Network Working Group
Internet-Draft
Updates: 7950,8407,8525 (if approved)
Intended status: Standards Track
Expires: September 18, 2020

B. Claise
J. Clarke
R. Rahman
R. Wilton, Ed.
Cisco Systems, Inc.
B. Lengyel
Ericsson
J. Sterne
Nokia
K. D'Souza
AT&T

March 17, 2020

Updated YANG Module Revision Handling
draft-ietf-netmod-yang-module-versioning-00

Abstract

This document specifies a new YANG module update procedure that can document when non-backwards-compatible changes have occurred during the evolution of a YANG module. It extends the YANG import statement with an earliest revision filter to better represent inter-module dependencies. It provides help and guidelines for managing the lifecycle of YANG modules and individual schema nodes. This document updates RFC 7950, RFC 8407 and RFC 8525.

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 18, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
  1.1. Updates to YANG RFCs 4
2. Terminology and Conventions 4
3. Refinements to YANG revision handling 5
  3.1. Updating a YANG module with a new revision 5
    3.1.1. Backwards-compatible changes 5
    3.1.2. Non-backwards-compatible changes 6
  3.2. nbc-changes revision extension statement 6
  3.3. Revision label 6
  3.4. YANG status description extension statement 7
  3.5. Examples for updating the YANG module revision history . 7
4. Import by derived revision 10
  4.1. Module import examples 11
5. Updates to ietf-yang-library 13
  5.1. Resolving ambiguous module imports 13
  5.2. YANG library versioning augmentations 14
    5.2.1. Advertising revision-label 14
    5.2.2. Reporting how deprecated and obsolete nodes are handled 14
6. Versioning of YANG instance data 15
7. Guidelines for using the YANG module update rules 15
  7.1. Guidelines for YANG module authors 15
    7.1.1. Making non-backwards-compatible changes to a YANG module 16
  7.2. Versioning Considerations for Clients 17
8. Module Versioning Extension YANG Modules 18
9. Contributors 25
10. Security Considerations 26
11. IANA Considerations 26
  11.1. YANG Module Registrations 26
12. References 26

12.1. Normative References 26
12.2. Informative References 27
Appendix A. Appendix 28
A.1. Examples of guidelines for making NBC changes to a YANG module 28
A.1.1. Removing a data node 28
A.1.2. Changing the type of a leaf node 29
A.1.3. Reducing the range of a leaf node 30
A.1.4. Changing the key of a list 30
A.1.5. Renaming a node 31
A.1.6. Changing a default value 32
Authors' Addresses 32

1. Introduction

This document defines a solution to the YANG module lifecycle problems described in [I-D.verdt-netmod-yang-versioning-reqs]. Complementary documents provide a complete solution to the YANG versioning requirements, with the overall relationship of the solution drafts described in [I-D.verdt-netmod-yang-solutions].

Specifically, this document recognises a need (within standards organizations, vendors, and the industry) to sometimes allow YANG modules to evolve with non-backwards-compatible changes, which could cause breakage to clients and importing YANG modules. Accepting that non-backwards-compatible changes do sometimes occur, it is important to have mechanisms to report where these changes occur, and to manage their effect on clients and the broader YANG ecosystem.

The solution comprises five parts:

Refinements to the YANG 1.1 module revision update procedure, supported by new extension statements to indicate when a revision contains non-backwards-compatible changes, and an optional revision label.

A YANG extension statement allowing YANG module imports to specify an earliest module revision that may satisfy the import dependency.

Updates and augmentations to ietf-yang-library to include the revision label in the module descriptions, to report how "deprecated" and "obsolete" nodes are handled by a server, and to clarify how module imports are resolved when multiple versions could otherwise be chosen.

Considerations of how versioning applies to YANG instance data.

Guidelines for how the YANG module update rules defined in this document should be used, along with examples.

Open issues are tracked at <https://github.com/netmod-wg/yang-ver-dt/issues>.

1.1. Updates to YANG RFCs

This document updates [RFC7950] section 11. Section 3 describes modifications to YANG revision handling and update rules, and Section 4 describes a YANG extension statement to do import by derived revision.

This document updates [RFC8525] section 3. Section 5 defines how a client of a YANG library datastore schema chooses which revision of an import-only module is used to resolve a module import when the definition is otherwise ambiguous.

This document updates [RFC8407] section 4.7. Section 7 provides guidelines on managing the lifecycle of YANG modules that may contain non-backwards-compatible changes and a branched revision history.

2. Terminology and Conventions

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.

In addition, this document uses the terminology:

- YANG module revision: An instance of a YANG module, uniquely identified with a revision date, with no implied ordering or backwards compatibility between different revisions of the same module.

- Backwards-compatible (BC) change: A backwards-compatible change between two YANG module revisions, as defined in Section 3.1.1

- Non-backwards-compatible (NBC) change: A non-backwards-compatible change between two YANG module revisions, as defined in Section 3.1.2

3. Refinements to YANG revision handling

[RFC7950] assumes, but does not explicitly state, that the revision history for a YANG module is strictly linear, i.e., it is prohibited to have two independent revisions of a YANG module that are both directly derived from the same parent revision.

This document clarifies [RFC7950] to explicitly allow non linear development of YANG module revisions, so modules MAY have multiple revisions that directly derive from the same parent revision. As per [RFC7950], YANG module revisions continue to be uniquely identified by the module's revision date, and hence all revisions of a module MUST have unique revision dates.

A module's name and revision date identifies a specific immutable definition of that module within its revision history. Hence, if a module includes submodules then the module's "include" statements MUST use "revision-date" substatements to specify the exact revision date of each included submodule.

[RFC7950] section 11 requires that all updates to a YANG module are BC to the previous revision of the module. This document allows for more flexible evolution of YANG modules: NBC changes between module revisions are allowed and are documented using a new "nbc-changes" YANG extension statement in the module revision history.

3.1. Updating a YANG module with a new revision

This section updates [RFC7950] section 11 to refine the rules for permissible changes when a new YANG module revision is created.

Where pragmatic, updates to YANG modules SHOULD be backwards-compatible, following the definition in Section 3.1.1.

A new module revision MAY contain NBC changes, i.e., the semantics of an existing definition MAY be changed in an NBC way without requiring a new definition with a new identifier. A new module revision with NBC changes MUST include the "rev:nbc-changes" extension substatement to signal the potential for incompatibility to existing module users and readers.

3.1.1. Backwards-compatible changes

A change between two module revisions is defined as being "backwards-compatible" if the change conforms to the module update rules specified in [RFC7950] section 11, updated by the following rules:

- o A "status" "deprecated" statement MAY be added, or changed from "current" to "deprecated", but adding or changing "status" to "obsolete" is not a backwards-compatible change.

- o Obsolete definitions MAY be removed from published modules, and are classified as backwards-compatible changes. In some circumstances it may be helpful to retain the obsolete definitions to ensure that their identifiers are not reused with a different meaning.

- o In statements that have any data definition statements as substatements, those data definition substatements MAY be reordered, as long as they do not change the ordering or any "rpc" "input" substatements. If new data definition statements are added, they can be added anywhere in the sequence of existing substatements.

3.1.2. Non-backwards-compatible changes

Any changes to YANG modules that are not defined by Section 3.1.1 as being backwards-compatible are classified as "non-backwards-compatible" changes.

3.2. nbc-changes revision extension statement

The "rev:nbc-changes" extension statement is used to indicate YANG module revisions that contain NBC changes.

If a revision of a YANG module contains changes, relative to the preceding revision in the revision history, that do not conform to the module update rules defined in Section 3.1.1, then a "rev:nbc-changes" extension statement MUST be added as a substatement to the "revision" statement.

Conversely, if a revision does not contain an "rev:nbc-changes" extension substatement then all changes, relative to the preceding revision in the revision history, MUST be backwards-compatible.

3.3. Revision label

Each revision entry in a module or submodule MAY have a revision label associated with it, providing an alternative alias to identify a particular revision of a module or submodule. The revision label could be used to provide an additional versioning identifier associated with the revision.

YANG Semver [I-D.verdt-netmod-yang-semver] defines a versioning scheme based on Semver 2.0.0 [semver] that can be used as a revision

label. All revision labels that match the pattern for the "version" typedef in the ietf-yang-semver YANG module MUST be interpreted as YANG semantic version numbers.

The revision date and revision label within a submodule's revision history have no effect on the including module's revision. Submodules MUST NOT use revision label schemes that could be confused with the including module's revision label scheme.

If a revision has an associated revision label, then it may be used instead of the revision date in two places:

In an "rev:revision-or-derived" extension statement argument.

In the filename of a YANG module, where it takes the form: module-or-submodule-name ['@' revision-label] ('.yang' / '.yin')

3.4. YANG status description extension statement

The ietf-yang-revision module specifies the YANG extension statement "status-description" that can be used as a substatement of the status statement. The argument to this extension statement can contain freeform text to help readers of the module understand why the node was deprecated or made obsolete, when it is anticipated that the node will no longer be available for use, and potentially reference other schema elements that can be used instead. An example is shown below.

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

3.5. Examples for updating the YANG module revision history

The following diagram, explanation, and module history illustrates how the branched revision history, "nbc-changes" extension statement, and "revision-label" extension statement could be used:

Example YANG module with branched revision history.

```
       Module revision date        Revision label
         2019-01-01                 <- 1.0.0
             |
         2019-02-01                 <- 2.0.0
             |      \
         2019-03-01  \              <- 3.0.0
             |        \
             |       2019-04-01     <- 2.1.0
             |           |
             |       2019-05-01     <- 2.2.0
             |
         2019-06-01                 <- 3.1.0
```

The tree diagram above illustrates how an example module's version history might evolve, over time. For example, the tree might represent the following changes, listed in chronological order from oldest revision to newest:

Example module, revision 2019-06-01:

```
module example-module {

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

  import ietf-yang-revisions { prefix "rev"; }

  description
    "to be completed";

  revision 2019-06-01 {
    rev:revision-label 3.1.0;
    description "Add new functionality.";
  }

  revision 2019-04-01 {
    rev:revision-label 3.0.0;
    rev:nbc-changes;
    description
      "Add new functionality. Remove some deprecated nodes.";
  }

  revision 2019-02-01 {
    rev:revision-label 2.0.0;
    rev:nbc-changes;
    description "Apply bugfix to pattern statement";
  }

  revision 2019-01-01 {
    rev:revision-label 1.0.0;
    description "Initial revision";
  }

  //YANG module definition starts here
```

Example module, revision 2019-05-01:

```
module example-module {

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

  import ietf-yang-revisions { prefix "rev"; }

  description
    "to be completed";

  revision 2019-05-01 {
    rev:revision-label 2.2.0;
    description "Backwards-compatible bugfix to enhancement.";
  }

  revision 2019-03-01 {
    rev:revision-label 2.1.0;
    description "Apply enhancement to older release train.";
  }

  revision 2019-02-01 {
    rev:revision-label 2.0.0;
    rev:nbc-changes;
    description "Apply bugfix to pattern statement";
  }

  revision 2019-01-01 {
    rev:revision-label 1.0.0;
    description "Initial revision";
  }

  //YANG module definition starts here
```

4. Import by derived revision

RFC 7950 allows YANG module "import" statements to optionally require the imported module to have a particular revision date. In practice, importing a module with an exact revision date is often too restrictive because it requires the importing module to be updated whenever any change to the imported module occurs. The alternative choice of using an import statement without any revision date statement is also not ideal because the importing module may not work with all possible revisions of the imported module.

Instead, it is desirable for a importing module to specify a "minimum required revision" of a module that it is compatible with, based on

the assumption that later revisions derived from that "minimum required revision" are also likely to be compatible. Many possible changes to a YANG module do not break importing modules, even if the changes themselves are not strictly backwards-compatible. E.g., fixing an incorrect pattern statement or description for a leaf would not break an import, changing the name of a leaf could break an import but frequently would not, but removing a container would break imports if that container is augmented by another module.

The ietf-revisions module defines the "revision-or-derived" extension statement, a substatement to the YANG "import" statement, to allow for a "minium required revision" to be specified during import:

> The argument to the "revision-or-derived" extension statement is a revision date or a revision label.
>
> A particular revision of an imported module satisfies an import's "revision-or-derived" extension statement if the imported module's revision history contains a revision statement with a matching revision date or revision label.
>
> An "import" statement MUST NOT contain both a "revision-or-derived" extension statement and a "revision-date" statement.
>
> The "revision-or-derived" extension statement MAY be specified multiple times, allowing the import to use any module revision that satifies at least one of the "revision-or-derived" extension statements.
>
> The "revision-or-derived" extension statement does not gaurantee that all module revisions that satisfy an import statement are necessarily compatible, it only gives an indication that the revisions are more likely to be compatible. Hence, NBC changes to an imported module may also require new revisions of any importing modules, updated to accommodation those changes, along with updated import "revision-or-derived" extension statements to depend on the updated imported module revision.

4.1. Module import examples

Consider the example module "example-module" from Section 3.5 that is hypothetically available in the following revision/label pairings: 2019-01-01/1.0.0, 2019-02-01/2.0.0, 2019-03-01/3.0.0, 2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0. The relationship between the revisions is as before:

```
          Module revision date        Revision label
            2019-01-01                 <- 1.0.0
                |
            2019-02-01                 <- 2.0.0
                |      \
            2019-03-01  \              <- 3.0.0
                |        \
                |       2019-04-01     <- 2.1.0
                |           |
                |       2019-05-01     <- 2.2.0
                |
            2019-06-01                 <- 3.1.0
```

4.1.1. Example 1

This example selects module revisions that match, or are derived from the revision 2019-02-01. E.g., this dependency might be used if there was a new container added in revision 2019-02-01 that is augmented by the importing module.It includes revisions/labels: 2019-02-01/2.0.0, 2019-03-01/3.0.0, 2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0.

```
import example-module {
  rev:revision-or-derived 2019-02-01;
}
```

Alternatively, the first example could have used the revision label "1.0.0" instead, which selects the same set of revisions/versions.

```
import example-module {
  rev:revision-or-derived 1.0.0;
}
```

4.1.2. Example 2

This example selects module revisions that are derived from 2019-04-01 by using the revision label 2.1.0. It includes revisions/labels: 2019-04-01/2.1.0 and 2019-05-01/2.2.0. Even though 2019-06-01/3.1.0 has a higher revision label version number than 2019-04-01/2.1.0 it is not a derived revision, and hence it is not a valid revision for import.

```
import example-module {
  rev:revision-or-derived 2.1.0;
}
```

4.1.3. Example 3

This example selects revisions derived from either 2019-04-01 or 2019-06-01. It includes revisions/labels: 2019-04-01/2.1.0, 2019-05-01/2.2.0, and 2019-06-01/3.1.0.

```
import example-module {
  rev:revision-or-derived 2019-04-01;
  rev:revision-or-derived 2019-06-01;
}
```

5. Updates to ietf-yang-library

This document updates YANG library [RFC7895] [RFC8525] to clarify how ambiguous module imports are resolved. It also defines the YANG module, ietf-yl-revisions that augments YANG library with new versioning related meta-data.

5.1. Resolving ambiguous module imports

A YANG datastore schema, defined in [RFC8525], can specify multiple revisions of a YANG module in the schema using the "import-only" list, with the requirement from [RFC7950] that only a single revision of a YANG module may be implemented.

If a YANG module import statement does not specify a specific revision within the datastore schema then it could be ambiguous as to which module revision the import statement should resolve to. Hence, a datastore schema constructed by a client using the information contained in YANG library may not exactly match the datastore schema actually used by the server.

The following two rules remove the ambiguity:

If a module import statement could resolve to more than one module revision defined in the datastore schema, and one of those revisions is implemented (i.e., not an "import-only" module), then the import statement MUST resolve to the revision of the module that is defined as being implemented by the datastore schema.

If a module import statement could resolve to more than one module revision defined in the datastore schema, and none of those revisions are implemented, then the import MUST resolve to the module revision with the latest revision date.

5.2. YANG library versioning augmentations

The "ietf-yl-revisions" YANG module has the following structure (using the notation defined in [RFC8340]):

```
module: ietf-yl-revisions
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:schema:
    +--ro deprecated-nodes-implemented?   empty
    +--ro obsolete-nodes-absent?          empty
```

5.2.1. Advertising revision-label

The ietf-yl-revisions YANG module augments the "module" list in ietf-yang-library with a "revision-label" leaf to optionally declare the revision label associated wth the particular revision of each module.

5.2.2. Reporting how deprecated and obsolete nodes are handled

The ietf-yl-revisions YANG module augments YANG library with two leaves to allow a server to report how it handles status "deprecated" and status "obsolete" nodes. The leaves are:

deprecated-nodes-implemented: If present, this leaf indicates that all schema nodes with a status "deprecated" child statement are implemented equivalently as if they had status "current", or otherwise deviations MUST be used to explicitly remove "deprecated" nodes from the schema. If this leaf is absent then the behavior is unspecified.

obsolete-nodes-absent: If present, this leaf indicates that the server does not implement any status "obsolete" nodes. If this leaf is absent then the behaviour is unspecified.

Servers SHOULD set both the "deprecated-nodes-implemented" and "obsolete-nodes-absent" leaves.

If a server does not set the "deprecated-nodes-implemented" leaf, then clients MUST NOT rely solely on the "rev:nbc-changes" statements to determine whether two module revisions are backwards-compatible, and MUST also consider whether the status of any nodes has changed to "deprecated" and whether those nodes are implemented by the server.

6. Versioning of YANG instance data

Instance data sets [I-D.ietf-netmod-yang-instance-file-format] do not directly make use of the updated revision handling rules described in this document, as compatibility for instance data is undefined.

However, instance data specifies the content-schema of the data-set. This schema SHOULD make use of versioning using revision dates and/or revision labels for the individual YANG modules that comprise the schema or potentially for the entire schema itself (e.g., [I-D.rwilton-netmod-yang-packages]).

In this way, the versioning of a content-schema associated with an instance data set may help a client to determine whether the instance data could also be used in conjunction with other revisions of the YANG schema, or other revisions of the modules that define the schema.

7. Guidelines for using the YANG module update rules

The following text updates section 4.7 of [RFC8407] to revise the guidelines for updating YANG modules.

7.1. Guidelines for YANG module authors

All IETF YANG modules MUST include revision-label statements for all newly published YANG modules, and all newly published revisions of existing YANG modules. The revision-label MUST take the form of a YANG semantic version number [I-D.verdt-netmod-yang-semver].

NBC changes to YANG modules may cause problems to clients, who are consumers of YANG models, and hence YANG module authors are RECOMMENDED to minimize NBC changes and keep changes BC whenever possible.

When NBC changes are introduced, consideration should be given to the impact on clients and YANG module authors SHOULD try to mitigate that impact.

A "rev:nbc-changes" statement SHOULD be added only if there are NBC changes relative to the previous revision.

Removing old revision statements from a module's revision history could break import by revision, and hence it is RECOMMENDED to retain them. If all depencencies have been updated to not import specific revisions of a module, then the corresponding revision statements can be removed from that module. An alternative solution, if the

revision section is too long, would be remove, or curtail, the older description statements associated with the previous revisions.

The "rev:revision-or-derived" extension should be used in YANG module imports to indicate revision dependencies between modules in preference to the "revision-date" statement, which causes overly strict import dependencies and SHOULD NOT be used.

A module that includes submodules MUST use the "revision-date" statement to include specific submodule revisions. Changing a module's include statements to include different submodule revisions requires a new revision of the module.

7.1.1. Making non-backwards-compatible changes to a YANG module

There are various valid situations where a YANG module has to be modified in an NBC way. Here are the different ways in which this can be done:

- o NBC changes can be sometimes be done incrementally using the "deprecated" status to provide clients time to adapt to NBC changes.

- o NBC changes are done at once, i.e. without using "status" statements. Depending on the change, this may have a big impact on clients.

- o If the server can support multiple versions of the YANG module or of YANG packages(as specified in [I-D.rwilton-netmod-yang-packages]), and allows the client to select the version (as per [I-D.wilton-netmod-yang-ver-selection]), then NBC changes MAY be done without using "status" statements. Clients would be required to select the version which they support and the NBC change would have no impact on them

Here are some guidelines on how non-backwards-compatible changes can be made incrementally, with the assumption that deprecated nodes are implemented by the server, and obsolete nodes are not:

1. The changes should be made gradually, e.g. a data node's status SHOULD NOT be changed directly from "current" to "obsolete" (see Section 4.7 of [RFC8407]), instead the status SHOULD first be marked "deprecated" and then when support is removed its status MUST be changed to "obsolete". Instead of using the "obsolete" status, the data node MAY be removed from the model but this has the risk of breaking modules which import the modified module.

2. The new "status-description" extension statement SHOULD be used for nodes which are "obsolete" or "deprecated".

3. For status "deprecated", the "status-description" SHOULD also indicate until when support for the node is guaranteed (if known). If there is a replacement data node, rpc, action or notification for the deprecated node, this SHOULD be stated in the "status-description". The reason for deprecating the node can also be included in the "status-description" if it is deemed to be of potential interest to the user.

4. For status "obsolete", it is RECOMMENDED to keep the "status-description" information, from when the node had status "deprecated, which is still relevant.

5. When obsoleting or deprecating data nodes, the "deprecated" or "obsolete" status SHOULD be applied at the highest possible level in the data tree with an appropriate "status-description" statement. For clarity, the "status" statement SHOULD also be applied to all descendent data nodes, but the "status-description" statement does not need to be repeated if it does not introduce any additional information.

See Appendix A.1 for examples on how NBC changes can be made.

7.2. Versioning Considerations for Clients

Guidelines for clients of modules using the new module revision update procedure:

- Clients SHOULD be liberal when processing data received from a server. For example, the server may have increased the range of an operational node causing the client to receive a value which is outside the range of the YANG model revision it was coded against.

- Clients SHOULD monitor changes to published YANG modules through their revision history, and use appropriate tooling to understand the specific changes between module revision. In particular, clients SHOULD NOT migrate to NBC revisions of a module without understanding any potential impact of the specific NBC changes.

- Clients SHOULD plan to make changes to match published status changes. When a node's status changes from "current" to "deprecated", clients SHOULD plan to stop using that node in a timely fashion. When a node's status changes to "obsolete", clients MUST stop using that node.

8. Module Versioning Extension YANG Modules

YANG module with extension statements for annotating NBC changes, revision label, status description, and importing by version.

```
<CODE BEGINS> file "ietf-yang-revisions@2019-09-18.yang"
module ietf-yang-revisions {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-revisions";
  prefix rev;

  import ietf-yang-types {
    prefix yang;
    reference
      "XXXX [ietf-netmod-rfc6991-bis]: Common YANG Data Types";
  }

  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:   Reshad Rahman
               <mailto:rrahman@cisco.com>

     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>

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

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

     Author:   Jason Sterne
               <mailto:jason.sterne@nokia.com>";
  description
    "This YANG 1.1 module contains definitions and extensions to
     support updated YANG revision handling.

     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; 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 remove this note.
     // RFC Ed.: replace XXXX (inc above) with actual RFC number and
     // remove this note.

     revision 2019-09-18 {
       description
         "Initial version.";
       reference
         "XXXX: Updated YANG Module Revision Handling";
     }

     typedef revision-label {
       type string {
         length "1..255";
         pattern '[^\s@]+';
         pattern '\d{4}-\d{2}-\d{2}' {
           modifier invert-match;
         }
       }
       description
         "A label associated with a YANG revision.

          Excludes spaces and '@'.  MUST NOT match revision-date.

          Revision labels that classify as YANG semantic versions,
          as defined by the ietf-yang-semver:version typedef,
          MUST conform to the versioning behaviour defined in
          XXXX [verdt]: YANG Semantic Versioning.";
       reference
```

```
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.3, Revision label";
    }

    typedef revision-date-or-label {
      type union {
        type yang:revision-identifier;
        type revision-label;
      }
      description
        "Represents either a YANG revision date or a revision label";
    }

    extension nbc-changes {
      description
        "This statement is used to indicate YANG module revisions that
         contain non-backwards-compatible changes.

         Each 'revision' statement MAY have a single 'nbc-changes'
         substatement.

         If a revision of a YANG module contains changes, relative to
         the preceding revision in the revision history, that do not
         conform to the module update rules defined in RFC-XXX, then
         the 'nbc-changes' statement MUST be added as a substatement to
         the revision statement.

         Conversely, if a revision of a YANG module only contains
         changes, relative to the preceding revision in the revision
         history, that are classified as 'backwards-compatible' then
         the revision statement MUST NOT contain any 'nbc-changes'
         substatement.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.2, nbc-changes revision extension statement";
    }

    extension revision-label {
      argument revision-label;
      description
        "The revision label can be used to provide an additional
         versioning identifier associated with the revision.  E.g., one
         option for a versioning scheme that could be used is [TODO -
         Reference semver draft].

         The format of the revision-label argument MUST conform to the
         pattern defined for the revision-label typedef.
```

```
          Each 'revision' statement MAY have a single 'revision-label'
          substatement.

          Revision labels MUST be unique amongst all revisions of a
          module.";

        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 3.3, Revision label";
      }

      extension revision-or-derived {
        argument revision-date-or-label;
        description
          "Restricts the revision of the module that may be imported to
           one that matches or is derived from the specified
           revision-date or revision-nlabel.

           The argument value MUST conform to the
           'revision-date-or-label' defined type.

           Each 'import' statement MAY have one or more
           'revision-or-derived' substatements.  If specified multiple
           times, then any module revision that satifies at least one of
           the 'revision-or-derived' statements is an acceptable revision
           for import.

           An 'import' statement MUST NOT contain both a
           'revision-or-derived' extension statement and a
           'revision-date' statement.

           A particular revision of an imported module satisfies an
           import's 'revision-or-derived' extension statement if the
           imported module's revision history contains a revision
           statement with a matching revision date or revision label.

           The 'revision-or-derived' extension statement does not
           gaurantee that all module revisions that satisfy an import
           statement are necessarily compatible, it only gives an
           indication that the revisions are more likely to be
           compatible.";

        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 4, Import by derived revision";
      }

      extension status-description {
```

```
      argument description;

      description
        "Freeform text that describes why a given node has been
         deprecated or made obsolete.  E.g., the description could be
         used to give the reason for removal, or it could point to an
         alternative schema elements that can be used in lieu of the
         given node.

         Each 'status' statement MAY have a single 'status-description'
         substatement.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.4, YANG status description extension statement";
    }
  }
  <CODE ENDS>
```

YANG module with augmentations to YANG Library to revision labels

```
  <CODE BEGINS> file "ietf-yl-revisions@2019-09-18.yang"
  module ietf-yl-revisions {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yl-revisions";
    prefix yl-rev;

    import ietf-yang-revisions {
      prefix rev;
      reference
        "XXXX: Updated YANG Module Revision Handling";
    }

    import ietf-yang-library {
      prefix yanglib;
      reference "RFC 8525: YANG Libary";
    }

    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:   Reshad Rahman
                  <mailto:rrahman@cisco.com>

        Author:   Robert Wilton
                  <mailto:rwilton@cisco.com>

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

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

        Author:   Jason Sterne
                  <mailto:jason.sterne@nokia.com>";
     description
       "This module contains augmentations to YANG Library to add module
        level revision label and to provide an indication of how
        deprecated and obsolete nodes are handled by the server.

        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; 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 remove this note.
     // RFC Ed.: replace XXXX (including in the imports above) with
     // actual RFC number and remove this note.
     // RFC Ed.: please replace revision-label version with 1.0.0 and
     // remove this note.
     revision 2019-09-18 {
       rev:revision-label 0.1.0;
```

```
      description
        "Initial revision";
      reference
        "XXXX: Updated YANG Module Revision Handling";
    }

    augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
      description
        "Augmentation modules with a revision label";
      leaf revision-label {
        type rev:revision-label;
        description
          "The revision label associated with this module revision.
           The label MUST match the rev:label value in the specific
           revision of the module loaded in this module-set.";

        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 5.2.1, Advertising revision-label";
      }
    }

    augment "/yanglib:yang-library/yanglib:schema" {
      description
        "Augmentations to the ietf-yang-library module to indicate how
         deprecated and obsoleted nodes are handled for each datastore
         schema supported by the server.";

      leaf deprecated-nodes-implemented {
        type empty;
        description
          "If present, this leaf indicates that all schema nodes with a
           status 'deprecated' child statement are implemented
           equivalently as if they had status 'current', or otherwise
           deviations MUST be used to explicitly remove 'deprecated'
           nodes from the schema.  If this leaf is absent then the
           behavior is unspecified.";

        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 5.2.2, Reporting how deprecated and obsolete nodes
           are handled";
      }

      leaf obsolete-nodes-absent {
        type empty;
        description
          "If present, this leaf indicates that the server does not
```

```
            implement any status 'obsolete' nodes.  If this leaf is
            absent then the behaviour is unspecified.";

        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 5.2.2, Reporting how deprecated and obsolete nodes
           are handled";
      }
    }
  }
CODE ENDS>
```

9. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following individuals are (or have been) members of the design team and have worked on the YANG versioning project:

- Balazs Lengyel
- Benoit Claise
- Ebben Aries
- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu
- Reshad Rahman
- Rob Wilton

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update].

Discussons on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models. We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

10. Security Considerations

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

11. IANA Considerations

11.1. YANG Module Registrations

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

The ietf-yang-revisions module:

Name: ietf-yang-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-revisions

Prefix: rev

Reference: [RFCXXXX]

The ietf-yl-revisions module:

Name: ietf-yl-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yl-revisions

Prefix: yl-rev

Reference: [RFCXXXX]

12. References

12.1. Normative References

[I-D.ietf-netmod-rfc6991-bis]
Schoenwaelder, J., "Common YANG Data Types", draft-ietf-netmod-rfc6991-bis-02 (work in progress), November 2019.

[I-D.verdt-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-verdt-netmod-yang-semver-01 (work in progress), October 2019.

[I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 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>.

[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>.

[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>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

12.2. Informative References

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", draft-clacla-netmod-yang-model-update-06 (work in progress), July 2018.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-08 (work in progress), March 2020.

[I-D.rwilton-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Packages", draft-rwilton-netmod-yang-packages-03 (work in progress), February 2020.

[I-D.verdt-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-verdt-netmod-yang-solutions-03 (work in progress), February 2020.

[I-D.wilton-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Schema Selection", draft-wilton-netmod-yang-ver-selection-02 (work in progress), February 2020.

[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>.

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

Appendix A. Appendix

A.1. Examples of guidelines for making NBC changes to a YANG module

Examples of NBC changes include:

- o Deleting a data node, or changing it to status obsolete.
- o Changing the name, type, or units of a data node.
- o Modifying the description in a way that changes the semantic meaning of the data node.
- o Any changes that change or reduce the allowed value set of the data node, either through changes in the type definition, or the addition or changes to "must" statements, or changes in the description.
- o Adding or modifying "when" statements that reduce when the data node is available in the schema.
- o Making the statement conditional on if-feature.

The following sections give guidance for how some of these NBC changes could be made to a YANG module:

A.1.1. Removing a data node

Removing a leaf or container from the data tree, e.g. because support for the corresponding feature is being removed:

1. The node's status is changed to "deprecated" and it is supported for at least one year. This is a BC change.

2. When the node is not available anymore, its status is changed to "obsolete" and the "status-description" updated, this is an NBC change. The "status-description" is used to explain why the node is not available anymore.

If the server can support NBC versions of the YANG module simultaneously using version selection, then the changes can be done immediately:

1. The new revision of the YANG module has the node's status changed to "obsolete" and the "status-description" updated, this is an NBC change.

2. Clients which require the data node select the older module revision

A.1.2. Changing the type of a leaf node

Changing the type of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed to a string:

1. The status of node "vpn-id" is changed to "deprecated" and the node should be available for at least one year. This is a BC change.

2. A new node, e.g. "vpn-name", of type string is added to the same location as the existing node "vpn-id". This new node has status "current" and its description explains that it is replacing node "vpn-id".

3. During the period of time where both nodes are available, how the server behaves when either node is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server may prevent the new node from being set if the old node is already set (and vice-versa). The new node may have a when statement to achieve this. The old node must not have a when statement since this would be an NBC change, but the server could reject the old node from being set if the new node is already set.

    2. If the new node is set and a client does a get or get-config operation on the old node, the server could map the value. For example, if the new node "vpn-name" has value "123" then

the server could return integer value 123 for the old node "vpn-id". However, if the value can not be mapped, we need a way of returning "unsupported" TBD.

4. When node "vpn-id" is not available anymore, its status is changed to "obsolete" and the "status-description" is updated. This is an NBC change.

If the server can support NBC versions of the YANG module simultaneously using version selection, then the changes can be done immediately:

1. In the new revision of the YANG module, the status of node "vpn-id" is changed to "obsolete". This is an NBC change.

2. New node "vpn-name" is added to the same location as described above.

3. Clients which require the data node select the older module revision

4. A server should not map between the nodes "vpn-id" and "vpn-name", i.e. if a client creates a data instance with "vpn-name" then that data instance should not be visible to a client using a module revision which has "vpn-id" (and vice-versa).

A.1.3. Reducing the range of a leaf node

Reducing the range of values of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed from type uint32 to type uint16:

1. If all values which are being removed were never supported, e.g. if a vpn-id of 65536 or higher was never accepted, this is a BC change for the functionality (no functionality change). Even if it is an NBC change for the YANG model, there should be no impact for clients using that YANG model.

2. If one or more values being removed was previously supported, e.g. if a vpn-id of 65536 was accepted previously, this is an NBC change for the YANG model. Clients using the old YANG model will be impacted, so a change of this nature should be done carefully, e.g. by using the steps described in Appendix A.1.2

A.1.4. Changing the key of a list

Changing the key of a list has a big impact to the client. For example, consider a "sessions" list which has a key "interface" and

there is a need to change the key to "dest-address", such a change can be done in steps:

1. The status of list "sessions" is changed to "deprecated" and the list should be available for at least one year. This is a BC change.

2. A new list is created in the same location with the same data but with "dest-address" as key. Finding an appropriate name for the new list can be tricky especially if the name of the existing list was perfect. In this case the new list is called "sessions-address", has status "current" and its description should explain that it is replacing list "session".

3. During the period of time where both lists are available, how the server behaves when either list is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server could prevent the new list from being set if the old list already has entries (and vice-versa).

    2. If the new list is set and a client does a get or get-config operation on the old list, the server could map the entries. However if the new list has entries which would lead to duplicate keys in the old list, the mapping can not be done.

4. When list "sessions" is not available anymore, its status is changed to "obsolete" and the "status-description" is updated. This is an NBC change.

If the server can support NBC versions of the YANG module simultaneously using version selection, then the changes can be done immediately:

1. The new revision of the YANG module has the list "sessions" modified to have "dest-address" as key, this is an NBC change.

2. Clients which require the previous functionality select the older module revision

A.1.5. Renaming a node

A leaf-node or a container may be renamed, either due to a spelling error in the previous name or because of a better name. For example a node "ip-adress" could be renamed to "ip-address":

1. The status of the existing node "ip-adress" is changed to "deprecated" and the node should be available for at least one year. This is a BC change.

2. The new node "ip-address" is added to the same location as the existing node "ip-adress". This new node has status "current" and its description should explain that it is replacing node "ip-adress".

3. During the period of time where both nodes are available, how the server behaves when either node is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server could prevent the new node from being set if the old node is already set (and vice-versa). The new node could have a when statement to achieve this. The old node must not have a when statement since this would be an NBC change, but the server could reject the old node from being set if the new node is already set.

    2. If the new node is set and a client does a get or get-config operation on the old node, the server could use the value of the new node. For example, if the new node "ip-address" has value X then the server may return value X for the old node "ip-adress".

4. When node "ip-adress" is not available anymore, its status is changed to "obsolete" and the "status-description" is updated. This is an NBC change.

If the server can support NBC versions of the YANG module simultaneously using version selection, then the changes can be done immediately:

1. The new revision of the YANG module has the node with the new name replacing the node with the old name, this is an NBC change.

2. Clients which require the previous node name select the older module revision

A.1.6. Changing a default value

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

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Balazs Lengyel
Ericsson
Magyar Tudosok Korutja
1117 Budapest
Hungary

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

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

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

Email: kd6913@att.com