Network Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: January 14, 2021

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

July 13, 2020

YANG Semantic Versioning
draft-ietf-netmod-yang-semver-01

Abstract

This document specifies a scheme and guidelines for applying a modified set of semantic versioning rules to revisions of YANG modules. Additionally, this document defines a revision-label for this modified semver scheme.

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 14, 2021.

Table of Contents

1. Introduction

[I-D.ietf-netmod-yang-module-versioning] puts forth a number of concepts relating to modified rules for updating modules, a means to signal when a new revision of a module has non-backwards-compatible (NBC) changes compared to its previous revision, and a versioning scheme that uses the revision history as a lineage for determining from where a specific revision of a YANG module is derived. Additionally, section 3.3 of [I-D.ietf-netmod-yang-module-versioning] defines a revision label which can be used as an overlay or alias to provide additional context or an additional way to refer to a specific revision.

This document defines a revision-label scheme that uses modified [semver] rules for YANG artifacts (i.e., YANG modules and YANG packages [I-D.ietf-netmod-yang-packages]) as well as the revision label definition for using this scheme. The goal of this is to add a human readable version label that provides compatibility information for the YANG artifact without one needing to compare or parse its body. The label and rules defined herein represent the RECOMMENDED revision label scheme for IETF YANG artifacts.

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.

Additionally, this document uses the following terminology:

- YANG artifact: YANG modules, YANG packages [I-D.ietf-netmod-yang-packages], and YANG schema elements are examples of YANG artifacts for the purposes of this document.

3. YANG Semantic Versioning

This section defines YANG Semantic Versioning, explains how it is used with YANG artifacts, and the rules associated with changing an artifact's semantic version number when its contents are updated.

3.1. YANG Semantic Versioning Pattern

YANG artifacts that employ semantic versioning as defined in this document MUST use a version string (e.g., in revision-label or as a package version) that corresponds to the following pattern: X.Y.Z_COMPAT. Where:

- X, Y and Z are mandatory non-negative integers that are each less than 2147483647 (i.e., the maximum signed 32-bit integer value) and MUST NOT contain leading zeroes

- The '.' is a literal period (ASCII character 0x2e)

- The '_' is an optional single literal underscore (ASCII character 0x5f) and MUST only present if the following COMPAT element is included

- COMPAT, if it is specified, MUST be either the literal string "compatible" or the literal string "non_compatible"

Additionally, [semver] defines two specific types of metadata that may be appended to a semantic version string. Pre-release metadata MAY be appended to a semver string after a trailing '-' character. Build metadata MAY be appended after a trailing '+' character. If both pre-release and build metadata are present, then build metadata MUST follow pre-release metadata. While build metadata MUST be ignored by YANG semver parsers, pre-release metadata MUST be used during module development and MUST be considered base on Section 5. Both pre-release and build metadata are allowed in order to support all of the [semver] rules. Thus, a version lineage that follows strict [semver] rules is allowed for a YANG artifact.

To signal the use of this versioning scheme, modules MUST set the revision-label-scheme extension as defined in [I-D.ietf-netmod-yang-module-versioning] to the identity "yang-semver". That identity value is defined in the ietf-yang-semver module below.

Additionally, this ietf-yang-semver module defines a typedef that formally specifies the syntax of the YANG semver version string.

3.2. Semantic Versioning Scheme for YANG Artifacts

This document defines the YANG semantic versioning scheme that is used for YANG artifacts that employ the YANG semver label. The versioning scheme has the following properties:

The YANG semantic versioning scheme is extended from version 2.0.0 of the semantic versioning scheme defined at semver.org [semver] to cover the additional requirements for the management of YANG artifact lifecyles that cannot be addressed using the semver.org 2.0.0 versioning scheme alone.

Unlike the [semver] versioning scheme, the YANG semantic versioning scheme supports limited updates to older versions of YANG artifacts, to allow for bug fixes and enhancements to artifact versions that are not the latest. However, it does not provide for the unlimited branching and updating of older revisions which are documented by the general rules in [I-D.ietf-netmod-yang-module-versioning].

YANG artifacts that follow the [semver] versioning scheme are fully compatible with implementations that understand the YANG semantic versioning scheme defined in this document.

If updates are always restricted to the latest revision of the artifact only, then the version numbers used by the YANG semantic

versioning scheme are exactly the same as those defined by the [semver] versioning scheme.

Every YANG module versioned using the YANG semantic versioning scheme specifies the module's semantic version number as the argument to the 'rev:revision-label' statement.

Because the rules put forth in [I-D.ietf-netmod-yang-module-versioning] are designed to work well with existing versions of YANG and allow for artifact authors to migrate to this scheme, it is not expected that all revisions of a given YANG artifact will have a semantic version label. For example, the first revision of a module may have been produced before this scheme was available.

YANG packages that make use of this semantic versioning scheme will have their semantic version as the value of the "revision_label" property.

As stated above, the YANG semver version number is expressed as a string of the form: 'X.Y.Z_COMPAT'; where X, Y, and Z each represent non-negative integers smaller than 2147483647 without leading zeroes, and _COMPAT represents an optional suffix of either "_compatible" or "_non_compatible".

- 'X' is the MAJOR version. Changes in the major version number indicate changes that are non-backwards-compatible to versions with a lower major version number.

- 'Y' is the MINOR version. Changes in the minor version number indicate changes that are backwards-compatible to versions with the same major version number, but a lower minor version number and no patch "_compatible" or "_non_compatible" modifier.

- 'Z_COMPAT' is the PATCH version and modifier. Changes in the patch version number can indicate editorial, backwards-compatible, or non-backwards-compatible changes relative to versions with the same major and minor version numbers, but lower patch version number, depending on what form modifier "_COMPAT" takes:

  * If the modifier string is absent, the change represents an editorial change. An editorial change is defined to be a change in the YANG artifact's content that does not affect the semantic meaning or functionality provided by the artifact in any way. An example is correcting a spelling mistake in the description of a leaf within a YANG module. Note: restructuring how a module uses, or does not use, submodules is treated as an editorial level change on the condition that

there is no change in the module's semantic behavior due to the restructuring.

* If, however, the modifier string is present, the meaning is described below:

* "_compatible" - the change represents a backwards-compatible change

* "_non_compatible" - the change represents a non-backwards-compatible change

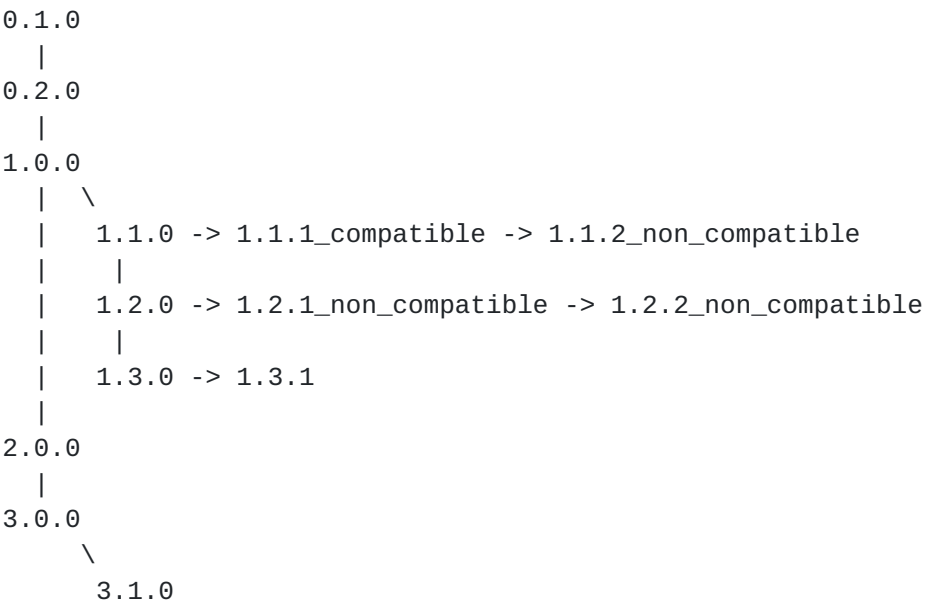
The YANG artifact name and YANG semantic version number uniquely identify a revision of said artifact. There MUST NOT be multiple instances of a YANG artifact definition with the same name and YANG semantic version number but different content (and in the case of modules, different revision dates).

There MUST NOT be multiple versions of a YANG artifact that have the same MAJOR, MINOR and PATCH version numbers, but different patch modifier strings. E.g., artifact version "1.2.3_non_compatible" MUST NOT be defined if artifact version "1.2.3" has already been defined.

3.2.1. Examples for YANG semantic version numbers

The following diagram and explanation illustrates how YANG semantic version numbers work.

Example YANG semantic version numbers for an example artifact:



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

0.1.0 - first beta module version

0.2.0 - second beta module version (with NBC changes)

1.0.0 - first release (may have NBC changes from 0.2.0)

1.1.0 - added new functionality, leaf "foo" (BC)

1.2.0 - added new functionality, leaf "baz" (BC)

1.3.0 - improve existing functionality, added leaf "foo-64" (BC)

1.3.1 - improve description wording for "foo-64" (Editorial)

1.1.1_compatible - backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0 (BC)

2.0.0 - change existing model for performance reasons, e.g. re-key list (NBC)

1.1.2_non_compatible - NBC point bug fix, not required in 2.0.0 due to model changes (NBC)

3.0.0 - NBC bugfix, rename "baz" to "bar"; also add new BC leaf "wibble"; (NBC)

1.2.1_non_compatible - backport NBC fix, changing "baz" to "bar"

1.2.2_non_compatible - backport "wibble". This is a BC change but "non_compatible" modifier is sticky.

3.1.0 - introduce new leaf "wobble" (BC)

The partial ordering relationships based on the semantic versioning numbers can be defined as follows:

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 2.0.0 < 3.0.0 < 3.1.0

1.0.0 < 1.1.0 < 1.1.1_compatible < 1.1.2_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.2.1_non_compatible < 1.2.2_non_compatible

There is no ordering relationship between 1.1.1_non_compatible and either 1.2.0 or 1.2.1_non_compatible, except that they share the common ancestor of 1.1.0.

Looking at the version number alone, the module definition in 2.0.0 does not necessarily contain the contents of 1.3.0. However, the module revision history in 2.0.0 may well indicate that it was edited from module version 1.3.0.

3.3. YANG Semantic Version Update Rules

When a new revision of an artifact is produced, then the following rules define how the YANG semantic version number for the new artifact revision is calculated, based on the changes between the two artifact revisions, and the YANG semantic version number of the base artifact revision from which the changes are derived:

1. If an artifact is being updated in a non-backwards-compatible way, then the artifact version "X.Y.Z[_compatible|_non_compatible]" MUST be updated to "X+1.0.0" unless that artifact version has already been defined with different content, in which case the artifact version "X.Y.Z+1_non_compatible" MUST be used instead.

2. Under some circumstances (e.g., to avoid adding a "_compatible" modifier) an artifact author MAY also update the MAJOR version when the only changes are backwards-compatible. This is where tooling is important to highlight all changes. Because, while avoiding the "_compatible" and "_non_compatible" modifiers have a clear advantage, bumping a MAJOR version when changes are entirely backwards-compatible may confuse end users.

3. If an artifact is being updated in a backwards-compatible way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version MUST be updated to "X.Y+1.0", unless that artifact version has already been defined with different content, when the artifact version MUST be updated to "X.Y.Z+1_compatible"" instead.

   ii "X.Y.Z_compatible" - the artifact version MUST be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version MUST be updated to "X.Y.Z+1_non_compatible".

4. If an artifact is being updated in an editorial way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version MUST be updated to "X.Y.Z+1"

   ii "X.Y.Z_compatible" - the artifact version MUST be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version MUST be updated to "X.Y.Z+1_non_compatible".

5. YANG artifact semantic version numbers beginning with 0, i.e "0.X.Y" are regarded as beta definitions and need not follow the rules above. Either the MINOR or PATCH version numbers may be updated, regardless of whether the changes are non-backwards-compatible, backwards-compatible, or editorial. See Section 5 for more details on using this notation during module development.

3.4. Examples of the YANG Semver Label

3.4.1. Example Module Using YANG Semver

Below is a sample YANG module that uses the YANG semver revision label based on the rules defined in this document.

```
module example-versioned-module {
  yang-version 1.1;
  namespace "urn:example:versioned:module";
  prefix "exvermod";
  rev:revision-label-scheme "yangver:yang-semver";

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

  description
    "to be completed";

  revision 2018-02-28 {
    description "Added leaf 'wobble'";
    rev:revision-label "3.1.0";
  }

  revision 2017-12-31 {
    description "Rename 'baz' to 'bar', added leaf 'wibble'";
    rev:revision-label "3.0.0";
    rev:nbc-changes;
```

```
        }

        revision 2017-10-30 {
          description "Change the module structure";
          rev:revision-label "2.0.0";
          rev:nbc-changes;
        }

        revision 2017-08-30 {
          description "Clarified description of 'foo-64' leaf";
          rev:revision-label "1.3.1";
        }

        revision 2017-07-30 {
          description "Added leaf foo-64";
          rev:revision-label "1.3.0";
        }

        revision 2017-04-20 {
          description "Add new functionality, leaf 'baz'";
          rev:revision-label "1.2.0";
        }

        revision 2017-04-03 {
          description "Add new functionality, leaf 'foo'";
          rev:revision-label "1.1.0";
        }

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

        // Note: semver rules do not apply to 0.X.Y labels.

        revision 2017-01-30 {
          description "NBC changes to initial revision";
          semver:module-version "0.2.0";
        }

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

        //YANG module definition starts here
```

3.4.2. Example of Package Using YANG Semver

Below is an example YANG package that uses the semver revision label based on the rules defined in this document.

```
   {
     "ietf-yang-instance-data:instance-data-set": {
       "name": "example-yang-pkg",
       "target-ptr": "TBD",
       "timestamp": "2018-09-06T17:00:00Z",
       "description": "Example IETF package definition",
       "content-data": {
         "ietf-yang-package:yang-package": {
           "name": "example-yang-pkg",
           "version": "1.3.1",
           ...
 }
```

4. Import Module by Semantic Version

[I-D.ietf-netmod-yang-module-versioning] allows for imports to be done based on a module or a derived revision of a module. The rev:revision-or-derived statement can specify either a revision date or a revision label. When importing by semver, the YANG semver revision label value MAY be used as an argument to rev:revision-or-derived. In so, any module which has that semver label as its latest revision label or has that label in its revision history can be used to satisfy the import requirement. For example:

```
       import example-module {
         rev:revision-or-derived "3.0.0";
       }
```

Note: the import lookup does not stop when a non-backward-compatible change is encountered. That is, if module B imports a module A at or derived from version 2.0.0, resolving that import will pass through a revision of module A with version 2.1.0_non_compatible in order to determine if the present instance of module A derives from 2.0.0.

5. Guidelines for Using Semver During Module Development

This section and the IETF-specific sub-section below provides YANG semver-specific guidelines to consider when developing new YANG modules. As such this section updates [RFC8407].

Development of a brand new YANG module outside of the IETF that uses YANG semver as its revision-label scheme SHOULD begin with a 0 for the MAJOR version component. This allows the module to disregard

strict semver rules with respect to non-backwards-compatible changes during its initial development. However, module developers MAY choose to use the semver pre-release syntax instead with a 1 for the MAJOR version component. For example, an initial module revision-label might be either 0.0.1 or 1.0.0-alpha.1. If the authors choose to use the 0 MAJOR version component scheme, they MAY switch to the pre-release scheme with a MAJOR version component of 1 when the module is nearing initial release (e.g., a module's revision label may transition from 0.3.0 to 1.0.0-beta.1 to indicate it is more mature and ready for testing).

When using pre-release notation, the format MUST include at least one alphabetic component and MUST end with a '.' and then one or more digits. These alphanumeric components will be used when deciding pre-release precedence. The following are examples of valid pre-release versions

1.0.0-alpha.1

1.0.0-alpha.3

2.1.0-beta.42

3.0.0-202007.rc.1

When developing a new revision of an existing module using the YANG semver revision-label scheme, the intended target semver version MUST be used along with pre-release notation. For example, if a released module which has a current revision-label of 1.0.0 is being modified with the intent to make non-backwards-compatible changes, the first development MAJOR version component must be 2 with some pre-release notation such as -alpha.1, making the version 2.0.0-alpha.1. That said, every publicly available release of a module MUST have a unique YANG semver revision-label (where a publicly available release is one that could be implemented by a vendor or consumed by an end user). Therefore, it may be prudent to include the year or year and month development began (e.g., 2.0.0-201907-alpha.1). As a module undergoes development, it is possible that the original intent changes. For example, a 1.0.0 version of a module that was destined to become 2.0.0 after a development cycle may have had a scope change such that the final version has no non-backwards-compatible changes and becomes 1.1.0 instead. This change is acceptable to make during the development phase so long as pre-release notation is present in both versions (e.g., 2.0.0-alpha.3 becomes 1.1.0-alpha.4). However, on the next development cycle (after 1.1.0 is released), if again the new target release is 2.0.0, new pre-release components must be used such that every revision-label for a given module MUST be unique throughout its entire lifecycle (e.g., the first pre-release version

might be 2.0.0-202005-alpha.1 if keeping the same year and month notation mentioned above).

5.1. Pre-release Version Precedence

[TODO: Describe precedence considering there could be changes during development and parallel development tracks.]

5.2. YANG Semver in IETF Modules

Net new module development within the IETF SHOULD begin with the 0 MAJOR number scheme as described above. When revising an existing IETF module, the revision-label MUST use the target (i.e., intended) MAJOR and MINOR version components with a 0 patch version component. If the intended ratified release will be non-backward-compatible with the current ratified release, the MINOR version component MUST be 0.

All IETF modules in development MUST use the whole document name as a pre-release version string, including the current document revision. For example, if a module which is currently released at version 1.0.0 is being revised to include non-backwards-compatible changes in draft-user-netmod-foo, its development revision-labels MUST include 2.0.0-draft-user-netmod-foo followed by the document's revision (e.g., 2.0.0-draft-user-netmod-foo-02). This will ensure each pre-release version is unique across the lifecycle of the module. Even when using the 0 MAJOR version for initial module development (where MINOR and PATCH can change), appending the draft name as a pre-release component helps to ensure uniqueness when there are perhaps multiple, parallel efforts creating the same module.

If a module is being revised and the original module never had a revision-label (i.e., you wish to start using YANG semver in future module revisions), choose a semver value that makes the most sense based on the module's history. For example, if a module started out in the pre-NMDA ([RFC8342]) world, and then had NMDA support added without removing any legacy "state" branches -- and you are looking to add additional new features -- a sensible choice for the target YANG semver would be 1.2.0 (since 1.0.0 would have been the initial, pre-NMDA release, and 1.1.0 would have been the NMDA revision).

See Appendix A for a detailed example of IETF pre-release versions.

6. YANG Module

This YANG module contains the typedef for the YANG semantic version.

<CODE BEGINS> file "ietf-yang-semver@2019-09-06.yang"

```
  module ietf-yang-semver {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-semver";
    prefix yangver;
    rev:revision-label-scheme "yang-semver";

    import ietf-yang-revisions {
      prefix rev;
    }

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

       Author:   Joe Clarke
                 <mailto:jclarke@cisco.com>";
    description
      "This module provides type and grouping definitions for YANG
       packages.

       Copyright (c) 2020 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.";

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

    revision 2020-06-30 {
      rev:revision-label "1.0.0-draft-ietf-netmod-yang-semver-01";
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }
```

```
    /*
     * Identities
     */

    identity yang-semver {
      base rev:revision-label-scheme-base-identity;
      description
        "The revision-label scheme corresponds to the YANG semver scheme
         which is defined by the pattern in the 'version' typedef below.
         The rules governing this revision-label scheme are defined in the
         reference for this identity.";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }

    /*
     * Typedefs
     */

    typedef version {
      type string {
        pattern '\d+[.]\d+[.]\d+(_(non_)?compatible)?(-[\w\d.]+)?([+][\w\d\.]
+)?';
      }
      description
        "Represents a YANG semantic version number.  The rules governing the
         use of this revision label scheme are defined in the reference for
         this typedef.";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }
  }
    <CODE ENDS>
```

7. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The design team consists of the following members whom 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 based on their own [openconfigsemver]. We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

8. Security Considerations

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

9. IANA Considerations

None.

10. References

10.1. Normative References

[I-D.ietf-netmod-yang-module-versioning]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "Updated YANG Module Revision Handling", draft-ietf-netmod-yang-module-versioning-00 (work in progress), March 2020.

[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>.

[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>.

10.2. 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-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Packages", draft-ietf-netmod-yang-packages-00 (work in progress), March 2020.

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

[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>.

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

Appendix A. Example IETF Module Development

Assume a new YANG module is being developed in the netmod working group in the IETF. Initially, this module is being developed in an individual internet draft, draft-jdoe-netmod-example-module. The following represents the initial version tree (i.e., value of revision-label) of the module as it's being initially developed.

Version lineage for initial module development:

```
      0.0.1-draft-jdoe-netmod-example-module-00
        |
      0.1.0-draft-jdoe-netmod-example-module-01
        |
      0.2.0-draft-jdoe-netmod-example-module-02
        |
      0.2.1-draft-jdoe-netmod-example-module-03
```

At this point, development stabilizes, and the workgroup adopts the draft. Thus now the draft becomes draft-ietf-netmod-example-module. The initial pre-release lineage continues as follows.

Continued version lineage after adoption:

```
    1.0.0-draft-ietf-netmod-example-module-00
      |
    1.0.0-draft-ietf-netmod-example-module-01
      |
    1.0.0-draft-ietf-netmod-example-module-02
```

At this point, the draft is ratified and becomes RFC12345 and the YANG module version number becomes 1.0.0.

A time later, the module needs to be revised to add additional capabilities. Development will be done in a backwards-compatible way. Two new individual drafts are proposed to go about adding the capabilities in different ways: draft-jdoe-netmod-exmod-enhancements and draft-jadoe-netmod-exmod-changes. These are initially developed in parallel with the following versions.

Parallel development for next module revision:

```
   1.1.0-draft-jdoe-netmod-exmod-enhancements-00 || 1.1.0-draft-jadoe-
netmod-exmod-changes-00
      |                                                  |
   1.1.0-draft-jdoe-netmod-exmod-enhancements-01 || 1.1.0-draft-jadoe-
netmod-exmod-changes-01
```

At this point, the WG decides to merge some aspects of both and adopt the work in jadoe's draft as draft-ietf-netmod-exmod-changes. A single version lineage continues.

```
1.1.0-draft-ietf-netmod-exmod-changes-00
  |
1.1.0-draft-ietf-netmod-exmod-changes-01
  |
1.1.0-draft-ietf-netmod-exmod-changes-02
  |
1.1.0-draft-ietf-netmod-exmod-changes-03
```

The draft is ratified, and the new module version becomes 1.1.0.

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 (editor)
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