteristics of a path (e.g., quadrature phase shift keying is more robust than 16 quadrature amplitude modulation). This way, transmission parameters can be (re-) configured based on physical layer changes. The YANG model presented in this draft enables to pre-program reconfiguration settings of data plane devices in case of failures or physical layer degradations. In particular, soft failures are assumed. Soft failures imply transmission performance degradation, in turns a bit error rate (BER) increase, e.g. due to the ageing of some network devices. Without loosing generality, the ABNO architecture is assumed for the control and management of EONs (RFC7491 [RFC7491]). Considering the state of the art, when pre-FEC BER passes above a predefined threshold, it is expected that an alarm is sent to the OAM Handler, which communicates with the ABNO controller that may trigger an SDN controller (that

could be the Provisioning Manager of ABNO RFC7491 [RFC7491]) for computing new transmission parameters. The involved ABNO modules are shown in the simplified ABNO architecture of Fig. 1. Then, transponders are reconfigured. When alarms related to several connections impacted by the soft failure are generated, this procedure may be particularly time consuming. The related workflow for transponder reconfiguration is shown in Fig. 2. The proposed model enables an SDN controller to instruct the transponder about reconfiguration of new transmission parameters values if a soft failure occurs. This can be done before the failure occurs (e.g., during the connection instantiation phase or during the connection service), so that data plane devices can promptly reconfigure themselves without querying the SDN controller to trigger an on-demand recovery. This is expected to speed up the recovery process from soft failures. The related flow chart is shown in Fig. 3.

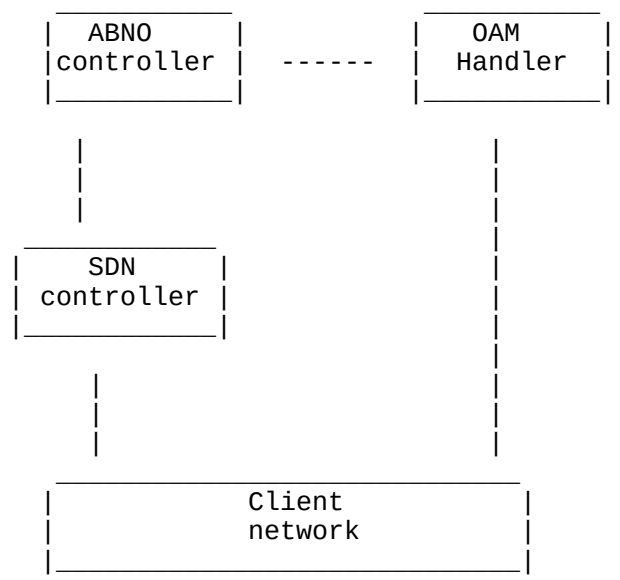

Figure 1: Assumed ABNO functional modules

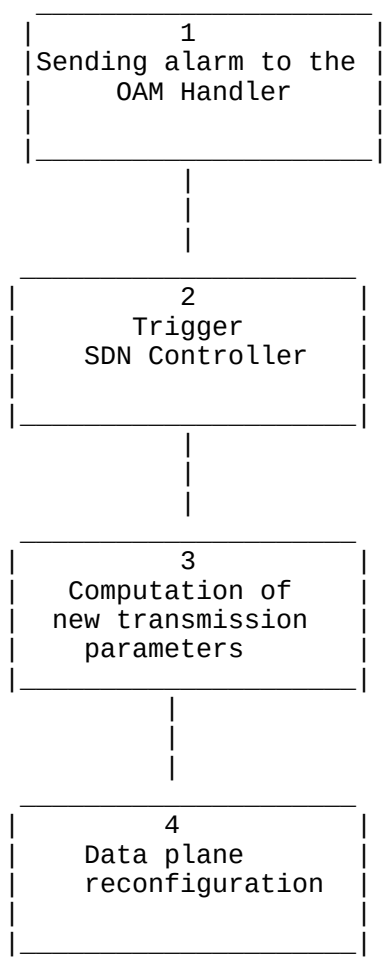

Figure 2: Flow chart of the expected state-of-the-art approach

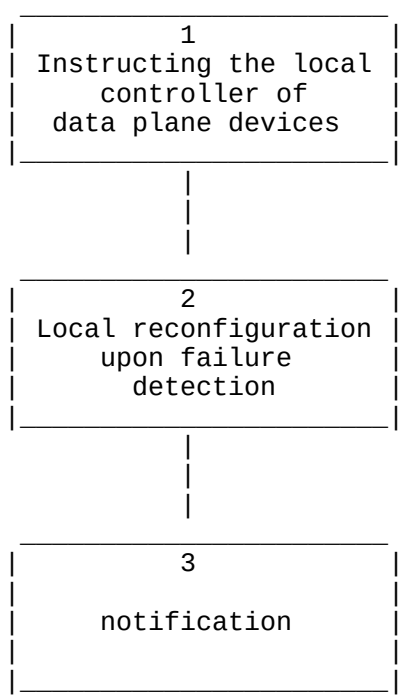

Figure 3: Flow chart of the approach exploiting YANG models in this draft

4.2. Deploying Dynamic Probes for Programmable Network Telemetry

In the past, network data analytics was considered a separate function from networks. They consume raw data extracted from networks through piecemeal protocols and interfaces. With the advent of user programmable data plane, we expect a paradigm shift that makes the data plane be an active component of the data telemetry and analytics solution. The programmable in-network data preprocessing is efficient and flexible to offload some light-weight data processing through dynamic data plane programming or configuration. A universal network data analytics platform built on top of this enables a tight and agile network control and OAM feedback loop. A proposed dynamic network telemetry system architecture is illustrated in Fig.4.

An application translates its data requirements into a set of Dynamic Network Probes (DNP) targeting a subset of data plane devices. After the probes are deployed, each probe conducts its corresponding in-network data preprocessing and feeds the preprocessed data to the

collector. The collector finishes the data post-processing and presents the results to the data-requesting application.

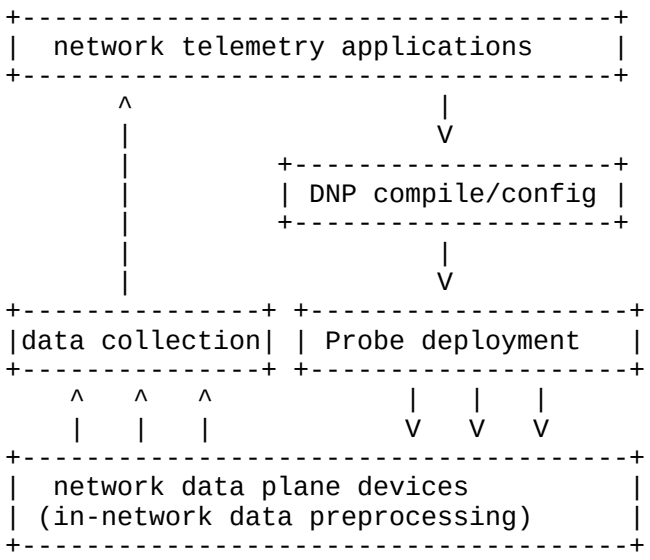

Figure 4: Deploy dynamic network probes using YANG FSM models

Many DNPs can be modeled as FSM which are configured to capture specific events. Here FSMs essentially preprocess the raw stream data and only report the necessary data to subscribing applications.

For example, a congestion control application needs to monitor the router buffer occupancy. Instead of polling the buffer depth periodically, it is only interested in the real-time events when the buffer depth crosses a low and a high threshold. We can install a probe to achieve this data plane function and the probe can be modeled as a three-state FSM. Each state represents a buffer region: below the low threshold, above the high threshold, and in between the two thresholds. A possible state transition is checked against the buffer depth for each incoming and outgoing packet. Whenever a state transition happens, an event is generated and reported to the application. This approach significantly reduces the amount of data sent to the application and also allows a timely event notification.

For another example, an application would like to monitor the delay experienced by a flow. The packet delay on its forwarding path can be acquired by using iOAM [I-D.brockners-inband-oam-requirements]. However, the application only needs to know that N consecutive flow packets experience a delay longer than T. Instead of forwarding the

raw delay data to the application, a probe can be deployed to detect the event. Similarly, the probe can be modeled as an FSM.

4.3. IP Performance Measurements on multipoint-to-multipoint large Networks

Networks offer rich sets of network performance measurement data, but traditional approaches run into limitations. One reason for this is the fact that in many cases, the bottleneck is the generation and export of the data and the amount of data that can be reasonably collected from the network runs into bandwidth and processing constraints in the network itself. In addition, management tasks related to determining and configuring which data to generate lead to significant deployment challenges.

In order to address these issues, an SDN controller application orchestrates network performance measurements tasks across the network to allow an optimized monitoring. In fact the IP Performance Measurement SDN Controller Application in Figure 5 can calibrate how deep can be obtained monitoring data from the network by configuring measurement points roughly or meticulously. This can be established by using the feedback mechanism reported in Figure 5.

For instance, the SDN Controller can configure initially an end to end monitoring between ingress points and egress points of the network. If the network does not experiment issues, this approximate monitoring is good enough and is very cheap in terms of network resources. But, in case of problems, the SDN Controller becomes aware of the issues from this approximate monitoring and, in order to localize the portion of the network that has issues, configures the measurement points more exhaustively. So a new detailed monitoring is performed. After the detection and resolution of the problem the initial approximate monitoring can be used again. This idea is general and can be applied to different performance measurements techniques both active and passive (and hybrid).

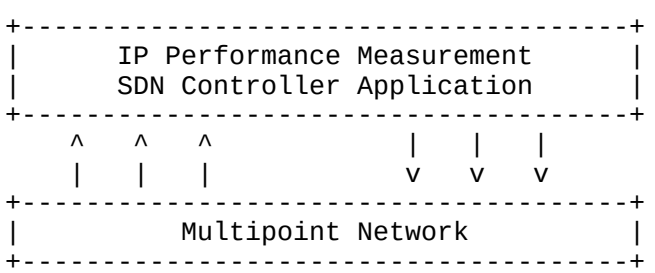

Figure 5: Feedback mechanism on multipoint-to-multipoint large Networks

One of the most efficient methodology to perform packet, loss delay and jitter measurements both in an IP and Overlay Networks is the Alternate Marking method, as presented in [I-D.ietf-ippm-alt-mark] and [I-D.fioccola-ippm-multipoint-alt-mark].

This technique can be applied to point-to-point flows but also to multipoint.to-multipoint flows (see [I-D.ietf-ippm-alt-mark] and [I-D.fioccola-ippm-multipoint-alt-mark]). The Alternate Marking method creates batches of packets by alternating the value of 1 or 2 bits of the packet header. These batches of packets are unambiguously recognized over the network and the comparison of packet counters permits the packet loss calculation. The same idea can be applied for delay measurement by selecting special packets with a marking bit dedicated for delay measurements. This method needs two counters each marking period for each flow under monitor. For this reason by considering n measurement points and n monitored flows, the order of magnitude of the packet counters for each time interval is n*n*2 (1 per color).

Multipoint Alternate Marking, described in [I-D.fioccola-ippm-multipoint-alt-mark], aims to reduce this value and makes the performance monitoring more flexible in case a detailed analysis is not needed.

It is possible to monitor a Multipoint Network without examining in depth by using the Network Clustering (subnetworks that are portions of the entire network that preserve the same property of the entire network). So in case there is packet loss or the delay is too high the filtering criteria could be specified more in order to perform a per flow detailed analysis, as described in [I-D.ietf-ippm-alt-mark].

An application of the multipoint performance monitoring can be done by using FSM (each state is a composition of clusters) and feedback mechanism where the SDN Controller is the brain of the network and can manage flow control to the switches and routers and, in the same way, can calibrate the performance measurements depending on the necessity.

5. YANG for finite state machine (FSM)

This model defines a list of states and transitions to describe a generic finite state machine (FSM). The related code and tree are shown in the Appendix.

<current-state>: it defines the current state of the FSM.
<states>: this element defines the FSM as follows.
<state>: this list defines all the FSM states.
<id>: this leaf attribute of <state> defines the

identifier of the state
<name>: this leaf attribute of <state> defines the name of the state
<description>: this leaf is a "string" describing the state
<transitions>: this attribute defines a list of transitions to other states in the FSM.
<name>: this attribute defines the name of a transition
<type>: this attribute defines the type of the transition from a pool of possible transition types predefined inside the YANG model. Together with the <name> attribute, it uniquely identifies the transition.
<description>: this optional attribute is a "string" describing the transition
<filters>: this leaf is a list of input parameters related to the transition. This attribute enables to further express a transition: as an example, if a transition can be triggered by a parameter (e.g., a monitored performance parameter) exceeding a threshold (as in Sec. 5), an element of the list defines this threshold. Thus, if the parameter is outside the threshold, the transition is taken, otherwise not.
<filter>: this leaf of <filters> defines a filter parameter.
<filter-id>: this leaf of <filters> define the identifier number associated with the <filter> attribute.
<actions>: this attribute defines a list of actions to take during the transition.
<action>: this attribute is the list of actions
<id>: this leaf of <action> defines the identifier number of an action.
<type>: this leaf of <action> defines the type of an action.
<simple>: this leaf defines (differently from <conditional> detailed below) an action that has to be directly executed.
<execute>: this attribute recalls an RPC encapsulating the effective task (action) to be executed by the

hardware. If more actions (e.g., "A" and "B"), defined in the <action> list, have to be executed, these actions can be executed sequentially according to the <next-action> attribute detailed below. Thus, by referring to the tree of the Appendix, when an action ("A") is executed, the <next-action> attribute will bring to another action ("B"). If more actions have to be executed in parallel (e.g., "A" & "B"), not sequentially, an element of the <action> list should be defined to express an action (e.g., "A&B") consisting of more actions to be executed in parallel.
<next-action>: this attribute defines the identification number of a next action that has to be taken. The <next-action> can assume a NULL value.
<conditional>: this leaf enables a check ("true" or "false") to be verified before executing the action. Based on the check, the proper attributes <execute> and <next-operation> are considered.
<statement>: this leaf of <conditional> defines the condition to be verified before executing the action.
<true>: this leaf of <conditional> defines a result of the check associated to <statement>. Proper <execute> and <next-operation> attributes are associated with this result of the

check.
<false>: this leaf of
<conditional> defines a
result of the check
associated to
<statement>. Proper
<execute> and
<next-operation>
attributes are associated
with this result of the
check.
<next-state>: this attribute defines the next state of FSM when an action is executed.

6. Implementation of the pre-programming resiliency schemes in EONs

These presented model can be used to enable a centralized network controller, managed by a network operator, to instruct data plane hardware on its reconfiguration if some events, such as a failure or physical layer degradation, occur. As an example, an optical signal impacted by a soft failure (i.e., a physical layer degradation inducing a pre forward error correction bit error rate increase - pre-FEC) can be maintained by adapting the FEC of the signal itself. This action to be taken and, more in general operations to be executed depending on critical events, can be (re-) programmed on the transponder by (re-) sending a NETCONF <edit-config> message to the device controller including a FSM defined by the YANG model. Such a system has the main goal to speed up the reaction of the network to certain events/faults and to alleviate the workload of the centralized controller. The speed up derives from the fact that the centralized controller is able to pre-compute and pre-configure on the network devices the actions to take when an event occurs taking into account a global view and knowledge of the network. In this way, the device is already aware of the actions to be locally applied to reconfigure a connection, avoiding to inform the controller and to wait for the response indicating what to do. Consequently, part of the workload is also removed from the centralized controller. When the reaction is successfully completed in the data plane, the centralized controller can be notified about the faults and the taken action. A flexible transponder supporting two FEC types, 7% and 20%, is considered. A two-states FSM is also assumed. The states have <name> attribute set to "Steady" and "Fec-Baud-Adapt", respectively. In the "Steady" state, the signal is in a healthy condition, adopting a 7% FEC, with a pre-FEC BER below an assigned threshold of 9 x 10-4. A transition from this state can be triggered by the event with <name>=BER_CHANGE and <filter-type>=9 x 10-4, thus expressing a change of the pre-FEC BER above the threshold. In case the pre-FEC

BER exceeds 9 x 10-4 due to a soft failure, the state machine evolves to the "Fec-Baud-Adapt" state and an adaptation to a more robust FEC of 20% (executed by the attribute <execute>) is performed. The system can return to the "Steady" state if the pre-FEC BER goes below another pre-defined threshold and the FEC is reconfigured to 7%.

7. Appendix

This appendix reports the YANG models code and the related tree.

7.1. YANG model for FSM - Tree

```
module: ietf-fsm
   +--rw current-state?   leafref
   +--rw states
      +--rw state [id]
         +--rw id             state-id-type
         +--rw description?   string
         +--rw transitions
            +--rw transition [name type]
               +--rw name           string
               +--rw type           transition-type
               +--rw description?   string
               +--rw filters
               |  +--rw filter [filter-id]
               |     +--rw filter-id    uint32
               +--rw actions
                  +--rw action [id]
                     +--rw id             transition-id-type
                     +--rw type           enumeration
                     +--rw conditional
                     |  +--rw statement    string
                     |  +--rw true
                     |  |  +--rw execute
                     |  |  +--rw next-action?   transition-id-type
                     |  |  +--rw next-state?    leafref
                     |  +--rw false
                     |     +--rw execute
                     |     +--rw next-action?   transition-id-type
                     |     +--rw next-state?    leafref
                     +--rw simple
                        +--rw execute
                        +--rw next-action?   transition-id-type
                        +--rw next-state?    leafref
```

7.2. YANG model for FSM - Code

<CODE BEGINS> file "ietf-fsm@2016-03-15.yang"

```
 module ietf-fsm {
   namespace "http://sssup.it/fsm";
   prefix fsm;



   identity TRANSITION {
       description "Base for all types of event";
   }

   identity ON_CHANGE {
       base TRANSITION;
       description
         "The event when the database changes.";
   }

   // typedef statements

   typedef transition-type {
        description "it defines the type of transition (event)";
     type identityref {
       base TRANSITION;
```

```
     }

   }



   typedef transition-id-type {
        description "it defines the id of the transition (event)";


     type uint32;

   }




   // grouping statements

   grouping action-block {

        description "it defines the action to perform when a transition occurs";

     leaf id {

description "it refers to the id of the transition";
       type transition-id-type;

     }

     leaf type {

description "it defines if the action has to be simply executed or if a conditio
nal statement has to be checked before execution";

       type enumeration {

         enum "CONDITIONAL_OP" {
description "it defines the type CONDITIONAL OPERATION to check a statement befo
re execution";
}

         enum "SIMPLE_OP" {
description "it defines the type SIMPLE OPERATION: i.e., an operation to be dire
ctly executed;
               }

       }

       mandatory true;
```

```
     }

     grouping execution-top {
        description "it defines the execution attribute";
       anyxml execute {
         description "Represent the action to perform";
       }
       leaf next-action {
         type transition-id-type;
         description "the id of the next action to execute";

       }
     }

     container conditional {
        description "it defines the container CONDITIONAL";
       when "../type = 'CONDITIONAL_OP'";
       leaf statement {
         type string;
         mandatory true;
         description
           "The statement to be evaluated before execution.
           E.g. if a=b";
       }
       container true {
```

```
description "it is referred to the result TRUE of a conditional statement ";
         uses execution-top;
       }
       container false {
description "it is referred to the result FALSE of a conditional statement ";
         uses execution-top;
       }
     }


     container simple {
description "Simple execution of an action without checking any condition";
       when "../type = 'SIMPLE_OP'";

       uses execution-top;
     }
   }


   grouping action-top {

description "it defines the grouping of action";
     list action {

description "it defines the list of actions";
       key "id";
       ordered-by user;
       uses action-block;
```

```
    }
  }

  grouping on-change {
    description
      "Event occuring when a modification of one or more
       objects occurs";

    container filters {
      description
        "This container contains a list of configurable filters
         that can be applied to subscriptions.  This facilitates
         the reuse of complex filters once defined.";
      list filter {
        key "filter-id";

        description
          "A list of configurable filters that can be applied to
           subscriptions.";
        leaf filter-id {
          type uint32;
          description
            "An identifier to differentiate between filters.";
        }
      }
```

```
     }
   }

   grouping transition-top {
description "it defines the grouping transition";
     leaf name {
description "it defines the transition name";
       type string;
       mandatory true;
     }

     leaf type {

description "it defines the transition type";
       type transition-type;
       mandatory true;
     }

     leaf description {

description "it describes the transition ";
       type string;
     }

     // list of all possible events
```

```
     uses on-change {
       when "type = 'ON_CHANGE'";
     }

     container actions {
description "it defines the container action";
       uses action-top;
     }
   }

   grouping transitions-top {
description "it defines the grouping transition";
     container transitions {
description "it defines the container transitions";
       list transition {
description "it defines the list of transitions";
         key "name type";
         uses transition-top;
       }
     }
   }

   // data definition statements
```

```
   uses transitions-top;

   // extension statements

   // feature statements

   // augment statements


   organization
     "Scuola Superiore Sant'Anna Network and Services Laboratory";

   contact
     " Editor: Matteo Dallaglio
               <mailto:m.dallaglio@sssup.it>
     ";

   description
     "This module contains a YANG definitions of a generic finite state
     machine.";

   revision 2016-03-15 {
     description "Initial Revision.";
     reference
       "RFC xxxx:";
```

```
   }

   // identity statements

   // typedef statements

   typedef state-id-type {
description "it defines the id type of the states";
     type uint32;
   }

   // grouping statements
   grouping state-top {
description "it defines the grouping state";
     leaf id {
description "it defines the id of a transition";
       type state-id-type;
     }

     leaf description {
description "it describes a transition";
       type string;
```

```
     }



     grouping next-state-top {
description "it defines the grouping for the next state";
       leaf next-state {
description "it defines the next state";
           type leafref {
description "it refers to its id";
             path "../../../../../../../../../states/state/id";
           }
           description "Id of the next state";
         }
     }

    uses transitions-top {
     augment "transitions/transition/actions/action/conditional/true" {
         uses next-state-top;
       }

     augment "transitions/transition/actions/action/conditional/false" {
```

```
         uses next-state-top;
       }
     augment "transitions/transition/actions/action/simple" {
         //uses next-state-top;
         leaf next-state {
description "it defines the next state";
           type leafref {
 description "it refers to its id";
             path "../../../../../../../../states/state/id";
           }
           description "Id of the next state";
         }
       }
    }

   }

   grouping states-top {
description "it defines the grouping states";
     leaf current-state {
description "it defines the current state";
       type leafref {
```

```
description "it refers to its id";
         path "../states/state/id";


       }

     }



     container states {
description "it defines the container states";

       list state {

   description "it defines the list of states";

         key "id";

         uses state-top;

       }

     }

   }




   // data definition statements




   uses states-top;


   // extension statements


   // feature statements
```

```
   // augment statements.

   // rpc statements

 }//module fsm
```

<CODE ENDS>

7.3. Example of values for the YANG model

FIELD NAME	YANG DATA TYPE	VALUE
Current State	leafref	"an existing state id in the FSM"
State		
id	uint32	1
name	string	Steady
description	string	"whatever string"
transition		
name	string	"whatever string"
type	enum	BER_CHANGE
description	string	"whatever string"
filter		
filter-id	uint32	2
filter-type	anyxml or xpath	BER>0.0009
action		
id	uint32	3
type	enum	SIMPLE
statement	string	"whatever string"
execute	anyxml	"this recalls an RPC where the FEC value is expressed"
next-operation	uint32	NULL
next-state	leafref	"an existing state id in the FSM"

8. Acknowledgements

This work has been partially supported by the European Commission through the H2020 ORCHESTRA (Optical peRformanCe monitoring enabling dynamic networks using a Holistic cross-layEr, Self-configurable Truly flexible approach, grant agreement no: H2020-645360) project. The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

9. Other Contributors

Matteo Dallaglio (Scuola Superiore Sant’Anna), Andrea Di Giglio (Telecom Italia), Giacomo Bernini (Nextworks).

10. Security Considerations

TBD

11. IANA Considerations

TBD

12. References

12.1. 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>.

[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/info/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/info/rfc6241>.

[RFC7491] King, D. and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations", RFC 7491, DOI 10.17487/RFC7491, March 2015, <https://www.rfc-editor.org/info/rfc7491>.

12.2. Informative References

[I-D.brockners-inband-oam-requirements] Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.fioccola-ippm-multipoint-alt-mark] Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate Marking method for passive and hybrid performance monitoring", draft-fioccola-ippm-multipoint-alt-mark-02 (work in progress), March 2018.

[I-D.ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-20 (work in progress), December 2017.

[I-D.ietf-ippm-alt-mark]
Fioccola, G., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate Marking method for passive and hybrid performance monitoring", draft-ietf-ippm-alt-mark-14 (work in progress), December 2017.

[I-D.song-opsawg-dnp4iq]
Song, H. and J. Gong, "Requirements for Interactive Query with Dynamic Network Probes", draft-song-opsawg-dnp4iq-01 (work in progress), June 2017.

[I-D.vergara-ccamp-flexigrid-yang]
Madrid, U., Perdices, D., Lopezalvarez, V., Dios, O., King, D., Lee, Y., and G. Galimberti, "YANG data model for Flexi-Grid Optical Networks", draft-vergara-ccamp-flexigrid-yang-06 (work in progress), January 2018.

[I-D.zhang-ccamp-l1-topo-yang]
zhenghaomian@huawei.com, z., Fan, Z., Sharma, A., and X. Liu, "A YANG Data Model for Optical Transport Network Topology", draft-zhang-ccamp-l1-topo-yang-07 (work in progress), April 2017.

Authors' Addresses

Nicola Sambo
Scuola Superiore Sant'Anna
Via Moruzzi 1
Pisa 56124
Italy

Email: nicola.sambo@sssup.it

Piero Castoldi
Scuola Superiore Sant'Anna
Via Moruzzi 1
Pisa 56124
Italy

Email: piero.castoldi@sssup.it

Giuseppe Fioccola
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: giuseppe.fioccola@telecomitalia.it

Filippo Cugini
CNIT
Via Moruzzi 1
Pisa 56124
Italy

Email: filippo.cugini@cnit.it

Haoyu Song
Huawei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: haoyu.song@huawei.com

Tianran Zhou
Huawei
156 Beiqing Road
Beijing 100095
China

Email: zhoutianran@huawei.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 22, 2019

N. Sambo
P. Castoldi
Scuola Superiore Sant'Anna
G. Fioccola
Huawei Technologies
F. Cugini
CNIT
H. Song
T. Zhou
Huawei

May 21, 2019

YANG model for finite state machine
draft-sambo-netmod-yang-fsm-05

Abstract

Network operators and service providers are facing the challenge of deploying systems from different vendors while looking for a trade-off among transmission performance, network device reuse, and capital expenditure without the need of being tied to single vendor equipment. The deployment and operation of more dynamic and programmable network infrastructures can be driven by adopting model-driven and software-defined control and management paradigms. In this context, YANG enables to compile a set of consistent vendor-neutral data models for networks and components based on actual operational needs emerging from heterogeneous use cases. This document proposes YANG models to describe events, operations, and finite state machine of YANG-defined network elements. The proposed models can be applied in several use cases: i) in the context of optical networks to pre-instruct data plane devices (e.g., an optical transponder) on the actions to be performed (e.g., code adaptation) in case some events, such as physical layer degradations, occur; ii) in general data networks, network telemetry applications can define and embed custom data probes into data plane devices. A probe in many cases can be modeled as an FSM; iii) the monitoring of packet loss and delay through a network clustering approach; iv) for re-routing in optical networks.

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 November 22, 2019.

Table of Contents

1. Introduction

Networks are evolving toward more programmability, flexibility, and multi-vendor interoperability. Multi-vendor interoperability can be applied in the context of nodes, i.e. a node composed of components provided by different vendors (named fully disaggregated white box) is assembled under the same control system. This way, operators can optimize costs and network performance without the need of being tied to single vendor equipment. NETCONF protocol RFC6241 [RFC6241] based on YANG data modeling language RFC6020 [RFC6020] is emerging as a candidate Software Defined Networking (SDN) enabled protocol. First, NETCONF supports both control and management functionalities, thus permits high programmability. Then, YANG enables data modeling in a vendor-neutral way. Some recent works have provided YANG models to describe attributes of links (e.g., identification), nodes (e.g., connectivity matrix), media channels, and transponders (e.g., supported forward error correction - FEC) of networks ([I-D.ietf-i2rs-yang-network-topo] [I-D.vergara-ccamp-flexigrid-yang] [I-D.zhang-ccamp-l1-topo-yang]), also including optical technologies. This document presents YANG models to describe events, operations, and finite state machine of YANG-defined network elements. Such models can be applied to several use cases. In the context of elastic optical networks (EONs), the model enables a centralized remote network controller (managed by a network operator) to instruct a transponder controller about the actions to perform when certain events (e.g., failures) occur. The actions to be taken and the events can be re-programmed on the device. In general data networks, programmable network telemetry is considered a killer SDN application which can help applications gain unprecedented visibility to network data plane. Instead of providing raw data, network devices can be configured to filter and process data directly on the data plane and only hand preprocessed data to the collector, in order to save data bandwidth and reduce reaction delay ([I-D.song-opsawg-dnp4iq]) . Such configurations can be programed as custom probes and dynamically deployed into data plane devices. A probe in many cases can be modeled as an FSM. Another use case is the monitoring of packet loss and delay through a network clustering approach: in this case, each FSM state is determined by a specific subdivision of the network in Clusters ([I-D.ietf-ippm-multipoint-alt-mark]). Finally, a use case is related to enable automatic local reconfiguration of a backup route in optical networks.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [RFC2119].

3. Terminology

ABNO: Application-Based Network Operations

BER: Bit Error Rate

EON: Elastic Optical Network

FEC: Forward Error Correction

FSM: Finite State Machine

NETCONF: Network Configuration Protocol

OAM: Operation Administration and Maintenance

SDN: Software Defined Network

YANG: Yet Another Network Generator

DNP: Dynamic Network Probe

AMM: Alternate Marking Method

4. Example of application

4.1. Pre-programming resiliency schemes in EONs

EONs (optical networks based on flexible grid supporting circuits of different bandwidth) are expected to employ flexible transponders, i.e. transponders supporting multiple bit rates, multiple modulation formats, and multiple codes. Such transponders permits the (re-) configuration of the bit rate value based on traffic requirements, as well as the configuration of the modulation format and code based on the physical characteristics of a path (e.g., quadrature phase shift keying is more robust than 16 quadrature amplitude modulation). This way, transmission parameters can be (re-) configured based on physical layer changes. The YANG model presented in this draft enables to pre-program reconfiguration settings of data plane devices in case of failures or physical layer degradations. In particular, soft failures are assumed. Soft failures imply transmission performance degradation, in turns a bit error rate (BER) increase,

e.g. due to the ageing of some network devices. Without loosing generality, the ABNO architecture is assumed for the control and management of EONs (RFC7491 [RFC7491]). Considering the state of the art, when pre-FEC BER passes above a predefined threshold, it is expected that an alarm is sent to the OAM Handler, which communicates with the ABNO controller that may trigger an SDN controller (that could be the Provisioning Manager of ABNO RFC7491 [RFC7491]) for computing new transmission parameters. The involved ABNO modules are shown in the simplified ABNO architecture of Fig. 1. Then, transponders are reconfigured. When alarms related to several connections impacted by the soft failure are generated, this procedure may be particularly time consuming. The related workflow for transponder reconfiguration is shown in Fig. 2. The proposed model enables an SDN controller to instruct the transponder about reconfiguration of new transmission parameters values if a soft failure occurs. This can be done before the failure occurs (e.g., during the connection instantiation phase or during the connection service), so that data plane devices can promptly reconfigure themselves without querying the SDN controller to trigger an on-demand recovery. This is expected to speed up the recovery process from soft failures. The related flow chart is shown in Fig. 3. The whole mechanism is based on a finite state machine where each state is associated to a specific configuration of transmission parameters (e.g., modulation format). The transition from a state to another state is triggered by specific events at the physical layer such as the bit error rate above a threshold. The transition from a state to another state implies a set of actions, including the change of transmission parameters (e.g., modulation format), which are actually suitable for the current condition at the physical layer. Moreover, since transmission and receiver must be synchronized about the transmission settings (modulation format and so no) for a proper transmission, another action consists of this synchronization. Thus, when the transponder at the receiver side decides to change its transmission parameters based on the monitored BER, the remote transponder at the transmitter side has to do the same state transition. In particular, the transponder at the receiver side sends a message to the transmitter to synchronize about the transmission parameters to be adopted. This message can be sent over a control channel. This way both the transmitter and receiver operates with the same transmission parameters: e.g. the format, FEC, and so on. No central controller is involved at this stage, only a notification can be sent to the central controller to inform it about the successful reconfiguration.

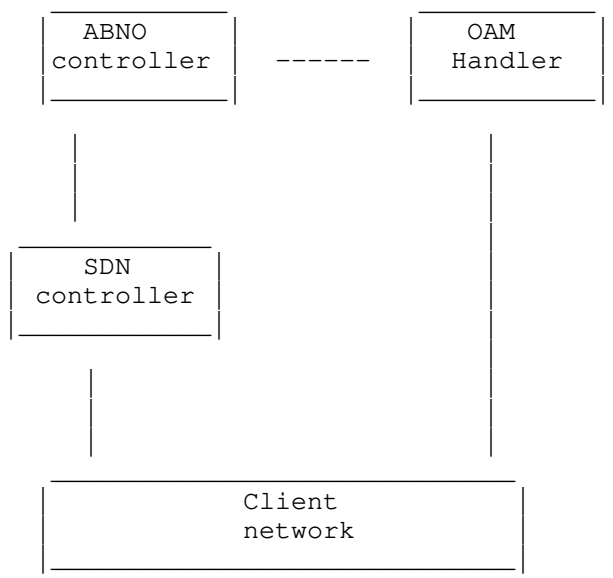

Figure 1: Assumed ABNO functional modules

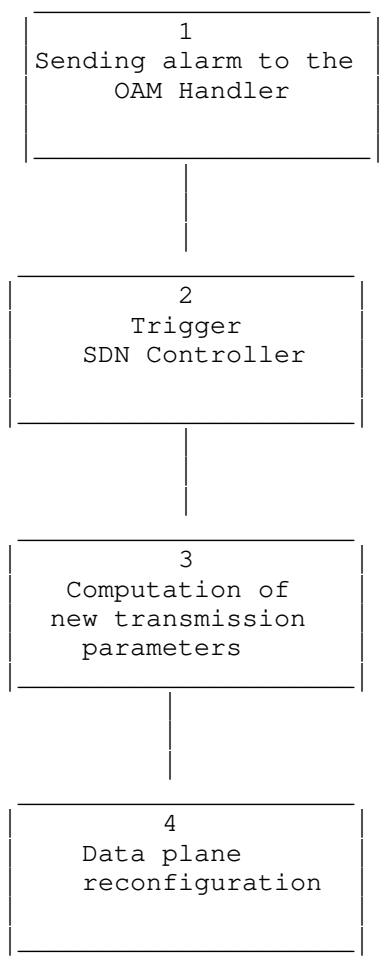

Figure 2: Flow chart of the expected state-of-the-art approach

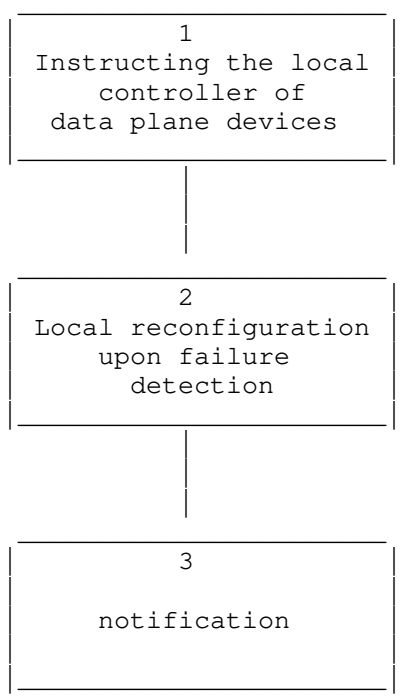

Figure 3: Flow chart of the approach exploiting YANG models in this draft

4.2. Deploying Dynamic Probes for Programmable Network Telemetry

In the past, network data analytics was considered a separate function from networks. They consume raw data extracted from networks through piecemeal protocols and interfaces. With the advent of user programmable data plane, we expect a paradigm shift that makes the data plane be an active component of the data telemetry and analytics solution. The programmable in-network data preprocessing is efficient and flexible to offload some light-weight data processing through dynamic data plane programming or configuration. A universal network data analytics platform built on top of this enables a tight and agile network control and OAM feedback loop. A proposed dynamic network telemetry system architecture is illustrated in Fig.4.

An application translates its data requirements into a set of Dynamic Network Probes (DNP) targeting a subset of data plane devices. After the probes are deployed, each probe conducts its corresponding in-network data preprocessing and feeds the preprocessed data to the

collector. The collector finishes the data post-processing and presents the results to the data-requesting application.

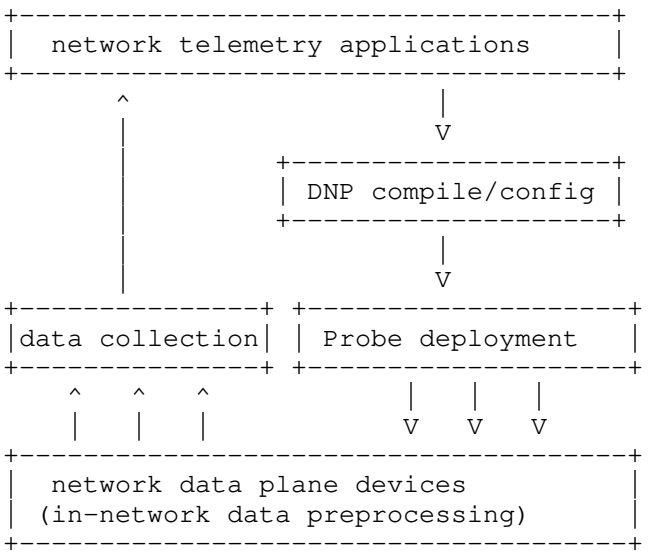

Figure 4: Deploy dynamic network probes using YANG FSM models

Many DNPs can be modeled as FSM which are configured to capture specific events. Here FSMs essentially preprocess the raw stream data and only report the necessary data to subscribing applications.

For example, a congestion control application needs to monitor the router buffer occupancy. Instead of polling the buffer depth periodically, it is only interested in the real-time events when the buffer depth crosses a low and a high threshold. We can install a probe to achieve this data plane function and the probe can be modeled as a three-state FSM. Each state represents a buffer region: below the low threshold, above the high threshold, and in between the two thresholds. A possible state transition is checked against the buffer depth for each incoming and outgoing packet. Whenever a state transition happens, an event is generated and reported to the application. This approach significantly reduces the amount of data sent to the application and also allows a timely event notification.

For another example, an application would like to monitor the delay experienced by a flow. The packet delay on its forwarding path can be acquired by using iOAM [I-D.brockners-inband-oam-requirements]. However, the application only needs to know that N consecutive flow packets experience a delay longer than T. Instead of forwarding the

raw delay data to the application, a probe can be deployed to detect the event. Similarly, the probe can be modeled as an FSM.

4.3. IP Performance Measurements on multipoint-to-multipoint large Networks

Networks offer rich sets of network performance measurement data, but traditional approaches run into limitations. One reason for this is the fact that in many cases, the bottleneck is the generation and export of the data and the amount of data that can be reasonably collected from the network runs into bandwidth and processing constraints in the network itself. In addition, management tasks related to determining and configuring which data to generate lead to significant deployment challenges.

In order to address these issues, an SDN controller application orchestrates network performance measurements tasks across the network to allow an optimized monitoring. In fact the IP Performance Measurement SDN Controller Application in Figure 5 can calibrate how deep can be obtained monitoring data from the network by configuring measurement points roughly or meticulously. This can be established by using the feedback mechanism reported in Figure 5.

For instance, the SDN Controller can configure initially an end to end monitoring between ingress points and egress points of the network. If the network does not experiment issues, this approximate monitoring is good enough and is very cheap in terms of network resources. But, in case of problems, the SDN Controller becomes aware of the issues from this approximate monitoring and, in order to localize the portion of the network that has issues, configures the measurement points more exhaustively. So a new detailed monitoring is performed. After the detection and resolution of the problem the initial approximate monitoring can be used again. This idea is general and can be applied to different performance measurements techniques both active and passive (and hybrid).

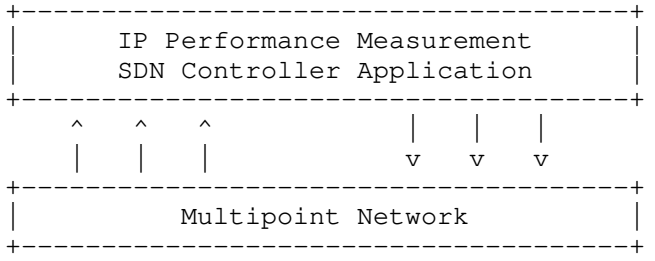

Figure 5: Feedback mechanism on multipoint-to-multipoint large Networks

One of the most efficient methodology to perform packet, loss delay and jitter measurements both in an IP and Overlay Networks is the Alternate Marking method, as presented in RFC8321 [RFC8321] and [I-D.ietf-ippm-multipoint-alt-mark].

This technique can be applied to point-to-point flows but also to multipoint.to-multipoint flows (see RFC8321 [RFC8321] and [I-D.ietf-ippm-multipoint-alt-mark]). The Alternate Marking method creates batches of packets by alternating the value of 1 or 2 bits of the packet header. These batches of packets are unambiguously recognized over the network and the comparison of packet counters permits the packet loss calculation. The same idea can be applied for delay measurement by selecting special packets with a marking bit dedicated for delay measurements. This method needs two counters each marking period for each flow under monitor. For this reason by considering n measurement points and n monitored flows, the order of magnitude of the packet counters for each time interval is n*n*2 (1 per color).

Multipoint Alternate Marking, described in [I-D.ietf-ippm-multipoint-alt-mark], aims to reduce this value and makes the performance monitoring more flexible in case a detailed analysis is not needed.

It is possible to monitor a Multipoint Network without examining in depth by using the Network Clustering (subnetworks that are portions of the entire network that preserve the same property of the entire network). So in case there is packet loss or the delay is too high the filtering criteria could be specified more in order to perform a per flow detailed analysis, as described in RFC8321 [RFC8321].

An application of the multipoint performance monitoring can be done by using FSM (each state is a composition of clusters) and feedback mechanism where the SDN Controller is the brain of the network and can manage flow control to the switches and routers and, in the same way, can calibrate the performance measurements depending on the necessity.

4.4. Re-routing in optical networks

FSM can be also applied for rerouting purposes as dynamic restoration in optical networks. In such a use case all the nodes along the backup route of a media channel should hold a FSM indicating the reconfigurations to be performed in case that media channel is rerouted along the backup path. Note that resources along the backup route are not reserved neither configured during a normal operation of the network as instead it happens for protection.

The proposed approach aims at achieving a sort of hybrid centralized/ distributed rerouting solution applicable to all the networks controlled with NETCONF. More specifically, the decision of the backup route is taken centrally by the SDN controller, but it is acted in a distributed way through FSM when a problem appears.

In particular, the transmitter, the receiver, and the ROADMs along the backup route have installed in their agent a FSM including a "re-routing" state, associated to a specific media channel. For the transmitter and receiver, this state is associated to the transmission parameters of the backup media channel: e.g., modulation format, central frequency, FEC, and so on. Regarding the ROADMs, the "re-routing" state is associated to the cross connections of the backup route.

When a link failure occurs (e.g., fiber cut), the receiver reveals a loss of light, which triggers the transition to the "re-routing" state. Thus, the receiver can set itself to the new transmission parameters. Moreover, similarly to the message sent over a control channel to the transmitter in Sec. 4.1, the receiver sends a message to the transmitter and the backup Reconfigurable Add-Drop Multiplexer (ROADMs). The task of this message is to simply trigger a state transition to the "re-routing" state so that each backup ROADM and the transmitter can apply the proper reconfigurations. This way, the SDN controller is not interrogated during the failure and the recovery can be speeded up. After reconfigurations, the SDN controller can be notified and the SDN controller can tear down the primary path. Backup routes can be reprogrammed based on the network states evolutions (e.g., link and spectrum occupancy).

5. YANG for finite state machine (FSM)

This model defines a list of states and transitions to describe a generic finite state machine (FSM). The related code and tree are shown in the Appendix.

```
 <current-state>: it defines the current state of the FSM.
 <states>: this element defines the FSM as follows.
           <state>: this list defines all the FSM states.
                  <id>: this leaf attribute of <state> defines the
                  identifier of the state
                  <name>: this leaf attribute of <state> defines the
                  name of the state
                  <description>: this leaf is a "string" describing the
                  state
                  <transitions>: this attribute defines a list of
                  transitions to other states in the FSM.
                          <name>: this attribute defines the name of a
```

transition
<type>: this attribute defines the type of the transition from a pool of possible transition types predefined inside the YANG model. Together with the <name> attribute, it uniquely identifies the transition.
<description>: this optional attribute is a "string" describing the transition
<filters>: this leaf is a list of input parameters related to the transition. This attribute enables to further express a transition: as an example, if a transition can be triggered by a parameter (e.g., a monitored performance parameter) exceeding a threshold (as in Sec. 5), an element of the list defines this threshold. Thus, if the parameter is outside the threshold, the transition is taken, otherwise not.
<filter>: this leaf of <filters> defines a filter parameter.
<filter-id>: this leaf of <filters> define the identifier number associated with the <filter> attribute.
<actions>: this attribute defines a list of actions to take during the transition.
<action>: this attribute is the list of actions
<id>: this leaf of <action> defines the identifier number of an action.
<type>: this leaf of <action> defines the type of an action.
<simple>: this leaf defines (differently from <conditional> detailed below) an action that has to be directly executed.
<execute>: this attribute recalls an RPC encapsulating the effective task (action) to be executed by the hardware. If more actions (e.g., "A" and "B"), defined in the <action> list, have to be executed, these actions can be executed sequentially according to the <next-action> attribute detailed below. Thus, by

referring to the tree of the Appendix, when an action ("A") is executed, the <next-action> attribute will bring to another action ("B"). If more actions have to be executed in parallel (e.g., "A" & "B"), not sequentially, an element of the <action> list should be defined to express an action (e.g., "A&B") consisting of more actions to be executed in parallel.
<next-action>: this attribute defines the identification number of a next action that has to be taken. The <next-action> can assume a NULL value.
<conditional>: this leaf enables a check ("true" or "false") to be verified before executing the action. Based on the check, the proper attributes <execute> and <next-operation> are considered.
<statement>: this leaf of <conditional> defines the condition to be verified before executing the action.
<true>: this leaf of <conditional> defines a result of the check associated to <statement>. Proper <execute> and <next-operation> attributes are associated with this result of the check.
<false>: this leaf of <conditional> defines a result of the check associated to <statement>. Proper <execute> and <next-operation>

attributes are associated with this result of the check.

<next-state>: this attribute defines the next state of FSM when an action is executed.

6. Implementation of the pre-programming resiliency schemes in EONs

These presented model can be used to enable a centralized network controller, managed by a network operator, to instruct data plane hardware on its reconfiguration if some events, such as a failure or physical layer degradation, occur. As an example, an optical signal impacted by a soft failure (i.e., a physical layer degradation inducing a pre forward error correction bit error rate increase - pre-FEC) can be maintained by adapting the FEC of the signal itself. This action to be taken and, more in general operations to be executed depending on critical events, can be (re-) programmed on the transponder by (re-) sending a NETCONF <edit-config> message to the device controller including a FSM defined by the YANG model. Such a system has the main goal to speed up the reaction of the network to certain events/faults and to alleviate the workload of the centralized controller. The speed up derives from the fact that the centralized controller is able to pre-compute and pre-configure on the network devices the actions to take when an event occurs taking into account a global view and knowledge of the network. In this way, the device is already aware of the actions to be locally applied to reconfigure a connection, avoiding to inform the controller and to wait for the response indicating what to do. Consequently, part of the workload is also removed from the centralized controller. When the reaction is successfully completed in the data plane, the centralized controller can be notified about the faults and the taken action. A flexible transponder supporting two FEC types, 7% and 20%, is considered. A two-states FSM is also assumed. The states have <name> attribute set to "Steady" and "Fec-Baud-Adapt", respectively. In the "Steady" state, the signal is in a healthy condition, adopting a 7% FEC, with a pre-FEC BER below an assigned threshold of 9 x 10-4. A transition from this state can be triggered by the event with <name>=BER_CHANGE and <filter-type>=9 x 10-4, thus expressing a change of the pre-FEC BER above the threshold. In case the pre-FEC BER exceeds 9 x 10-4 due to a soft failure, the state machine evolves to the "Fec-Baud-Adapt" state and an adaptation to a more robust FEC of 20% (executed by the attribute <execute>) is performed. The system can return to the "Steady" state if the pre-FEC BER goes below another pre-defined threshold and the FEC is reconfigured to 7%.

7. Appendix

This appendix reports the YANG models code and the related tree.

7.1. YANG model for FSM - Tree

```
module: ietf-fsm
   +--rw current-state?   leafref
   +--rw states
      +--rw state [id]
         +--rw id             state-id-type
         +--rw description?   string
         +--rw transitions
            +--rw transition [name type]
               +--rw name           string
               +--rw type           transition-type
               +--rw description?   string
               +--rw filters
               |  +--rw filter [filter-id]
               |     +--rw filter-id    uint32
               +--rw actions
                  +--rw action [id]
                     +--rw id             transition-id-type
                     +--rw type           enumeration
                     +--rw conditional
                     |  +--rw statement    string
                     |  +--rw true
                     |  |  +--rw execute
                     |  |  +--rw next-action?   transition-id-type
                     |  |  +--rw next-state?    leafref
                     |  +--rw false
                     |     +--rw execute
                     |     +--rw next-action?   transition-id-type
                     |     +--rw next-state?    leafref
                     +--rw simple
                        +--rw execute
                        +--rw next-action?   transition-id-type
                        +--rw next-state?    leafref
```

7.2. YANG model for FSM - Code

<CODE BEGINS> file "ietf-fsm@2016-03-15.yang"

```
 module ietf-fsm {

   namespace "http://sssup.it/fsm";
```

```
   prefix fsm;



   identity TRANSITION {
       description "Base for all types of event";
   }

   identity ON_CHANGE {
       base TRANSITION;
       description
         "The event when the database changes.";
   }

   // typedef statements

   typedef transition-type {
        description "it defines the type of transition (event)";
     type identityref {
       base TRANSITION;
     }
   }

   typedef transition-id-type {
        description "it defines the id of the transition (event)";
```

```
     type uint32;

   }




   // grouping statements

   grouping action-block {

        description "it defines the action to perform when a transition
        occurs";

     leaf id {

description "it refers to the id of the transition";
       type transition-id-type;

     }

     leaf type {

description "it defines if the action has to be simply executed or if
a conditional statement has to be checked before execution";

       type enumeration {

         enum "CONDITIONAL_OP" {
description "it defines the type CONDITIONAL OPERATION to check a
statement before execution";
}

         enum "SIMPLE_OP" {
description "it defines the type SIMPLE OPERATION: i.e., an operation
to be directly executed;
                }

       }

       mandatory true;

     }



     grouping execution-top {
```

```
        description "it defines the execution attribute";
       anyxml execute {
         description "Represent the action to perform";
       }
       leaf next-action {
         type transition-id-type;
         description "the id of the next action to execute";

       }
     }


     container conditional {
        description "it defines the container CONDITIONAL";
       when "../type = 'CONDITIONAL_OP'";
       leaf statement {
         type string;
         mandatory true;
         description
           "The statement to be evaluated before execution.
           E.g. if a=b";
       }
       container true {
description "it is referred to the result TRUE of a conditional
statement";
         uses execution-top;
```

```
       }
       container false {
description "it is referred to the result FALSE of a conditional
statement ";
         uses execution-top;
       }
     }

     container simple {
description "Simple execution of an action without checking any
condition";
       when "../type = 'SIMPLE_OP'";

       uses execution-top;
     }
   }

   grouping action-top {

description "it defines the grouping of action";
     list action {

description "it defines the list of actions";
       key "id";
       ordered-by user;
       uses action-block;
     }
```

```
   }

   grouping on-change {
     description
       "Event occuring when a modification of one or more
        objects occurs";

     container filters {
       description
         "This container contains a list of configurable filters
          that can be applied to subscriptions.  This facilitates
          the reuse of complex filters once defined.";
       list filter {
         key "filter-id";

         description
           "A list of configurable filters that can be applied to
            subscriptions.";
         leaf filter-id {
           type uint32;
           description
             "An identifier to differentiate between filters.";
         }
       }
     }
```

```
   }

   grouping transition-top {
description "it defines the grouping transition";
     leaf name {
description "it defines the transition name";
       type string;
       mandatory true;
     }

     leaf type {
description "it defines the transition type";
       type transition-type;
       mandatory true;
     }

     leaf description {
description "it describes the transition ";
       type string;
     }

     // list of all possible events
     uses on-change {
```

```
       when "type = 'ON_CHANGE'";
     }

     container actions {
description "it defines the container action";
       uses action-top;
     }
   }

   grouping transitions-top {
description "it defines the grouping transition";
     container transitions {
description "it defines the container transitions";
       list transition {
description "it defines the list of transitions";
         key "name type";
         uses transition-top;
       }
     }
   }

   // data definition statements
```

```
   uses transitions-top;

   // extension statements

   // feature statements

   // augment statements


   organization
     "Scuola Superiore Sant'Anna Network and Services Laboratory";

   contact
     " Editor: Matteo Dallaglio
               <mailto:m.dallaglio@sssup.it>
     ";

   description
     "This module contains a YANG definitions of a generic finite state
     machine.";

   revision 2016-03-15 {
     description "Initial Revision.";
     reference
       "RFC xxxx:";
```

```
   }

   // identity statements

   // typedef statements

   typedef state-id-type {

description "it defines the id type of the states";
     type uint32;
   }

   // grouping statements
   grouping state-top {

description "it defines the grouping state";
     leaf id {

description "it defines the id of a transition";
       type state-id-type;
     }

     leaf description {

description "it describes a transition";

       type string;
```

```
     }



     grouping next-state-top {

description "it defines the grouping for the next state";

       leaf next-state {

description "it defines the next state";

           type leafref {

description "it refers to its id";

             path "../../../../../../../../../states/state/id";

           }

           description "Id of the next state";

         }

     }


    uses transitions-top {

     augment "transitions/transition/actions/action/conditional/true" {

         uses next-state-top;

       }


     augment "transitions/transition/actions/action/conditional/false" {
```

```
         uses next-state-top;
       }
     augment "transitions/transition/actions/action/simple" {
         //uses next-state-top;
         leaf next-state {
description "it defines the next state";
           type leafref {
 description "it refers to its id";
             path "../../../../../../../../states/state/id";
           }
           description "Id of the next state";
         }
       }
    }
   }
   grouping states-top {
description "it defines the grouping states";
     leaf current-state {
description "it defines the current state";
       type leafref {
```

```
description "it refers to its id";
         path "../states/state/id";


       }

     }



     container states {
description "it defines the container states";

       list state {

   description "it defines the list of states";

         key "id";

         uses state-top;

       }

     }

   }




   // data definition statements




   uses states-top;


   // extension statements


   // feature statements
```

```
   // augment statements.

   // rpc statements

 }//module fsm
```

<CODE ENDS>

7.3. Example of values for the YANG model

FIELD NAME	YANG DATA TYPE	VALUE
Current State	leafref	"an existing state id in the FSM"
State		
id	uint32	1
name	string	Steady
description	string	"whatever string"
transition		
name	string	"whatever string"
type	enum	BER_CHANGE
description	string	"whatever string"
filter		
filter-id	uint32	2
filter-type	anyxml or xpath	BER>0.0009
action		
id	uint32	3
type	enum	SIMPLE
statement	string	"whatever string"
execute	anyxml	"this recalls an RPC where the FEC value is expressed"
next-operation	uint32	NULL
next-state	leafref	"an existing state id in the FSM"

8. Acknowledgements

This work has been partially supported by the European Commission through the H2020 ORCHESTRA (Optical peRformanCe monitoring enabling dynamic networks using a Holistic cross-layEr, Self-configurable Truly flexible approach, grant agreement no: H2020-645360) project. The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

9. Other Contributors

Matteo Dallaglio (Scuola Superiore Sant'Anna), Andrea Di Giglio (Telecom Italia), Giacomo Bernini (Nextworks).

10. Security Considerations

TBD

11. IANA Considerations

TBD

12. References

12.1. 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>.

[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/info/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/info/rfc6241>.

[RFC7491] King, D. and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations", RFC 7491, DOI 10.17487/RFC7491, March 2015, <https://www.rfc-editor.org/info/rfc7491>.

12.2. Informative References

[I-D.brockners-inband-oam-requirements] Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., Lapukhov, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.ietf-i2rs-yang-network-topo] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-20 (work in progress), December 2017.

[I-D.ietf-ippm-multipoint-alt-mark]
Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate Marking method for passive and hybrid performance monitoring", draft-ietf-ippm-multipoint-alt-mark-01 (work in progress), March 2019.

[I-D.song-opsawg-dnp4iq]
Song, H. and J. Gong, "Requirements for Interactive Query with Dynamic Network Probes", draft-song-opsawg-dnp4iq-01 (work in progress), June 2017.

[I-D.vergara-ccamp-flexigrid-yang]
Madrid, U., Perdices, D., Lopezalvarez, V., Dios, O., King, D., Lee, Y., and G. Galimberti, "YANG data model for Flexi-Grid Optical Networks", draft-vergara-ccamp-flexigrid-yang-06 (work in progress), January 2018.

[I-D.zhang-ccamp-l1-topo-yang]
zhenghaomian@huawei.com, z., Fan, Z., Sharma, A., and X. Liu, "A YANG Data Model for Optical Transport Network Topology", draft-zhang-ccamp-l1-topo-yang-07 (work in progress), April 2017.

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

Authors' Addresses

Nicola Sambo
Scuola Superiore Sant'Anna
Via Moruzzi 1
Pisa 56124
Italy

Email: nicola.sambo@sssup.it

Piero Castoldi
Scuola Superiore Sant'Anna
Via Moruzzi 1
Pisa 56124
Italy

Email: piero.castoldi@sssup.it

Giuseppe Fioccola
Huawei Technologies
Riesstrasse, 25
Munich 80992
Germany

Email: giuseppe.fioccola@huawei.com

Filippo Cugini
CNIT
Via Moruzzi 1
Pisa 56124
Italy

Email: filippo.cugini@cnit.it

Haoyu Song
Huawei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: haoyu.song@huawei.com

Tianran Zhou
Huawei
156 Beiqing Road
Beijing 100095
China

Email: zhoutianran@huawei.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2018

M. Wang
Q. Wu
Huawei

February 12, 2018

A YANG Data Model for module revision management
draft-wang-netmod-module-revision-management-00

Abstract

This document defines a YANG Data Model for module revision management. It is intended this model be used by vendors who support multiple revisions of the same YANG module in their systems but implement one revision of a module. In addition, this document introduces a new generic mechanism based on RPC, denoted as module-revision-change, that allow to report discrepancy information of a YANG module with two or multiple revisions that is defined in design time., e.g.,identifies a place in the node hierarchy where data node gets changed or new data gets inserted and indicate whether the change to the data node is backward compatible.

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 August 16, 2018.

Copyright Notice

Table of Contents

1. Introduction

The revised Network Management Datastore Architecture (NMDA) defines a set of new datastores that each hold YANG-defined data [RFC7950] and represent a different "viewpoint" on the data that is maintained by a server. To support the NMDA, many modules may need to be updated or restructured especially for some published non-NMDA modules, the model should be republished with an NMDA-compatible structure, deprecating non-NMDA constructs [I-D.ietf-netmod-rfc6087bis]. Therefore, how to support hackward-compatible and indicate the module's changes is an issue.

As described in RFC7950, a module name MUST NOT be changed when definitions contained in a module are available to be imported by any other module and are referenced in "import" statements via the module name. In some case, when we make non-backward compatible updates, the module name might be forced to change.

In order to provide an easy way to indicate how backward-compatible a given YANG module actually is, This document defines a YANG Data Model for module revision management. It is intended this model be used by vendors who support multiple revisions of the same YANG

module in their systems but implement one revision of a module. In addition, this document introduces a new generic mechanism based on RPC, denoted as module-revision-change, that allow to report discrepancy information of a YANG module with two or multiple revisions that is defined in design time., e.g.,identifies a place in the node hierarchy where data node gets changed or new data gets inserted and indicate whether the change to the data node is backward compatible.

2. Terminologies

The keywords "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].

The following terms are defined in [RFC6241] and are not redefined here:

- client
- notification
- server

The following terms are defined in [RFC7950] and are not redefined here:

action

container

list

operation

3. Design of YANG data model for module revision management

The "ietf-module-revision-management" module provides per revision the module discrepancy information used by a server. This module is defined using YANG version 1.1, but it supports the description of YANG modules written in any revision of YANG.

All data nodes in "ietf-module-revision-management" are "config false", and thus accessible in the operational state datastore.

Following is the YANG Tree Diagram for the "ietf-module-revision-management" module:

```
module: ietf-module-revision
    +--ro yang-modules
       +--ro module* [name revision]
          +--ro name                    yang:yang-identifier
          +--ro revision                yanglib:revision-identifier
          +--ro backward-compatible?    boolean
          +--ro module-change-log* [index]
             +--ro index                uint32
             +--ro change-operation     identityref
             +--ro (update-blocks)?
                +--:(data-definition-statement)
                |  +--ro data-definition
                |     +--ro target-node        yang:xpath1.0
                |     +--ro location-point?    yang:xpath1.0
                |     +--ro where?             enumeration
                |     +--ro data-definition?
                +--:(other-statement)
                   +--ro other-statement
                      +--ro statement-name?         identityref
                      +--ro statement-definition?
                      +--ro substatements* [statement-name]
                         +--ro statement-name             identityref
                         +--ro substatement-definition?
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module:
    +--ro backward-compatible?   boolean
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module/yanglib:submod
ule:
    +--ro backward-compatible?   boolean

  rpcs:
    +---x module-revision-change
       +---w input
       |  +---w module-name        yang:yang-identifier
       |  +---w source-revision?   yanglib:revision-identifier
       |  +---w target-revision?   yanglib:revision-identifier
       +--ro output
          +--ro data-nodes* [data-node-name]
             +--ro data-node-name      string
             +--ro is-new-node?        boolean
             +--ro change-operation?   identityref
```

3.1. yang-modules

This container holds all per revision the module discrepancy information used by a server.

3.1.1. yang-modules/module

This mandatory list contains one entry for each revision of each YANG module that is used by the server. It is possible for multiple revisions of the same module to be imported, in addition to an entry for the revision that is implemented by the server. Multiple revisions of the same module are either backward-compatible or non backward-compatible.

3.1.2. yang-modules/change-log

This list contains one entry for each schema node change from previous revision known by the server, and identifies schema node change path,location, operation and associated with corresponding schema node in the "change-log" list. Each revision of the YANG module has multiple entries.

A change log is an ordered collection of changes that are applied to one revision of YANG module. Each change is identified by an "index", and it has an change operation ("create", "delete","move", "modify") that is applied to the target resource. Each change can be applied to a sub-resource "target" within the target resource. If the operation is "move", then the "where" parameter indicates how the node is moved. For values "before" and "after", the "point" parameter specifies the data node insertion point.

Each entry within a change log MUST identify exactly one data definition change or other statement change.

3.2. RPC definition for module revision change

The "module-revision-change" rpc statement is defined to retrieve the schema data node changes between any two revisions of the same module, i.e. the data node that get updated or newly added during module revision change. This rpc statement takes module identification information as input, and provides the list of data nodes that make changes or are newly added in the later revision.

3.2.1. Usage Example

For example, there are two revisions of the same module, the yang codes are shown as below:

```
module example-a{
  yang-version 1.1;
  namespace "urn:example:a";
  prefix "a";
```

```
          organization "foo.";
          contact "fo@example.com";
          description
            "foo.";

          revision 2017-12-01 {
            description "Initial revision.";
          }

          container system {
            leaf host-name {
              type string;
              description
                "Hostname for this system.";
            }
          }
         }

        module example-a{
          yang-version 1.1;
          namespace "urn:example:a";
          prefix "a";

          organization "foo.";
          contact "fo@example.com";
          description
            "foo.";

          revision 2017-12-20{
            description "Initial revision.";
          }

          container system {
            leaf host-name {
              type string;
              description
                "Hostname for this system.";
            }
            leaf b {
              type string;
              description
                "foo";
           }
          }
```

If we initiate a "module-revision-change" RPC to retrieve the changes between two revisions of module "a", the NETCONF XML example are shown as below:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <module-revision-change xmlns="http://example.com/system">
    <module-name>example-a</module-namee>
    <source-revisions>1.0.0</source-revisions>
    <target-revisions>2.0.0</target-revisions>
  </module-revision-change>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data-nodes>
   <id>0001</id>
   <data-node-name>b</data-node-name>
   <is-new-node>true</is-new-node>
   <data-node-change>major-change</data-node-change>
  </data-nodes>
</rpc-reply>
```

4. Library Augmentation

Backward compatibility for each revision of YANG module can also be read using the yang library [I-D.ietf-netconf-rfc7895bis] if a server supports both YANG library and the augmentation defined below. If a server supports indication of backward compatibility forone revision of and the YANG module, it SHOULD also support the "ietf-module-revision" module.

The tree associated with the defined augmentation is:

```
module: ietf-module-revisions
  augment /yanglib:yang-library/yanglib:modules/yanglib:module:
+--ro backward-compatible?   bool

augment /yanglib:yanglibrary/yanglib:modules/yanglib:module
/yanglib:submodule:
    +--ro backward-compatible?   bool
```

5. Multiple revisions module management

As experience is gained with a module, it may be desirable to support multiple revisions of that module in their systems but implement one revision of a module at each time. To indicate the details changes of that module, e.g., identifies schema node change path,location, operation and associated with corresponding schema node, it will be desirable to use ’ietf-module-revision’ defined in this document to manage all the revisions of that module and keep track of module

change discrepancy in different revision, especially when the new revision is not backward compatible with previous revision.

6. Yang Data Model Definition

<CODE BEGINS> file "ietf-module-revision@2018-02-11.yang"

```
module ietf-module-revision {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-module-revision";
  prefix ml;

  import ietf-yang-library {
    prefix yanglib;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF Network Modeling (NETMOD) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>

     WG List:  <mailto:netmod@ietf.org>

     Author:   Qin Wu
               <mailto:bill.wu@huawei.com>
               Zitao Wang
               <mailto:wangzitao@huawei.com>";
  description
    "This YANG module defines an module log.";

  revision 2018-02-11 {
    description
      "Initial revision.";
    reference "RFC XXXX: Using Metadata with YANG for Module revisions";
  }

  identity operation-type {
    description
      "Abstract base identity for the operation type ";
  }

  identity create {
    base operation-type;
    description
      "Denotes create new data nodes";
```

```
  }

  identity delete {
    base operation-type;
    description
      "Denotes delete the target node";
  }

  identity move {
    base operation-type;
    description
      "Denote move the target node.";
  }

  identity modify {
    base operation-type;
    description
      "Denote modify the target data node.";
  }

  identity statement-type {
    description
      "Base identity for statement type";
  }

  identity feature-statement {
    base statement-type;
    description
      "feature statement, if this type be chose, it means that the
       feature or if-feature statement been modified";
  }

  identity identity-statement {
    base statement-type;
    description
      "identity statement, if this type be chose, it means that the
       identity statement been modified, for example, add new identity, etc.";
  }

  identity grouping-statement {
    base statement-type;
    description
      "grouping statement, if this type be chose, it means that the grouping
       statement been modified.";
  }

  identity typedef-statement {
    base statement-type;
```

```
    description
      "typedef statement, if this type be chose, it means that the typedef
       statement been modified.";
  }

  identity augment-statement {
    base statement-type;
    description
      "augment statement, if this type be chose, it means that the augment
       statement been modified.";
  }

  identity rpc-statement {
    base statement-type;
    description
      "rpc statement, if this type be chose, it means that the rpc
       statement been modified.";
  }

  identity notification-statement {
    base statement-type;
    description
      "notification statement, if this type be chose, it means that the notifica
tion
       statement been modified.";
  }

  grouping data-definition {
    container data-definition {
      leaf target-node {
        type yang:xpath1.0;
        mandatory true;
        description
          "Identifies the target data node for update.
           Notice that, if the update-type equal to move or delete,
           this target-node must point to the data node of old version.
           \t
           For example, suppose the target node is a YANG leaf named a,
           and the previous version is:
           \t
           container foo {
            leaf a { type string; }
            leaf b { type int32; }
           }
           \t
           the new version is:
            container foo {
           leaf b {type int32;}
            }
```

```
          \t
          Therefore, the targe-node should be /foo/a.";
      }
      leaf location-point {
        type yang:xpath1.0;
        description
          "Identifies the location point where the updates happened.";
      }
      leaf where {
        when "derived-from-or-self(../../change-operation, 'move')" {
          description
            "This leaf only applies for 'move'
             updates.";
        }
        type enumeration {
          enum "before" {
            description
              "Insert or move a data node before the data resource
               identified by the 'point' parameter.";
          }
          enum "after" {
            description
              "Insert or move a data node after the data resource
               identified by the 'point' parameter.";
          }
          enum "first" {
            description
              "Insert or move a data node so it becomes ordered
                  as the first entry.";
          }
          enum "last" {
            description
              "Insert or move a data node so it becomes ordered
               as the last entry.";
          }
        }
        default "last";
        description
          "Identifies where a data resource will be inserted
              or moved.";
      }
      anydata data-definition {
        when "derived-from-or-self(../../change-operation, 'modify')" {
          description
            "This nodes only be present when
                the 'change-operation' equal to 'modify'.";
        }
        description
```

```
          "This nodes used for present the definitions before updated.
           And this nodes only be present when
           the 'change-operation' equal to 'modify'.";
      }
      description
        "Container for data statement";
    }
    description
      "Grouping for data definition";
  }

  grouping other-statement {
    container other-statement {
      leaf statement-name {
        type identityref {
          base statement-type;
        }
        description
          "Statement name, for example, identity, feature, typedef, etc.";
      }
      anydata statement-definition {
        description
          "This nodes used for present new the definitions.";
      }
      list substatements {
        key "statement-name";
        leaf statement-name {
          type identityref {
            base statement-type;
          }
          description
            "Statement name, for example, identity, feature, typedef, etc.";
        }
        anydata substatement-definition {
          description
            "This nodes used for present new the definitions.";
        }
        description
          "List for substatements updates";
      }
      description
        "Container for header statement updates";
    }
    description
      "Grouping for header statement";
  }

  grouping change-log {
```

```
    list module-change-log {
      key "index";
      leaf index {
        type uint32;
        description
          "Index for module change log";
      }
      leaf change-operation {
        type identityref {
          base operation-type;
        }
        mandatory true;
        description
          "This leaf indicate the change operation, such as create, move, delete
, modify, etc.";
      }
      choice blocks-update {
        description
          "Choice for update block, this nodes can explicit present the update b
locks";
        case data-definition-statement {
          uses data-definition;
        }
        case other-statement {
          uses other-statement;
        }
      }
      description
        "List for module change log";
    }
    description
      "Grouping for module updated log";
  }

  container yang-modules {
    config false;
    list module {
      key "name revision";
      leaf name {
        type yang:yang-identifier;
        description
          "The YANG module or submodule name.";
      }
      leaf revision {
        type yanglib:revision-identifier;
        description
          "The YANG module or submodule revision date.  If no revision
           statement is present in the YANG module or submodule, this
           leaf is not instantiated.";
      }
```

```
      leaf backward-compatible {
        type boolean;
        description
          "Indicates whether it is a backward compatible version.
           If this parameter is set to true, it means that this version is
           a backwards compatible version";
      }
      uses change-log;
      description
        "List for module updated log";
    }
    description
      "This container present the modules updated log.";
  }
  augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
    description
      "Augment the yang library with backward compatibility indication.";
    leaf backward-compatible {
      type boolean;
      description
        "backward compatibility indication.";
    }
  }
  augment "/yanglib:yang-library/yanglib:module-set/yanglib:module/yanglib:submo
dule" {
    description
      "Augment the yang library with backward compatibility indication.";
    leaf backward-compatible {
      type boolean;
      description
        "backward compatibility indication.";
    }
  }
  rpc module-revision-change {
    description
      "Module Node change query operation.";
    input {
      leaf module-name {
        type yang:yang-identifier;
        mandatory true;
        description
          "The YANG module or submodule name.";
      }
      leaf source-revision {
        type yanglib:revision-identifier;
        description
          "The Source YANG module revision date.  If no revision
           statement is present in the YANG module or submodule, this
           leaf is not instantiated.";
```

```
      }
      leaf target-revision {
        type yanglib:revision-identifier;
        description
          "The target YANG module revision date.  If no revision
           statement is present in the YANG module or submodule, this
           leaf is not instantiated.";
      }
    }
    output {
      list data-nodes {
        key "data-node-name";
        description
          "Each entry represents a data node of a given module that
           have been changed from source revision of
           a module to target revision of the module.";
        leaf data-node-name {
          type string;
          description
            "a data node name of a given module that
             has been changed.";
        }
        leaf is-new-node {
          type boolean;
          description
            "indicate the data node is newly introduced node in the target revis
ion.";
        }
        leaf change-operation {
          type identityref {
            base operation-type;
          }
          description
          "This leaf indicate the change operation,
           such as create, move, delete, modify, etc.";
        }
      }
    }
  }
}
```

<CODE ENDS>

7. Security Considerations

This document defines a mechanism that is put at a place in the node hierarchy where data node gets changed or new data gets inserted and indicate whether the change to the data node is backward compatible and as such doesn’t introduce new security issues.

8. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-module-revision

Registrant Contact: The NETMOD WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

Name:	ietf-module-revision
Namespace:	urn:ietf:params:xml:ns:yang:ietf-module-revision
Prefix:	md
Reference:	RFC xxxx

---

9. Acknowledgements

This work is motivated from the discussions of module version in IETF 100 Singapore meeting. Thanks to Juergen Schoenwaelder and Adrian Farrel for useful comments on this work.

10. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-04 (work in progress), January 2018.

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-15 (work in progress), December 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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/info/rfc6020>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[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>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

Appendix A. Example of Module Revision Management

This section provides an example to generate XML snippet of ietf-module-revision based on the changes between the new revision of interface model as specified in [draft-ietf-netmod-rfc7223bis] and the old revision of interface model as specified in [RFC7223].

```
<yang-modules>
 <module>
  <name>ietf-interfaces</name>
  <revision>2018-01-09</revision>
  <backwards-compatible>false</backwards-compatible>
  <module-change-log>
   <index>0001</index>
   <update-type>delete</update-type>
   <update-blocks>
    <data-definition-statement>
     <data-definition>
      <target>/if:interfaces-state</target>
     </data-definition>
    </data-definition-statement>
   </update-blocks>
  </module-change-log>

  <module-change-log>
   <index>0002</index>
   <update-type>create</update-type>
   <update-blocks>
    <other-statement>
     <statement>
      <statement-name>feature-statement</statement-name>
```

```
            <statement-definition>
             feature if-mib {
              description
              "This feature indicates that the device implements
               the IF-MIB.";
              reference
              "RFC 2863: The Interfaces Group MIB";
             }
            </statement-definition>
           </statement>
          </other-statement>
         </update-blocks>
        </module-change-log>

        <module-change-log>
         <index>0003</index>
         <update-type>modify</update-type>
         <update-blocks>
          <data-definition-statement>
           <data-definition>
            <target>
            /if:interfaces/if:interface/if:link-up-down-trap-enable
            </target>
            <data-node>
             leaf link-up-down-trap-enable {
             if-feature if-mib; // add if-feature statement
              type enumeration {
              ....
            </data-node>
           </data-definition>
          </data-definition-statement>
         </update-blocks>
        </module-change-log>

        <module-change-log>
         <index>0004</index>
         <update-type>move</update-type>
         <update-blocks>
          <data-definition-statement>
           <data-definition>
            <target>
            /if:interfaces-state/if:interface/if:admin-status
            </target>
           <location-point>
            /if:interfaces/if:interface/link-up-down-trap-enable
           </location-point>
           <where>after</where>
          </data-definition-statement>
```

```
         </update-blocks>
        </module-change-log>

        <module-change-log>
         <index>0005</index>
         <update-type>move</update-type>
         <update-blocks>
          <data-definition-statement>
           <data-definition>
            <target>
            /if:interfaces-state/if:interface/if:statistics
            </target>
            <location-point>
            /if:interfaces/if:interface/if:speed
            </location-point>
            <where>after</where>
          </data-definition-statement>
         </update-blocks>
        </module-change-log>
        .......
       </module>
      </yang-modules>
```

Authors' Addresses

Michael Wang
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 22, 2019

M. Wang
Q. Wu
Huawei
A. Wang
China Telecom

September 18, 2018

A YANG Data Model for module revision management
draft-wang-netmod-module-revision-management-01

Abstract

This document defines a YANG Data Model for module revision change management. It is intended this model be used by vendors who support multiple revisions of the same YANG module in their systems but implement only one revision of a module. In addition, this document introduces a new generic mechanism based on RPC, denoted as module-revision-change, that allow datanode backwards compatibility detection and provide a report on change type and change details of a YANG module with two or multiple revisions that is defined in design time., e.g.,identifies a place in the node hierarchy where data node gets changed or new data gets inserted and indicate whether the change to the data node is backward compatible.

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 March 22, 2019.

Table of Contents

1. Introduction

The revised Network Management Datastore Architecture (NMDA) defines a set of new datastores that each hold YANG-defined data [RFC7950] and represent a different "viewpoint" on the data that is maintained by a server. To support the NMDA, many modules may need to be updated or restructured especially for some published non-NMDA modules, the model should be republished with an NMDA-compatible structure, deprecating non-NMDA constructs [I-D.ietf-netmod-rfc6087bis]. Therefore, how to support hackward-compatible and indicate the module's changes is an issue.

As described in [RFC7950] , a module name MUST NOT be changed when definitions contained in a module are available to be imported by any other module and are referenced in "import" statements via the module

name. In some case, when we make non-backward compatible updates, the module name might be forced to change, according to[RFC7950] module update rule.

In order to provide an easy way to indicate how backward-compatible a given YANG module actually is, this document defines a YANG Data Model for module revision change management. It is intended this model be used by vendors who support multiple revisions of the same YANG module in their systems but implement only one revision of a module. In addition, this document introduces a new generic mechanism based on RPC, denoted as module-revision-change, that allow Datanode backwards compatibility detection and provide a report on change type and change details of a YANG module with two or multiple revisions that is defined in design time., e.g.,identifies a place in the node hierarchy where data node gets changed or new data gets inserted and indicate whether the change to the data node is backward compatible.

In addition, in order to identify whether changes between two revisions represent bug fixes, new functionality, or both, etc [I-D.verdt-netmod-yang-versioning-reqs], this document also defines a new YANG extension statement which can help the user to understand the purpose of changing a specific node. See More details in section 4.

2. Terminologies

The keywords "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].

The following terms are defined in [RFC6241] and are not redefined here:

- client
- notification
- server

The following terms are defined in [RFC7950] and are not redefined here:

action

container

list

operation

3. Design of YANG data model for module revision management

The "ietf-module-revision-management" module provides per revision the module change type and change details used by a server. This module is defined using YANG version 1.1, but it supports the description of YANG modules written in any revision of YANG.

All data nodes in "ietf-module-revision-management" are "config false", and thus accessible in the operational state datastore.

Following is the YANG Tree Diagram for the "ietf-module-revision-management" module:

```
module: ietf-module-revision
    +--ro yang-modules
       +--ro module* [name revision]
          +--ro name                   yang:yang-identifier
          +--ro revision               yanglib:revision-identifier
          +--ro backward-compatible?   boolean
          +--ro revision-change-log* [index]
             +--ro index               uint32
             +--ro change-operation    identityref
             +--ro (yang-statements)?
                +--:(data-definition-statement)
                |  +--ro data-definition
                |     +--ro target-node        yang:xpath1.0
                |     +--ro location-point?    yang:xpath1.0
                |     +--ro where?             enumeration
                |     +--ro data-definition?
                +--:(other-statement)
                   +--ro other-statement
                      +--ro statement-name?         identityref
                      +--ro statement-definition?
                      +--ro substatements* [statement-name]
                         +--ro statement-name             identityref
                         +--ro substatement-definition?
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module:
    +--ro backward-compatible?   boolean
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module/yanglib:submod
ule:
    +--ro backward-compatible?   boolean

  rpcs:
    +---x module-revision-change
       +---w input
       |  +---w source module-name yang:yang-identifier
       |  +---w source-revision?   yanglib:revision-identifier
       |  +---w target module-name yang:yang-identifier
       |  +---w target-revision?   yanglib:revision-identifier
       +--ro output
          +--ro (status-response)?
          |  +--:(wrong-match)
          |  |  +--ro wrong-match?    empty
          |  +--ro data-nodes* [data-node-name]
          |     +--ro data-node-name      string
          |     +--ro is-new-node?        boolean
          |     +--ro change-operation?   identityref
```

3.1. yang-modules

This container holds all per revision the module discrepancy information used by a server.

3.1.1. yang-modules/module

This mandatory list contains one entry for each revision of each YANG module that is used by the server. It is possible for multiple revisions of the same module to be imported, in addition to an entry for the revision that is implemented by the server. Multiple revisions of the same module are either backward-compatible or non backward-compatible.

3.1.2. yang-modules/change-log

This list contains one entry for each schema node change from previous revision known by the server, and identifies schema node change path,location, operation and associated with corresponding schema node in the "change-log" list. Each revision of the YANG module has multiple entries.

A change log is an ordered collection of changes that are applied to one revision of YANG module. Each change is identified by an "index", and it has an change operation ("create", "delete","move", "modify") that is applied to the target resource. Each change can be applied to a sub-resource "target" within the target resource. If the operation is "move", then the "where" parameter indicates how the node is moved. For values "before" and "after", the "point" parameter specifies the data node insertion point.

Each entry within a change log MUST identify exactly one data definition change or other statement change.

3.2. RPC definition for module revision change

The "module-revision-change" rpc statement is defined to retrieve the schema data node changes between any two revisions of the same module, i.e. the data node that get updated or newly added during module revision change. This rpc statement takes module identification information as input, and provides the list of data nodes that make changes or are newly added in the later revision.

3.2.1. Usage Example

For example, there are two revisions of the same module, the yang codes are shown as below:

```
module example-a{
  yang-version 1.1;
  namespace "urn:example:a";
  prefix "a";

  organization "foo.";
  contact "fo@example.com";
  description
    "foo.";

  revision 2017-12-01 {
    description "Initial revision.";
  }

  container system {
    leaf host-name {
      type string;
      description
        "Hostname for this system.";
    }
  }
 }
```

```
module example-a{
  yang-version 1.1;
  namespace "urn:example:a";
  prefix "a";

  organization "foo.";
  contact "fo@example.com";
  description
    "foo.";

  revision 2017-12-20{
    description "Initial revision.";
  }

  container system {
    leaf host-name {
      type string;
      description
        "Hostname for this system.";
    }
    leaf b {
      type string;
      description
        "foo";
   }
```

```
        }
```

If we initiate a "module-revision-change" RPC to retrieve the changes between two revisions of module "a", the NETCONF XML example are shown as below:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <module-revision-change xmlns="http://example.com/system">
    <module-name>example-a</module-namee>
    <source-revisions>1.0.0</source-revisions>
    <target-revisions>2.0.0</target-revisions>
  </module-revision-change>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data-nodes>
   <id>0001</id>
   <data-node-name>b</data-node-name>
   <is-new-node>true</is-new-node>
   <data-node-change>major-change</data-node-change>
  </data-nodes>
</rpc-reply>
```

4. YANG Extension for Purpose Marking

In order to identify whether changes between two revisions represent bug fixes, new functionality, or both, etc [I-D.verdt-netmod-yang-versioning-reqs], purpose marking are defined via the YANG extension "mr:purpose". This YANG extension statement is defined in the module "ietf-module-revision" (Section 7). This YANG extension can help the user to understand the purpose of changing a specific node.

4.1. Usage Example: Add New Function

For example, there is a YANG data model "example-a", the yang codes are shown as below:

```
module example-a{
  yang-version 1.1;
  namespace "urn:example:a";
  prefix "a";

  organization "foo.";
  contact "fo@example.com";
  description
    "foo.";

  revision 2017-12-01 {
    description "Initial revision.";
  }

  container system {
    leaf host-name {
      type string;
      description
        "Hostname for this system.";
    }
  }
 }
```

When the author of "example-a" designs a new revision to add some new attributes, the "mr:purpose" marking can be used to mark the purpose of the updates. For example:

```
module example-a{
  yang-version 1.1;
  namespace "urn:example:a";
  prefix "a";

  organization "foo.";
  contact "fo@example.com";
  description
    "foo.";

  revision 2017-12-20{
    description "Initial revision.";
  }

  container system {
    leaf host-name {
      type string;
      description
        "Hostname for this system.";
    }
    leaf b {
      type string;
      mr:purpose "new-function";
      description
        "foo";

   }
  }
```

4.2. Usage example: Bug Fix

For example, there are some bugs in a published model "example-b", the yang codes are shown as below:

```
module example-b{
  yang-version 1.1;
  namespace "urn:example:b";
  prefix "b";

  organization "foo.";
  contact "fo@example.com";
  description
    "foo.";

  revision 2017-12-01 {
    description "Initial revision.";
  }

  container system {
    leaf host-name {
      type uint32;
      description
        "Hostname for this system.";
    }
  }
 }
```

The following updates allow to fix bugs in a backward-compatible way:

```
module example-b{
  yang-version 1.1;
  namespace "urn:example:b";
  prefix "b";

  organization "foo.";
  contact "fo@example.com";
  description
    "foo.";

  revision 2017-12-01 {
    description "Initial revision.";
  }

  container system {
    leaf host-name-update {
     type string;
     mr:purpose "bug-fix";
     description
     "Hostname for this system.";
    }
    leaf host-name {
      type uint32;
      status deprecated;
      description
        "Hostname for this system.";
    }
  }
 }
```

5. Library Augmentation

Backward compatibility for each revision of YANG module can also be read using the yang library [I-D.ietf-netconf-rfc7895bis] if a server supports both YANG library and the augmentation defined below. If a server supports indication of backward compatibility forone revision of and the YANG module, it SHOULD also support the "ietf-module-revision" module.

The tree associated with the defined augmentation is:

```
   module: ietf-module-revisions
     augment /yanglib:yang-library/yanglib:modules/yanglib:module:
   +--ro backward-compatible?   bool

   augment /yanglib:yanglibrary/yanglib:modules/yanglib:module
   /yanglib:submodule:
       +--ro backward-compatible?   bool
```

6. Multiple revisions module management

As experience is gained with a module, it may be desirable to support multiple revisions of that module in their systems but implement one revision of a module at each time. To indicate the details changes of that module, e.g., identifies schema node change path,location, operation and associated with corresponding schema node, it will be desirable to use 'ietf-module-revision' defined in this document to manage all the revisions of that module and keep track of module change discrepancy in different revision, especially when the new revision is not backward compatible with previous revision.

7. Yang Data Model Definition

<CODE BEGINS> file "ietf-module-revision@2018-08-08.yang"

```
module ietf-module-revision {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-module-revision";
  prefix ml;

  import ietf-yang-library {
    prefix yanglib;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF Network Modeling (NETMOD) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>

     WG List:  <mailto:netmod@ietf.org>

     Author:   Qin Wu
               <mailto:bill.wu@huawei.com>
               Zitao Wang
               <mailto:wangzitao@huawei.com>";
  description
```

```
    "This YANG module defines an module log.";

  revision 2018-08-08 {
    description
      "Initial revision.";
    reference "RFC XXXX: Using Metadata with YANG for Module revisions";
  }

  identity operation-type {
    description
      "Abstract base identity for the operation type ";
  }

  identity create {
    base operation-type;
    description
      "Denotes create new data nodes";
  }

  identity delete {
    base operation-type;
    description
      "Denotes delete the target node";
  }

  identity move {
    base operation-type;
    description
      "Denote move the target node.";
  }

  identity modify {
    base operation-type;
    description
      "Denote modify the target data node.";
  }

  identity statement-type {
    description
      "Base identity for statement type";
  }

  identity feature-statement {
    base statement-type;
    description
      "feature statement, if this type be chose, it means that the
       feature or if-feature statement been modified";
  }
```

```
  identity identity-statement {
    base statement-type;
    description
      "identity statement, if this type be chose, it means that the
       identity statement been modified, for example, add new identity, etc.";
  }

  identity grouping-statement {
    base statement-type;
    description
      "grouping statement, if this type be chose, it means that the grouping
       statement been modified.";
  }

  identity typedef-statement {
    base statement-type;
    description
      "typedef statement, if this type be chose, it means that the typedef
       statement been modified.";
  }

  identity augment-statement {
    base statement-type;
    description
      "augment statement, if this type be chose, it means that the augment
       statement been modified.";
  }

  identity rpc-statement {
    base statement-type;
    description
      "rpc statement, if this type be chose, it means that the rpc
       statement been modified.";
  }

  identity notification-statement {
    base statement-type;
    description
      "notification statement, if this type be chose, it means that the notifica
tion
       statement been modified.";
  }

  extension purpose {
   argument name;
    description
      "The purpose can be used to mark the data nodes change purpose.
       The name argument can be specified in the following recommended mode
```

```
       - bug-fix, which can help user to understand the data nodes' changes pres
ent bug fix,
       - new-function, which can help user to understand the data nodes' changes
 present new function,
       - nmda-conform, which can help user to understand the data nodes' changes
 conform to NMDA,

       and note that the user can argument the purpose name according to their s
epcific requirements.";
  }

  grouping data-definition {
    container data-definition {
      leaf target-node {
        type yang:xpath1.0;
        mandatory true;
        description
          "Identifies the target data node for update.
           Notice that, if the update-type equal to move or delete,
           this target-node must point to the data node of old version.
           \t
           For example, suppose the target node is a YANG leaf named a,
           and the previous version is:
           \t
           container foo {
            leaf a { type string; }
            leaf b { type int32; }
           }
           \t
           the new version is:
            container foo {
           leaf b {type int32;}
            }
           \t
           Therefore, the targe-node should be /foo/a.";
      }
      leaf location-point {
        type yang:xpath1.0;
        description
          "Identifies the location point where the updates happened.";
      }
      leaf where {
        when "derived-from-or-self(../../change-operation, 'move')" {
          description
            "This leaf only applies for 'move'
             updates.";
        }
        type enumeration {
          enum "before" {
            description
              "Insert or move a data node before the data resource
               identified by the 'point' parameter.";
```

```
          }
          enum "after" {
            description
              "Insert or move a data node after the data resource
               identified by the 'point' parameter.";
          }
          enum "first" {
            description
              "Insert or move a data node so it becomes ordered
                 as the first entry.";
          }
          enum "last" {
            description
              "Insert or move a data node so it becomes ordered
               as the last entry.";
          }
        }
        default "last";
        description
          "Identifies where a data resource will be inserted
              or moved.";
      }
      anydata data-definition {
        when "derived-from-or-self(../../change-operation, 'modify')" {
          description
            "This nodes only be present when
                the 'change-operation' equal to 'modify'.";
        }
        description
          "This nodes used for present the definitions before updated.
           And this nodes only be present when
           the 'change-operation' equal to 'modify'.";
      }
      description
        "Container for data statement";
    }
    description
      "Grouping for data definition";
  }

  grouping other-statement {
    container other-statement {
      leaf statement-name {
        type identityref {
          base statement-type;
        }
        description
          "Statement name, for example, identity, feature, typedef, etc.";
```

```
      }
      anydata statement-definition {
        description
          "This nodes used for present new the definitions.";
      }
      list substatements {
        key "statement-name";
        leaf statement-name {
          type identityref {
            base statement-type;
          }
          description
            "Statement name, for example, identity, feature, typedef, etc.";
        }
        anydata substatement-definition {
          description
            "This nodes used for present new the definitions.";
        }
        description
          "List for substatements updates";
      }
      description
        "Container for header statement updates";
    }
    description
      "Grouping for header statement";
  }

  grouping change-log {
    list revision-change-log {
      key "index";
      leaf index {
        type uint32;
        description
          "Index for module change log";
      }
      leaf change-operation {
        type identityref {
          base operation-type;
        }
        mandatory true;
        description
          "This leaf indicate the change operation, such as create, move, delete
, modify, etc.";
      }
      choice yang-statements {
        description
          "Choice for various YANG statements that have been impacted.";
        case data-definition-statement {
```

```
          uses data-definition;
        }
        case other-statement {
          uses other-statement;
        }
      }
      description
        "List for module revision change log";
    }
    description
      "Grouping for module revision change log";
  }

  container yang-modules {
    config false;
    list module {
      key "name revision";
      leaf name {
        type yang:yang-identifier;
        description
          "The YANG module or submodule name.";
      }
      leaf revision {
        type yanglib:revision-identifier;
        description
          "The YANG module or submodule revision date.  If no revision
           statement is present in the YANG module or submodule, this
           leaf is not instantiated.";
      }
      leaf backward-compatible {
        type boolean;
        description
          "Indicates whether it is a backward compatible version.
           If this parameter is set to true, it means that this version is
           a backwards compatible version";
      }
      uses change-log;
      description
        "List for module updated log";
    }
    description
      "This container present the modules updated log.";
  }
  augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
    description
      "Augment the yang library with backward compatibility indication.";
    leaf backward-compatible {
      type boolean;
```

```
      description
        "backward compatibility indication.";
    }
  }
  augment "/yanglib:yang-library/yanglib:module-set/yanglib:module/yanglib:submo
dule" {
    description
      "Augment the yang library with backward compatibility indication.";
    leaf backward-compatible {
      type boolean;
      description
        "backward compatibility indication.";
    }
  }
  rpc module-revision-change {
    description
      "Module Node change query operation.";
    input {
      leaf source-module-name {
        type yang:yang-identifier;
        mandatory true;
        description
          "The Source YANG module or submodule name.";
      }
      leaf source-revision {
        type yanglib:revision-identifier;
        description
          "The Source YANG module revision date.  If no revision
           statement is present in the YANG module or submodule, this
           leaf is not instantiated.";
      }
     leaf target-module-name {
        type yang:yang-identifier;
        mandatory true;
        description
          "The Target YANG module or submodule name.";
      }
      leaf target-revision {
        type yanglib:revision-identifier;
        description
          "The target YANG module revision date.  If no revision
           statement is present in the YANG module or submodule, this
           leaf is not instantiated.";
      }
    }
    output {
      choice status-response{
      leaf wrong-match{
          type empty;
```

```
          description
            "This leaf indicates that two modules have nothing in common.";
        }
        list data-nodes {
          key "data-node-name";
          description
            "Each entry represents a data node of a given module that
             have been changed from source revision of
             a module to target revision of the module.";
          leaf data-node-name {
            type string;
            description
              "a data node name of a given module that
               has been changed.";
          }
          leaf is-new-node {
            type boolean;
            description
              "indicate the data node is newly introduced node in the target revis
ion.";
          }
          leaf change-operation {
            type identityref {
              base operation-type;
            }
            description
            "This leaf indicate the change operation,
             such as create, move, delete, modify, etc.";
          }
        }
       }
      }
    }
  }
}
```

<CODE ENDS>

8. Security Considerations

This document defines a mechanism that is put at a place in the node hierarchy where data node gets changed or new data gets inserted and indicate whether the change to the data node is backward compatible and as such doesn't introduce new security issues.

9. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

```
---------------------------------------------------------------------
   URI: urn:ietf:params:xml:ns:yang:ietf-module-revision

   Registrant Contact: The NETMOD WG of the IETF.

   XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------
```

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
---------------------------------------------------------------------
   Name:         ietf-module-revision
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-module-revision
   Prefix:       md
   Reference:    RFC xxxx
---------------------------------------------------------------------
```

10. Acknowledgements

This work is motivated from the discussions of module version in IETF 100 Singapore meeting. Thanks to Juergen Schoenwaelder and Adrian Farrel for useful comments on this work.

11. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-04 (work in progress), January 2018.

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-15 (work in progress), December 2017.

[I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-00 (work in progress), July 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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/info/rfc6020>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[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>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

Appendix A. Example of Module Revision Management

This section provides an example to generate XML snippet of ietf-module-revision based on the changes between the new revision of interface model as specified in [draft-ietf-netmod-rfc7223bis] and the old revision of interface model as specified in [RFC7223].

```
<yang-modules>
 <module>
  <name>ietf-interfaces</name>
  <revision>2018-01-09</revision>
  <backwards-compatible>false</backwards-compatible>
  <revision-change-log>
   <index>0001</index>
   <change-operation>delete</change-operation>
   <yang-statements>
    <data-definition-statement>
     <data-definition>
      <target>/if:interfaces-state</target>
     </data-definition>
    </data-definition-statement>
   </yang-statements>
  </revision-change-log>

  <revision-change-log>
   <index>0002</index>
   <change-operation>create</change-operation>
   <yang-statements>
    <other-statement>
     <statement>
      <statement-name>feature-statement</statement-name>
```

```
            <statement-definition>
             feature if-mib {
              description
              "This feature indicates that the device implements
               the IF-MIB.";
              reference
              "RFC 2863: The Interfaces Group MIB";
             }
            </statement-definition>
           </statement>
          </other-statement>
         </yang-statements>
        </revision-change-log>

        <revision-change-log>
         <index>0003</index>
         <change-operation>modify</change-operation>
         <yang-statements>
          <data-definition-statement>
           <data-definition>
            <target>
            /if:interfaces/if:interface/if:link-up-down-trap-enable
            </target>
            <data-node>
             leaf link-up-down-trap-enable {
             if-feature if-mib; // add if-feature statement
              type enumeration {
              ....
            </data-node>
           </data-definition>
          </data-definition-statement>
         </yang-statements>
        </revision-change-log>

        <revision-change-log>
         <index>0004</index>
         <change-operation>move</change-operation>
         <yang-statements>
          <data-definition-statement>
           <data-definition>
            <target>
            /if:interfaces-state/if:interface/if:admin-status
            </target>
           <location-point>
            /if:interfaces/if:interface/link-up-down-trap-enable
           </location-point>
           <where>after</where>
          </data-definition-statement>
```

```
         </yang-statements>
        </revision-change-log>

        <revision-change-log>
         <index>0005</index>
         <change-operation>move</change-operation>
         <yang-statements>
          <data-definition-statement>
           <data-definition>
            <target>
            /if:interfaces-state/if:interface/if:statistics
            </target>
            <location-point>
            /if:interfaces/if:interface/if:speed
            </location-point>
            <where>after</where>
          </data-definition-statement>
         </yang-statements>
        </revision-change-log>
        .......
       </module>
      </yang-modules>
```

Authors' Addresses

Michael Wang
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing, Beijing 102209
China

Email: wangaj.bri@chinatelecom.cn

Netmod Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: July 30, 2018

Q. Wu
Huawei
A. Farrel
Juniper Networks
B. Claise
Cisco Systems, Inc.
January 26, 2018

Documentation Conventions for Expressing YANG in XML
draft-wu-netmod-yang-xml-doc-conventions-00

Abstract

Many documents that define YANG modules also include examples presented in XML.

IETF documentation has specific limits on line length and some XML examples have to include line wraps that would not normally be allowed according to the XML representation rules of RFC7950 and RFC7952.

This document lays out documentation conventions that allow YANG examples to be presented in IETF documentation when leaf node encoding would otherwise exceed the maximum line length. There are no implications in this document for YANG parsers: this document does not change the rules for presenting YANG models or for encoding YANG in data files or in the wire.

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 July 30, 2018.

Table of Contents

1. Introduction

YANG [RFC7950] defines four main types of data node for data modeling and describes how these are represented in XML [XML]. For list nodes and container nodes, any whitespace, carriage returns, or line feeds between the subelements is insignificant, i.e., an implementation MAY insert whitespace, carriage return, or line feed characters between subelements.

However for leaf nodes, [RFC7950] section 7.6.6 says

> The value of the leaf node is encoded to XML according to the type and is sent as character data in the element.

Thus whitespace, carriage return, and line feed characters are interpreted as part of the leaf value if the leaf is of type string and must not be included. The same applies to leaf-list nodes.

However, when documenting examples of YANG modules represented in XML encoding it is possible that the encoding of a single leaf node will exceed the available line length (73 characters).

This document describes documentation conventions that allow the presentation of such examples in a way that is easily parsed by a human reader, but which is not representative of how the XML must be presented to a software component or carried on the wire.

There are no implications in this document for YANG parsers: this document does not change the rules for presenting YANG models or for encoding YANG in data files or in the wire.

2. Conventions Used in this Document

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.

3. Separating Components of a Leaf Example

An example of the documentation of a leaf node is shown in Figure 1. The leaf is called "long-leaf-node-name" and is assigned the value "long-leaf-node-value". As can be seen in the example, this fits on one line. However it would only take the addition of a few more characters to the node label or value for the example to overflow the 73 character limit.

```
<long-leaf-node-label>long-leaf-node-value</long-leaf-node-label>
```

Figure 1: A Simple Leaf Node Example

For the sake of documentation, the representation shown in Figure 2 SHALL be considered as equivalent to that shown in Figure 1, but when a document uses this convention it MUST also include the text shown in Section 5.

```
<long-leaf-node-label>
  long-leaf-node-value
</long-leaf-node-label>
```

Figure 2: A Split Leaf Node Example

4. Splitting an Example Leaf Node Value Across Lines

When the XML representation of a leaf node value in an example would result in a line being longer than the maximum line length for an IETF document the value of the leaf node must be split and printed on more than one lines. This is most likely to happen when the example leaf node contains a string. Indeed, if this problem arises for other leaf types it may be indicative of poorly chosen leaf values, and the YANG definitions should be revised.

In this case, conventions MUST be observed:

- The broken line MUST be terminated with a backslash ("\") without the addition of any additional space before the backslash and with no further characters after the backslash.
- Any continuation lines MUST be indented with a whitespace offset of at least two characters.
- When a backslash appears in the node value, the example MUST be arranged so that the backslash is not the final character of a broken line

Furthermore, whenever a document uses this convention it MUST also include the text shown in Section 5.

Figure 3 shows an example leaf with a long value. As can be seen, the addition of a few more characters would cause the line to be too long.

Figure 4 shows a semantically equivalent representation of the example if the text from Section 5 is also present.

```
<long-leaf-string-node-label>
   Once upon a time, in a land far away, there lived a Great King.
</long-leaf-string-node-label>
```

Figure 3: An Example Leaf Node With a Long String Value

```
<long-leaf-string-node-label>
  Once upon a time, \
    in a land far away, \
    there lived a Great King.
</long-leaf-string-node-label>
```

Figure 4: A Long String Leaf Node Example Split Across Lines

Figure 5 and Figure 6 show a more complex example where the node value includes both line feeds and a backslash. Note how the line breaks are arranged to avoid potential confusion and to make the real characters evident.

```
<long-leaf-complex-string-node-label>
  Punctuation is important. As are line feeds.
  Some characters are special. E.g., the backslash \. Don't forget.
</long-leaf-string-node-label>
```

Figure 5: An Example Leaf Node With a Complex String Value

```
<long-leaf-complex-string-node-label>
  Punctuation is important. \
    As are line feeds.
  Some characters are special. \
    E.g., the backslash \. \
    Don't forget.
</long-leaf-string-node-label>
```

Figure 6: An Example Leaf Node With a Complex String Value Split Across Lines

5. Mandatory Boilerplate Text

When either of the conventions described in Section 3 or Section 4 is used for the benefit of the representation of an example of a YANG module or YANG fragment in XML, the following text MUST be included in the document presenting the example.

> The examples in this document adopt the conventions shown in BCP XX [RFCYYYY] for splitting node labels and node values onto separate lines. This convention is used to make the examples easier to read but does not change the encoding rules for the XML representation of YANG as described in [RFC7950].

RFC Editor Note: Please replace XX and YYYY with the numbers assigned for this document.

6. Representing XML Encodings of Metadata Annotations

[RFC7952] section 5.1 provides an encoding rule for metadata annotations in XML.

When an example XML representation of a leaf node element that includes metadata attributes results in a line being longer than the maximum number of characters allowed in a line of an IETF document, the value of the leaf node must be split across more than one line.

Where possible, all line breaks should be inserted between metadata attributes. Continuation lines MUST start with a whitespace offset of at least two characters. The leading and trailing whitespace of each line MUST be ignored. Figure 7 gives an example.

Whenever this documentation convention is used, the boilerplate text shown in Section 7 MUST be present in the document using the convention.

```
<error-path
   xmlns:t="http://example.com/schema/1.2/config/verylongpathname\
     thatcannotfitononeline">
   /t:top/t:interface[t:name="Ethernet0/0"]/t:mtu/t:anotherattribute
   /t:afinalattribute
</error-path>
```

Figure 7: An Example Leaf Node With Metadata Split Across Lines

7. Mandatory Boilerplate for Splitting Metadata Annotations

When the convention described in Section 6 is used for the benefit of the representation of an example of a YANG module or YANG fragment containing metadata annotations in XML, the following text MUST be included in the document presenting the example.

> The examples in this document adopt the conventions shown in BCP XX [RFCYYYY] for splitting metadata annotation across multiple lines. This convention is used to make the examples easier to read but does not change the encoding rules for the XML representation of YANG metadata annotations as described in [RFC7952].

RFC Editor Note: Please replace XX and YYYY with the numbers assigned for this document.

8. Automatic Generation of Valid XML From Examples

It should be noted that it is never the intention that example YANG fragment should be converted to XML that is passed to a YANG consumer. Nevertheless, there are good reasons to be able to convert an example into valid YANG in order to parse it and check its validity against the YANG model itself. This will ensure that examples in documents are accurate and useful.

When parsing a leaf or leaf-list node in an example, the following rules should be applied to generate valid XML.

- If a white space, carriage return, or line feed character is encountered between close (">") and open ("<") angle brackets it should be stripped.

- If a white space, carriage return, or line feed character is encountered within a string value of a leaf node or leaf-list node, it should generally be preserved exactly as shown except in the special case that follows.

- If a backslash character ("\") appears within the string value of a leaf node or leaf-list node and if and only if it is immediately followed by a carriage return or line feed character then all carriage return, line feed, and whitespace characters should be stripped until the next character is encountered.

- If a white space, carriage return, or line feed character is encountered within metadata annotations, but not within quotes, it should be stripped. Parsing may expect the next valid character found to indicate the start of a new metadata attribute.

- If a backslash character ("\") appears within the quoted value of a metadata attribute and if and only if it is immediately followed by a carriage return or line feed character then all carriage return, line feed, and whitespace characters should be stripped until the next character is encountered.

9. Security Considerations

There is no direct security impact related to the XML encoding documentation convention described in this document. However, attempting to provide actual XML using the documentation conventions described in this document would have unpredictable results. The risk here is that someone uses an example as a template for actual

XML. The mandatory boilerplate text provides a mitigation against this risk.

10. IANA Considerations

There are no IANA requests or assignments included in this document.

11. Acknowledgements

Thanks to Kent Watsen for discussions that kept us close to being on the right track. Additional thanks to John Scudder for flagging some nits.

12. 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>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[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>.

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <https://www.w3.org/TR/2008/REC-xml-20081126/>.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Adrian Farrel
Juniper Networks

Email: afarrel@juniper.net

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

Netmod Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: December 18, 2018

Q. Wu
Huawei
A. Farrel
Juniper Networks
B. Claise
Cisco Systems, Inc.
June 16, 2018

Documentation Conventions for lines wrapping and indentation in authored work
draft-wu-netmod-yang-xml-doc-conventions-05

Abstract

Many documents that define YANG modules or YANG fragments also include protocol message instance data examples.

IETF documentation has specific limits on line length (73 characters) and some YANG fragment example or protocol message instance data examples such as XML encoded YANG data node instance examples have to include line wraps that would not normally be allowed according to the XML representation rules of RFC7950 and RFC7952.

This document lays out documentation conventions that allow authored work to be presented in IETF documentation when authored work such as YANG fragment or protocol message instance data example would otherwise exceed the maximum line length and provide consistent representation of authored work within an Internet-Draft or RFC. There are no implications in this document for YANG tools: this document does not change the rules for presenting authored work in data files or in the wire.

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 December 18, 2018.

Copyright Notice

Table of Contents

1. Introduction

When documenting authored work such as YANG fragments example of example of YANG module represented in XML encoding it is possible that the representation of these authored work will exceed the available line length. Indentation may further aggravate this issue. The line wrapping is needed for formatting purposes, however different document author may take different ways to wrap line which

makes difficult to improve the readability and interoperability of published YANG data models.

This document lays out documentation conventions that allow authored work to be presented in IETF documentation when authored work such as YANG fragment or protocol message instance data example would otherwise exceed the maximum line length and provide consistent representation of authored work within an Internet-Draft or RFC.

Document conventions defined in this document are not representative of how the Authored work must be presented to a software component or carried on the wire. There are no implications in this document for YANG tools(e.g., libyang parser): this document does not change the rules for presenting YANG models or for encoding YANG in data files or in the wire.

2. Conventions Used in this Document

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] and are not redefined here:

- data node
- leaf
- leaf-list
- instance

The following term is defined in [RFC7951] and [RFC7952] and are not redefined here:

- data node Instance
- data node identifier

The following terms are defined in [RFC8340]and [I-D.ietf-netmod-rfc6087bis].

2.1. Glossary of New Terms

Authored work: A set of text format work representing YANG fragments, and protocol message instance data except YANG Tree Diagrams.

Wrap: Convert authored work with long lines not fitting into an Internet-Draft or RFC into authored work with split line fitting into an Internet-Draft or RFC.

Unwrap: Re-Convert authored work with split line fitting into an Internet-Draft or RFC back to valid authored work without split line that can be consumed by a software component or carried on the wire.

Indent: used to describe the distance, or number of blank spaces used to separate a paragraph from the left or right margins.

Libyang parser: YANG tool and library for parsing and validating YANG schemas and instance data.

3. Long line wrapping Example

An example of the documentation of a leaf node is shown in Figure 1. The container node is called <parent-node-label>, any whitespace, carriage returns, or line feeds between the subelements <parent-node-label> is insignificant, i.e., an implementation MAY insert whitespace, carriage return, or line feed characters between subelements. The leaf is called "long-leaf-node-label" and is assigned the value "long-leaf-node-value". As can be seen in the example, this fits on one line. However it would only take the addition of a few more characters to the node label or value for the example to overflow the 73 character limit if the line of leaf node instance is indented (e.g., start below <parent-node-label> with a whitespace offset of two characters. .

```
<parent-node-label>
  <long-leaf-node-label>long-leaf-node-value</long-leaf-node-label>
</parent-node-label>
```

Figure 1: A Simple Leaf Node Example

For the sake of documentation purpose, the representation shown in Figure 2 SHALL be considered as equivalent to that shown in Figure 1, but when a document uses this convention it MUST also include the text shown in Figure 3. Note that the first example representation in figure 2 is more easily parsed by a human reader than the second example in figure 2.

```
<parent-node-label>
  <long-leaf-node-label>\
    long-leaf-node-value\
  </long-leaf-node-label>
</parent-node-label>
Or
<parent-node-label>
  <long-leaf-node-label> long-leaf-node-value </long-leaf-nod\
    e-label>
</parent-node-label>
```

Figure 2: A Split Leaf Node Example

4. Objectives

In order to allow authored work to be presented in IETF documentation when authored work such as YANG fragment or protocol message instance data example would otherwise exceed the maximum line length and provide consistent representation of the authored work within an Internet-Draft or RFC, the following design criteria are used:

- Allow automatic wrapping line when any line presented in the authored work of I-D or RFCs exceed the maximum line length.

- Allow automatic unwrapping line in the artwork when the artwork needs to be presented to a software component or carried on the wire.

5. Line wrapping and indentation document convention

When the representation of an authored work (e.g.,a leaf node instance representation) in an example would result in a line being longer than the maximum line length for an IETF document the long line must be split and presented on more than one lines. The new line may be indented, if necessary, so that it starts below the first line with a whitespace offset of two characters, which improve readability and interoperability of published YANG data models.

When these authored work with split lines needs to be fed into software component or carried in the wire, these authored work with split lines should be unwrapped and reversed into the valid authored work with long line. If the indentation is applied to authored work with split lines, the indentation should be removed during unwrapped process.

5.1. Long line wrapping

Long line wrapping most likely to happen when the authored work example such as leaf node contains built-in type string or datetime or container node and list node includes metadata attributes. Indeed, if this problem arises for other YANG types it may be indicative of poorly chosen YANG type values, and the YANG definitions should be revised before applying document convention for line wrapping defined in this document.

In the case of long line exceeding 73 characters, the following long line wrapping conventions MUST be observed:

- Split long line in the authored work (e.g.,leaf node instance, YANG data node instance containing metadata annotation attributes) exceeding 73 characters limits with the backslash ("\") and use backslash ("\")to indicate wrapping at the end of the line. The broken line MUST be terminated with a backslash ("\") without the addition of any additional space before the backslash and with no further characters after the backslash.

- Any continuation lines or new line MUST align with the first line and MAY chose be indented with two whitespace offset for readability purposes.

- When a backslash appears in any line not used for split line, the representation of this artwork MUST be arranged so that this backslash is not the final character of a broken line. If this backslash is the second last character (e.g., backslash at the position 72) of a broken line, the line should be split at the position one or several characters before this backslash as the second last character with the backslash ("\") . In extreme case, if a long line is full of backslashes, the backslashs before backslash at position 73 in this line should be treated in the same way as other normal characters.

Furthermore, whenever a document uses long line wrapping conventions it MUST also include the following boilerplate text :

```
[!!! '\' line wrapping is for formatting only and adopt the conventions
shown in BCPXX [RFCYYYY]]
<WRAPPED TEXT BEGIN>
 ....//Authored work
<WRAPPED TEXT END>
RFC Editor Note: Please replace XX and YYYY with the numbers assigned
for this document.
```

Figure 3

Figure 4 shows an example of Backslash appearing in the long line not used for split line.

```
<long-leaf-complex-string-node-label>Punctuation is important. As
are line feeds.Some characters are special,e.g., the backslash\.
Don’t forget. </long-leaf-string-node-label>
```

Figure 4: An Example Leaf Node With a Complex String Value

Figure 5 shows a semantically equivalent representation of the example.

```
<long-leaf-complex-string-node-label>Punctuation is important. As \
are line feeds.Some characters are special,e.g., the backslash \.\
Don’t forget.</long-leaf-string-node-label>
```

Figure 5

5.2. Line unwrapping

If line wrapping is done for formatting purposes, the line wrapping in the authored work should be reversed back or unwrapped before the authored work is fed into software component for validation or carried in the wire. Therefore line unwrapping help remove backslash and additional carriage return or line feed character and make unwrapped authored work to be effectively compliant with the tool. The line wrapping for formatting purpose is indicated by the above boilerplate text in Figure 3. To unwrap line, the following conventions must be observed:

- Consecutive split lines in the authored work with backslash at the end of the line should be merged into one long line, the last split line in Consecutive split lines should not be terminated with backslash.

- If a backslash character ("\") doesn’t appear at the end of the line within authored work, it should not be stripped.

- If a backslash character ("\") appears at the end of the line within authored work, it should be stripped. In the meanwhile, if and only if it is immediately followed by a carriage return or line feed character then all carriage return, line feed, and whitespace characters should be stripped until the next character is encountered.

- In extrem case, if a backslash character ("\") or space character appears full of line, the full line of backslash character ("\") or space character should be stripped.

5.3. Auto indentation and dedentation

Consistent indentation should be used for all authored work in the I-D and RFCs, e.g., if a space or tab characters are used to index the text in the long line during wraping process, the space and tab characters used for indentation should be removed during unwrapping process. If the new line or continuation line indented with a whitespace offset of two characters during wrapping process, the indentation with a whitespace offset of two characters should be removed during unwrapping process.

6. Limitation and complexity

6.1. Limitations

All modules need to be extracted YANG modules from an Internet Draft or RFC and then validated before submission in an Internet Draft. However we don’t have automation tool to extract authored work such as YANG fragments or protocol message instance. To extract authored work, the similar strings "<CODE BEGINS>" and "<CODE ENDS>" MUST be defined and populated to identify each authored work component, e.g., the boilerplate text in Section 5 can be used to indicate the begining of authored work.

Applying wrapping and unwrapping functionality to example YANG module or YANG module extracted using existing tool also has limitation, even introduce confusion, e.g.,

1. The data definition description statement has long line exceeding 73 characters, it should be wrapped without using backslash as termination point.

"

```
 grouping link-ref {
   description
     "This grouping can be used to reference a link in a specific
      network.  Although it is not used in this module, it is
      defined here for the convenience of augmenting modules.";
```

"

2. Another example is when a plus character ("+") is used to concatenate two quoted string into one string, using backslash to split the line Confuses with using a plus character ("+") to split the line.

```
"
  container dhcp-relay {
    when "derived-from-or-self(../address-allocation-type, "+
    "'l3vpn-svc:provider-dhcp-relay')" {
      description
      "Only applies when provider is required to implement
      DHCP relay function.";
   }
"
```

6.2. Complexity

We can build tool to support auto wrap and auto indentation. However if the tool is designed to understand various encodings, e.g., XML encoding, JSON encoding or metadata annotation, it adds a lot of complexity to build such tool, therefore the only choice to make tool understand various encodings, is to build encoding specific tool which doesn't scale well,e.g., if the tool understands metadata annotation, we can decide where to insert backslash to split the lines: either inserted between metadata Attributes or insert at any place when the long line exceeding 73 characters limits. See more complexity details in Appendix A.

7. Security Considerations

There is no direct security impact related to the documentation convention for lines wrapping and indentation in authored work described in this document. However, attempting to provide representation of authored work using the documentation conventions described in this document would have unpredictable results. The risk here is that someone uses an example as a template for actual authored work representation. The mandatory boilerplate text provides a mitigation against this risk.

8. IANA Considerations

There are no IANA requests or assignments included in this document.

9. Acknowledgements

Thanks to Kent Watsen for discussions that kept us close to being on the right track. Additional thanks to John Scudder for flagging some nits, Martin Bjorklund, Charles Eckel, Robert Wilton and many others for valuable comments and review, special thanks Xiongjie to help support automation tool building.

10. Normative References

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-20 (work in progress), March 2018.

[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>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[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>.

[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>.

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <https://www.w3.org/TR/2008/REC-xml-20081126/>.

Appendix A. Representing XML and JSON Encodings of Metadata Annotations

[RFC7952] section 5.1 and section 5.2 provide an encoding rule for metadata annotations in XML and JSON respectively.

When an example XML representation of a leaf node element that includes metadata attributes results in a line being longer than the maximum number of characters allowed in a line of an IETF document, the value of the leaf node must be split across more than one line.

Where possible, all line breaks should be inserted between metadata attributes. Continuation lines MUST align with the first line and not be indented with any whitespace. The leading and trailing whitespace of each line MUST be ignored. Figure 6 gives a XML example.

When an example JSON representation of a leaf node element that includes metadata attributes starting with the "@" character results in a line being longer than the maximum number of characters allowed in a line of an IETF document,the value of the leaf node must be split across more than one line. Continuation lines MUST align with the first line and indented with one whitespace character. The leading and trailing whitespace of each line MUST be ignored. Figure 7 gives a JSON example.

Whenever this documentation convention is used, the boilerplate text shown in Figure 3 MUST be present in the document using the convention.

```
<foo xmlns:elm=http://example.org/example-last-modified\
elm:last-modified="2015-09-16T10:27:35+02:00">
  ...
</foo>
```

Figure 6: An XML Example Leaf Node With Metadata Split Across Lines

```
"cask": {
  "@": {
    "example-org-example-last-modified:last-modified":\
    "2015-09-16T10:27:35+02:00"
  },
  ...
}
```

Figure 7: A JSON Example Leaf Node With Metadata Split Across Lines

Appendix B. Auto-wrapping tool code

We provide examples of python code for aspects of line wrapping and unwrapping algorithms. There may be other implementation methods that are faster in particular operating environments or have other advantages. These implementation notes are for informational purposes only and are meant to clarify the this specification for line wrapping and unwrapping.

```
#!/usr/bin/env python2.7
# -*- coding: utf-8 -*-
"""Qin Wu, 2018-06-02
```

```
Autowrapper.py uses Text Wrap Module as library and support auto wrap and auto i
ndent
two functionalities.
(1)Lines with "\" in position 72 have been handled.
(2)Lines with space in position 73 have been handled.
(3)A line of "\" has been handled.
(4)A line of space has been hanled.
https://github.com/sunseawq/auto-wrap-indent/blob/master/autowrapper.py
Text Wrap module provides two convenience functions, wrap() and fill(), as well
as
TextWrapper, the class that does all the work, and a utility function dedent().
If
you're just wrapping or filling one or two text strings, the convenience functio
ns
should be good enough; otherwise, you should use an instance of TextWrapper for
efficiency.

https://github.com/python/cpython/blob/2.7/Lib/textwrap.py
"""
import textwrap
import string, re
import argparse
import os.path
import sys, getopt

def indent(text, prefix, predicate=None):
    """Adds 'prefix' to the beginning of selected lines in 'text'.

    If 'predicate' is provided, 'prefix' will only be added to the lines
    where 'predicate(line)' is True. If 'predicate' is not provided,
    it will default to adding 'prefix' to all non-empty lines that do not
    consist solely of whitespace characters.
    """
    if predicate is None:
        def predicate(line):
            return line.strip()

    def prefixed_lines():
        for line in text.splitlines(True):
            yield (prefix + line if predicate(line) else line)
    return ''.join(prefixed_lines())

def auto_wrap(input_file, dst_file):
    finput=open(input_file, "r")
    alllines=finput.readlines()
    finput.close()
    foutput = 0
    output_file = dst_file
    foutput = open(output_file, 'a')
    for eachline in alllines:
        bc = textwrap.fill(eachline,73)
        tmplines = bc.split('\n')
```

```
        tmplen = len(tmplines)
        if tmplen == 1 :
            foutput.writelines(bc)
            foutput.writelines('\n')
        else :
            i = 0
            while i < tmplen-1 :
                foutput.writelines(tmplines[i])
                foutput.writelines('\\')
                foutput.writelines('\n')
                i += 1
                foutput.writelines(tmplines[tmplen-1])
                foutput.writelines('\n')
    foutput.close

def auto_unwrap(input_file, dst_file) :
         finput=open(input_file, "r")
         alllines=finput.readlines()
         finput.close()
         foutput = 0
         output_file = dst_file
         foutput = open(output_file, 'a')
         for eachline in alllines:
             if eachline.endswith('\\\n') :
                 eachline = eachline.strip('\\\n')
             foutput.writelines(eachline)

def auto_wrap_indent(input_file, dst_file,width):
    finput=open(input_file, "r")
    alllines=finput.readlines()
    finput.close()
    foutput = 0
    flag_add = 0
    backslashpos = 0
    output_file = dst_file
    foutput = open(output_file, 'a')
    for eachline in alllines:
        backslashpos = eachline.rstrip('\\\n').rfind('\\',0,width)
        '''handle backslash at position 72'''
        if (backslashpos == width-1) :
            print("backslash appear at the end of the line,
            the line is wrapped at the position one or multiple characters
            before the backslash")
            bc = textwrap.fill(eachline,width-1)
        else :
            bc = textwrap.fill(eachline,73)
        '''handle space at position 71,72,73'''
        if eachline.rstrip('\n').rfind(' ',width-2,width) == width-2 :
```

```
            bc = bc[:width-2] + ' \n' + bc[width-1:]
        if eachline.rstrip(' \n').rfind(' ',width-2,width) == width-1 :
            bc = bc[:width-1] + ' \n' + bc[width:]
        if eachline.rstrip(' \n').rfind(' ',width-2,width+1) == width :
            bc = bc[:width] + ' \n' + bc[width+1:]
        tmplines = bc.split('\n')
        tmplen = len(tmplines)
        if tmplen == 1 :
            foutput.writelines(bc)
            foutput.writelines('\n')
        else :
            flag_add = 0
            i = 0
            while i < tmplen-1 :
                if(flag_add == 1) :
                    tmplines[i] = indent(tmplines[i], '  ')
                foutput.writelines(tmplines[i])
                foutput.writelines('\\')
                flag_add = 1
                foutput.writelines('\n')
                i += 1
                if(flag_add == 1) :
                    tmplines[i] = indent(tmplines[i], '  ')
                foutput.writelines(tmplines[tmplen-1])
                foutput.writelines('\n')
    foutput.close


def auto_unwrap_dedent(input_file, dst_file) :
    finput=open(input_file, "r")
    alllines=finput.readlines()
    finput.close()
    foutput = 0
    flag_del = 0
    flag_space = 0
    output_file = dst_file
    foutput = open(output_file, 'a')
    for eachline in alllines:
        print(eachline)
        if(flag_del == 1) :
            eachline = eachline[2:]
        if eachline.endswith('\\\n') :
            flag_del = 1
            eachline = eachline.rstrip('\\\n')
            if eachline == '':
                flag_del = 0
        else :
            flag_del = 0
```

```
        if eachline == '\n' :
            continue
        foutput.writelines(eachline)

if __name__ == "__main__":
    auto_wrap("in-1.txt","out-1.txt")
    auto_unwrap("out-1.txt", "out-2.txt")
    auto_wrap_indent("in-1.txt","out-1.txt",73)
    auto_unwrap_dedent("out-1.txt", "out-2.txt")
```

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Adrian Farrel
Juniper Networks

Email: afarrel@juniper.net

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com