

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 13, 2020

B. Claise
J. Clarke, Ed.
R. Rahman
R. Wilton, Ed.
Cisco Systems, Inc.
B. Lengyel
Ericsson
J. Sterne
Nokia
K. D'Souza
AT&T
October 11, 2019

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

Abstract

This document specifies a scheme 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 based on the specification in draft-verdt-netmod-yang-module-versioning.

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 13, 2020.

Copyright Notice

Copyright (c) 2019 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	2
2. Terminology and Conventions	3
3. YANG Semantic Versioning	3
3.1. YANG Semantic Versioning Pattern	3
3.2. Semantic Versioning Scheme for YANG Artifacts	4
3.2.1. Examples for YANG semantic version numbers	6
3.3. YANG Semantic Version Update Rules	7
3.4. Examples of the YANG Semver Label	8
3.4.1. Example Module Using YANG Semver	8
3.4.2. Example of Package Using YANG Semver	10
4. Import Module by Semantic Version	10
5. YANG Module	11
6. Contributors	12
7. Security Considerations	13
8. IANA Considerations	13
9. References	13
9.1. Normative References	13
9.2. Informative References	14
Authors' Addresses	14

1. Introduction

[I-D.verdt-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.verdt-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 labeling scheme that uses modified [semver] rules for YANG artifacts (i.e., YANG modules and YANG packages [I-D.rwilton-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:

- o YANG artifact: YANG modules, YANG packages [I-D.rwilton-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 a artifact's semantic version number when its contents are updated.

3.1. YANG Semantic Versioning Pattern

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

- o X, Y and Z are mandatory non-negative integers that are each less than 32768 and MUST NOT contain leading zeroes
- o The '.' is a literal period (ASCII character 0x2e)
- o v is an optional single character modifier that MUST be either 'm' or 'M' if it is specified

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. These optional elements MUST be ignored by YANG semver parsers, but 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.

Other version schemes MUST NOT use version strings that match this same pattern. For example, they may choose to use leading characters to distinguish themselves from YANG semver.

A YANG module is defined in this document which contains single typedef that formally specifies this version pattern.

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 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 lifecycles 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.verdt-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.verdt-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.Zv'; where X, Y, and Z each represent non-negative integers smaller than 32768 without leading zeroes, and v represents an optional single character suffix: 'm' or 'M'.

- o '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.
- o '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 'm' or 'M' modifier.
- o 'Zv' 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 'v' takes:
 - * If the modifier letter 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 letter is present, the meaning is described below:
 - * 'm' - the change represents a backwards-compatible change
 - * 'M' - the change represents a non-backwards-compatible change

The YANG artifact name and YANG semantic version number uniquely identifies a revision of said artifact. There MUST NOT be multiple instances of a YANG artifact definition with the same name and YANG

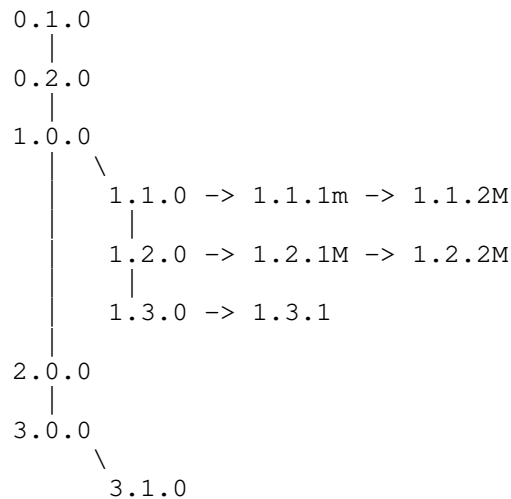
semantic version number but different content (and in the case of modules, different revision dates).

There MUST NOT be multiple versions of a YANG artifact that have the same MAJOR, MINOR and PATCH version numbers, but different patch modifier letters. E.g., artifact version "1.2.3M" 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)
- 1.1.1m - 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)
- 1.1.2M - 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)
- 1.2.1M - backport NBC fix, changing "baz" to "bar"
- 1.2.2M - backport "wibble". This is a BC change but "M" 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 < 1.1.1m < 1.1.2M
- 1.0.0 < 1.1.0 < 1.2.0 < 1.2.1M < 1.2.2M

There is no ordering relationship between 1.1.1M and either 1.2.0 or 1.2.1M, 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[m|M]" 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+1M" MUST be used instead.
2. 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+1m" instead.
 - ii "X.Y.Zm" - the artifact version MUST be updated to "X.Y.Z+1m".
 - iii "X.Y.ZM" - the artifact version MUST be updated to "X.Y.Z+1M".
3. 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.Zm" - the artifact version MUST be updated to "X.Y.Z+1m".
 - iii "X.Y.ZM" - the artifact version MUST be updated to "X.Y.Z+1M".
4. 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.

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 yang-module-name {
```

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

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

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.verdt-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 2.1.0M in order to determine if the present instance of module A derives from 2.0.0.

5. 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;

  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) 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.";

    // 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 2019-09-06 {
```

```
    description
        "Initial revision";
    reference
        "RFC XXXX: YANG Semantic Versioning.";
}

/*
 * Typedefs
 */

typedef version {
    type string {
        pattern '\d+[\.]\d+[\.]\d+[mM]?(-[\w\d.]+)?(#[\w\d.]+)?';
    }
    description
        "Represents a YANG semantic version number. Note:
        additional rules apply to the dot-separated numeric identifiers
        which are spelled out in the reference for this typedef.";
    reference
        "RFC XXXX: YANG Semantic Versioning.";
}
}
<CODE ENDS>
```

6. 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:

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

- o Reshad Rahman
- o Rob Wilton

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

Discussions 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.

7. Security Considerations

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

8. IANA Considerations

None.

9. References

9.1. Normative References

- [I-D.verdt-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-verdt-netmod-yang-module-versioning-00 (work in progress), July 2019.
- [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>>.
- [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>>.
- [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>>.

9.2. 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-04 (work in progress), August 2019.
- [I-D.rwilton-netmod-yang-packages]
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-01 (work in progress), March 2019.
- [openconfigsemver]
"Semantic Versioning for Openconfig Models",
<<http://www.openconfig.net/docs/semver/>>.
- [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 (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