Network Working Group
Internet-Draft
Updates: 8407, 8525, 7950 (if approved)
Intended status: Standards Track
Expires: 2 February 2026

J. Clarke, Ed.
R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman
Equinix
B. Lengyel
Ericsson
J. Sterne
Nokia
B. Claise
Huawei

1 August 2025

YANG Semantic Versioning
draft-ietf-netmod-yang-semver-22

Abstract

This document specifies a YANG extension along with guidelines for applying an extended set of semantic versioning rules to revisions of YANG artifacts (e.g., modules and packages). Additionally, this document defines a YANG extension for controlling module imports based on these modified semantic versioning rules. This document updates RFCs 7950, 8407, and 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 2 February 2026.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-netmod-yang-module-versioning] puts forth new concepts relating to modified rules for updating YANG modules and submodules, a means to signal when a new revision of a module or submodule has non-backwards-compatible (NBC) changes compared to its previous revision, and extends the YANG import statement with a minimum revision suggestion to help document inter-module dependencies.

This document defines a YANG extension that tags a YANG artifact (i.e., YANG modules, YANG submodules, and YANG packages [I-D.ietf-netmod-yang-packages]) with a version identifier that adheres to extended semantic versioning rules [SemVer]. The goal being to add a human readable version identifier that provides compatibility information for the YANG artifact without needing to compare or parse its body. The version identifier and rules defined herein represent the recommended approach to apply versioning to IETF YANG artifacts. This document defines augmentations to ietf-yang-library to reflect the version of YANG modules within the module-set data.

Note that a specific revision of the SemVer 2.0.0 specification is referenced here (from June 19, 2020) to provide an immutable version. This is because the 2.0.0 version of the specification has changed over time without any change to the semantic version itself.

2. Examples of How Versioning Is Applied To YANG Module Revisions

The following diagram illustrates how the branched revision history and the YANG Semver version extension statement could be used:

Example YANG module with branched revision history.

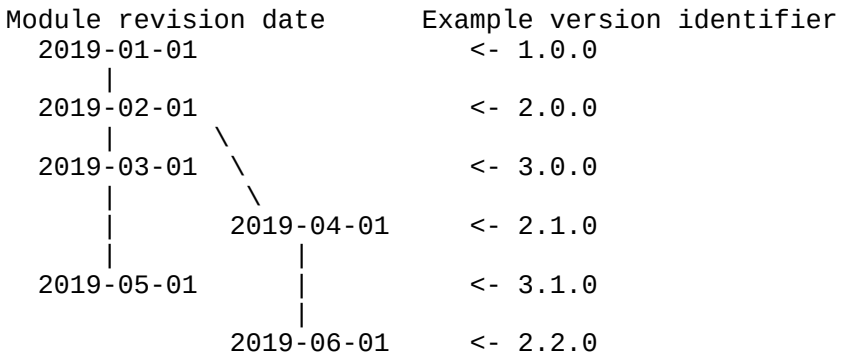


Figure 1

The tree diagram above illustrates how an example module's revision history might evolve, over time.

3. 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 submodules, and YANG packages [I-D.ietf-netmod-yang-packages] are examples of YANG artifacts for the purposes of this document.

* SemVer: A version string that corresponds to the rules defined in [SemVer]. This specific camel-case notation is the one used by the SemVer 2.0.0 website and used within this document to distinguish between SemVer 2.0.0 and YANG Semver.

* YANG Semver: A version identifier that is consistent with the extended set of semantic versioning rules, based on [SemVer], defined within this document.

4. YANG Semantic Versioning

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

4.1. Relationship Between SemVer and YANG Semver

[SemVer] is completely compatible with YANG Semver in that a SemVer semantic version number is legal according to the YANG Semver rules (though the inverse is not necessarily true). YANG Semver is a superset of the SemVer rules, and allows for limited branching within YANG artifacts. If no branching occurs within a YANG artifact (i.e., you do not use the compatibility modifiers described below), the YANG Semver version label will appear as a SemVer version number.

4.2. YANG Semantic Version Extension

The ietf-yang-semver module defines a "version" extension -- a substatement to a module or submodule's "revision" statement -- that takes a YANG semantic version as its argument and specifies the version for the given module or submodule. The syntax for the YANG semantic version is defined in a typedef in the same module and described below.

4.3. YANG Semver Pattern

YANG artifacts that employ semantic versioning as defined in this document MUST use a version identifier 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 or equal to 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 be present if the following COMPAT element is included,

* COMPAT, if 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 YANG 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 when comparing YANG semantic versions, pre-release metadata can be used during module and submodule development as specified in Section 6. Both pre-release and build metadata are allowed in order to support all the [SemVer] rules. Thus, a version lineage that follows strict [SemVer] rules is allowed for a YANG artifact.

The ietf-yang-semver module included in this document defines an extension to apply a YANG Semver identifier to a YANG artifact as well as a typedef that formally specifies the syntax of the YANG Semver.

4.4. Semantic Versioning Scheme for YANG Artifacts

This document defines the YANG semantic versioning scheme that is used for YANG artifacts. The versioning identifier 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 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 use 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 identifiers used by the YANG semantic versioning scheme are exactly the same as those defined by the [SemVer] versioning scheme.

Every YANG module and submodule versioned using the YANG semantic versioning scheme specifies the module's or submodule's semantic version as the argument to the 'ysv:version' 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 identifier. For example, the first revision of a module or submodule may have been produced before this scheme was available.

YANG packages that make use of this YANG Semver will reflect that in the package metadata.

As stated above, the YANG semantic version is expressed as a string of the form: 'X.Y.Z_COMPAT'.

* 'X' is the MAJOR version. Changes in the MAJOR version number indicate changes that are non-backwards-compatible (Section 3.1.1 of [I-D.ietf-netmod-yang-module-versioning]) 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 "_compatible" or "_non_compatible" modifier.

* 'Z' is the PATCH version. Changes in the PATCH version number can indicate an editorial change to the YANG artifact. In conjunction with the '_COMPAT' modifier (see below) changes to 'Z' may indicate a more substantive module 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. Some examples include correcting a spelling mistake in the description of a leaf within a YANG module or submodule, non-significant whitespace changes (e.g., realigning description statements or changing indentation), or changes to YANG comments. 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.

* '_COMPAT' is an additional modifier, unique to YANG Semver (i.e., not valid in [SemVer]), that indicates 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.

- 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 ’_COMPAT’ modifier string is "sticky". Once a revision of a module has a modifier in the version identifier, then all subsequent module versions in that branch (i.e., those with the same X.Y version digits) will also have a modifier. The modifier can change from "_compatible" to "_non_compatible" in a subsequent version, but the modifier MUST NOT change from "_non_compatible" to "_compatible" and MUST NOT be removed. The persistence of the "_non_compatible" modifier ensures that comparisons of versions do not give the false impression of compatibility between two potentially non-compatible versions. If "_non_compatible" was removed, for example between versions "3.3.2_non_compatible" and "3.3.3" (where "3.3.3" was simply an editorial change), then comparing versions "3.3.3" to "3.0.0" would look like they are backwards compatible when they are not (since "3.3.2_non_compatible" was on the same MAJOR.MINOR branch and introduced a non-backwards-compatible change).

The YANG artifact name and YANG semantic version 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 but different content (and in the case of modules and submodules, 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.

4.4.1. YANG Semver with submodules

YANG Semver MAY be used to version submodules. Submodule versions are separate of any version on the including module, but if a submodule has changed, then the version of the including module MUST also be updated.

The rules for determining the version change of a submodule are the same as those defined in Section 4.3 and Section 4.4 as applied to YANG modules, except they only apply to the part of the module schema defined within the submodule's file.

One interesting case is moving definitions from one submodule to another in a way that does not change the resulting schema of the including module. In this case:

1. The including module has editorial changes and must receive a new version number with a PATCH component modification.

2. The submodule with the schema definition removed has non-backwards-compatible changes and must receive a new version number with a MAJOR component modification.

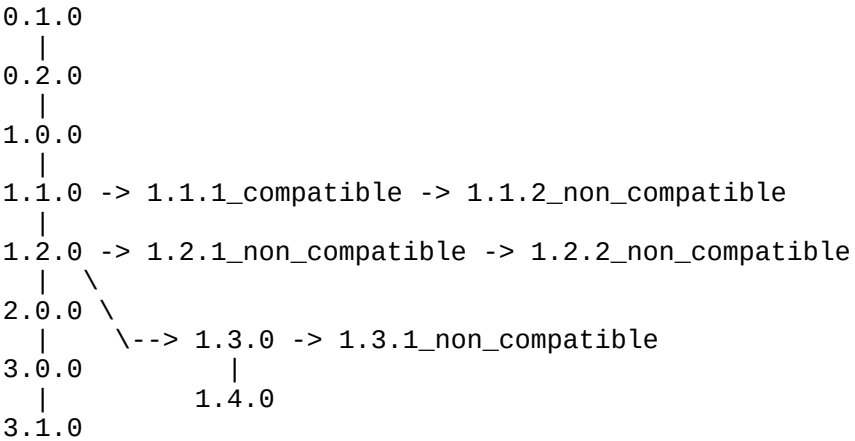
3. The submodule with the schema definitions added has backwards-compatible changes and must receive a new version number with a MINOR component modification.

Note that the meaning of a submodule may change drastically despite having no changes in content or revision due to changes in other submodules belonging to the same module (e.g. groupings and typedefs declared in one submodule and used in another).

4.4.2. Examples for YANG semantic versions

The following diagram and explanation illustrate how YANG semantic versions work.

YANG Semantic versions for an example module:



The tree diagram above illustrates how the version history might evolve for an example module. The tree diagram only shows the branching relationships between the versions. It does not describe the chronology of the versions (i.e. when in time each version was published relative to the other versions).

The following description lists an example of what the chronological order of the versions could look like, from oldest version to newest:

0.1.0 - first pre-release module version

0.2.0 - second pre-release 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)

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

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

1.1.1_compatible - backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0. This revision was created after 1.2.0 otherwise it may have been released as 1.2.0. (BC)

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

1.3.1_non_compatible - backport NBC fix, rename "baz" to "bar" (NBC)

1.2.1_non_compatible - backport NBC fix, rename "baz" to "bar" (NBC)

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

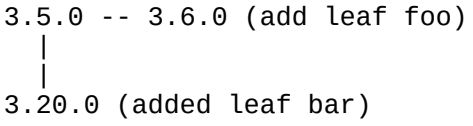
1.4.0 - introduce new leaf "ghoti" (BC)

3.1.0 - introduce new leaf "wobble" (BC)

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

4.4.3. Branching Limitations with YANG Semver

YANG artifacts that use the YANG Semver version scheme MUST ensure that two artifacts with the same MAJOR version number and no _compatible or _non_compatible modifiers are backwards compatible. Therefore, certain branching schemes cannot be used with YANG Semver. For example, the following branching approach using the following YANG Semver identifiers is not supported:



In this case, given only the YANG Semver identifiers 3.6.0 and 3.20.0, one would assume that 3.20.0 is backwards compatible with 3.6.0. But in the illegal example above, 3.20.0 is not backwards compatible with 3.6.0 since 3.20.0 does not contain the leaf foo.

Branching limitations should be carefully considered when doing software development. It is recommended to avoid only incrementing the PATCH digit on the main branch of YANG modules. See Appendix B Scenario 2 for an illustration and explanation.

4.5. YANG Semantic Version Update Rules

When a new version of an artifact is produced, then the following rules define how the YANG semantic version for the new artifact is calculated, based on the changes between the two artifact versions, and the YANG semantic version of the original artifact from which the changes are derived.

The following four rules specify the RECOMMENDED, and REQUIRED minimum, update to a YANG semantic version:

1. If an artifact is being updated in a non-backwards-compatible way, then the artifact version "X.Y.Z[_compatible|_non_compatible]" SHOULD be updated to "X+1.0.0" unless that version has already been used for this artifact but with different content, in which case the artifact version "X.Y.Z+1_non_compatible" SHOULD 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 SHOULD be updated to "X.Y+1.0", unless that version has already been used for this artifact but with different content, when the artifact version SHOULD be updated to "X.Y.Z+1_compatible" instead.

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

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

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

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

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

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

4. YANG artifact semantic version identifiers beginning with 0, i.e., "0.X.Y", are regarded as pre-release 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 6 for more details on using this notation during module and submodule development.

5. Additional pre-release rules for modules that have had at least one release are specified in Section 6.

Although artifacts SHOULD be updated according to the rules above, which specify the recommended (and minimum required) update to the version identifier, the following rules MAY be applied when choosing a new version identifier:

1. An artifact author MAY update the version identifier with a more significant update than described by the rules above. For example, an artifact could be given a new MAJOR version number (i.e., X+1.0.0), even though no non-backwards-compatible changes have occurred, or an artifact could be given a new MINOR version number (i.e., X.Y+1.0) even if the changes were only editorial.

2. An artifact author MAY skip versions. For example, a vendor might have released 1.0.0 of a module for general availability. Then, during subsequent development, used 1.1.0 and 1.2.0 for internal releases. The next generally available release could be 1.3.0 where external users never see 1.1.0 or 1.2.0.

Although YANG Semver always indicates when a non-backwards-compatible, or backwards-compatible change may have occurred to a YANG artifact, it does not guarantee that such a change has occurred, or that consumers of that YANG artifact will be impacted by the change. Hence, tooling, e.g., [I-D.ietf-netmod-yang-schema-comparison], also plays an important role for comparing YANG artifacts and calculating the likely impact from changes.

[I-D.ietf-netmod-yang-module-versioning] defines the "rev:non-backwards-compatible" extension statement to indicate where non-backwards-compatible changes have occurred in the module revision history. If a revision entry in a module's revision history includes the "rev:non-backwards-compatible" statement then that MUST be reflected in any YANG semantic version associated with that revision. However, the reverse does not necessarily hold, i.e., if the MAJOR version has been incremented it does not necessarily mean that a "rev:non-backwards-compatible" statement would be present.

4.6. Examples of the YANG Semver Version Identifier

4.6.1. Example Module Using YANG Semver

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

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

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

    description
      "to be completed";

    revision 2017-08-30 {
      description "Backport 'wibble' leaf";
      ysv:version 1.2.2_non_compatible;
    }
```

```
       revision 2017-07-30 {
         description "Rename 'baz' to 'bar'";
         ysv:version 1.2.1_non_compatible;
         rev:non-backwards-compatible;
       }

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

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

       revision 2017-02-07 {
         description "First release version.";
         ysv:version 1.0.0;
       }

       // Note: YANG Semver rules do not apply to 0.X.Y labels.
       // The following pre-release revision statements would not
       // appear in any final published version of a module. They
       // are removed when the final version is published.
       // During the pre-release phase of development, only a
       // single one of these revision statements would appear

       // revision 2017-01-30 {
       //   description "NBC changes to initial revision";
       //   ysv:version 0.2.0;
       //   rev:non-backwards-compatible; // optional
       //                         // (theoretically no
       //                         // 'previous released version')
       // }

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

       //YANG module definition starts here
       container example {
         description
           "This is an example container for the
            example-versioned-module.";
         leaf qux {
           type string;
```

```
          description
            "The qux instance of the device.";
        }
        leaf foo {
          type string;
          description
            "The foo instance of the device.";
        }
        leaf bar {
          type uint32;
          description
            "The magnitude of the foo.  This leaf was
             renamed from 'baz'.";
        }
        leaf wibble {
          type boolean;
          description
            "Does it wibble (true) or wobble (false)?";
        }
      }
    }
```

4.6.2. Example of Package Using YANG Semver

Below is an example YANG package that uses the YANG Semver version identifier based on the rules defined in this document. Note: '\' line wrapping per [RFC8792].

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-yang-pkg",
    "content-schema": {
      "module": "ietf-yang-packages@2022-03-04"
    },
    "timestamp": "2022-12-06T17:00:38Z",
    "description":  ["Example of a Package  \
       using YANG Semver"],
    "content-data": {
      "ietf-yang-packages:packages": {
        "package": [
          {
            "name": "example-yang-pkg",
            "version": "1.3.1",
            ...
          }
        ]
      }
    }
  }
}
```

Figure 2

5. Import Module by YANG Semantic Version

[I-D.ietf-netmod-yang-module-versioning] defines the rev:recommended-min-date extension that allows for the imported revision to be selected based on any revision that matches -- or is later -- than the specified revision date in the import. This section defines a similar extension for controlling import by YANG semantic version, as well as the rules for how imports are resolved.

5.1. The recommended-min-version Extension

The ietf-yang-semver module defines a "recommended-min-version" extension -- a substatement to the "import" statement -- that takes the numeric version triplet (i.e., X.Y.Z) of the YANG Semver as its argument and specifies that the minimum version of the associated module being imported SHOULD be greater than or equal to the specified value. The specific conditions for determining if a module's version is greater than or equal is defined in Section 5.2 below. At most one instance of recommended-min-version MAY be used. Removing, adding or changing the recommended-min-version statement is considered a backwards compatible change. An example use is:

```
import example-module {
    ysv:recommended-min-version 3.0.0;
}
```

5.2. Import by YANG Semantic Version Rules

When evaluating target module to import, only the numeric triplet of MAJOR.MINOR.PATCH is considered. The "_compatible" and "_non_compatible" modifiers as well as any metadata are ignored. A module to be imported is considered as meeting the recommended minimum version criteria if it meets one of the following conditions:

1. Has the exact same MAJOR, MINOR, PATCH as in the recommend-min-version value.

2. Has the same MAJOR and MINOR version numbers and a greater PATCH number.

3. Has the same MAJOR version number and greater MINOR number. In this case the PATCH number is ignored.

4. Has a greater MAJOR version number. In this case MINOR and PATCH numbers are ignored.

If the recommended-min-version is specified as 3.1.0, the following examples would satisfy that recommend-min-version:

3.1.0 (by condition 1 above)

3.1.1 (by condition 2 above)

3.2.0 (by condition 3 above)

4.1.2 (by condition 4 above)

3.1.1_compatible (by condition 2 above, noting that modifiers are ignored)

3.1.2_non_compatible (by condition 2 above, noting that modifiers are ignored)

3.3.0-00 (by condition 3 above, noting that the pre-release metadata is ignored)

If a compiler that supports the recommended-min-version extension cannot locate a module with a version that is viable according to the conditions above, that YANG compiler should emit a warning, and then continue to resolve the import based on established [RFC7950] rules.

6. Guidelines for Using YANG Semver During Module Development

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

All modules and submodules being developed or updated that use the YANG Semver versioning scheme in MUST have unique YANG Semver version identifiers throughout their lifecycle.

Development of a new YANG module or submodule outside of the IETF that uses the YANG Semver versioning scheme SHOULD begin with a 0 for the MAJOR version component. This allows the module or submodule to disregard strict SemVer rules with respect to non-backwards-compatible changes during its initial development. However, module or submodule developers MAY choose to use the SemVer pre-release syntax instead with a 1 for the MAJOR version number. For example, an initial module or submodule version might be either 0.1.0 or 1.0.0-alpha.1. If the authors choose to use the 0 MAJOR version number scheme, they MAY switch to the pre-release scheme with a MAJOR version number of 1 when the module or submodule is nearing initial release (e.g., a module’s or submodule’s version 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 pre-release metadata MUST adhere to [SemVer] syntax. The following are examples of valid pre-release versions:

1.0.0-alpha.1

1.0.0-alpha.3

1.0.0-beta.42

1.0.0-202007.rc.1

1.0.0-20250106

1.0.0-03

When updating an existing module or submodule using the YANG Semver versioning scheme, the intended target semantic version can be used along with pre-release notation. For example, if a released module or submodule which has a current version of 1.0.0 is being modified with the intent to make non-backwards-compatible changes, the first development MAJOR version component should be 2 with some pre-release notation such as -alpha.1, making the version 2.0.0-alpha.1. It may

be prudent to include the year or year and month development began (e.g., 2.0.0-201907-alpha.1) in order to guarantee uniqueness. As a module or submodule undergoes development, it is possible that the original intent changes. For example, a 1.0.0 version of a module or submodule 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 version for a given module or submodule is 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).

Module and submodule developers SHOULD maintain only one revision statement in a pre-released module or submodule that reflects its latest revision.

6.1. YANG Semver in IETF Modules

All published IETF modules and submodules MUST use YANG semantic versions in their revisions.

6.1.1. YANG Semver in New IETF Modules

Development of a new module or submodule (i.e., when there is no existing module of the same name) within the IETF MUST begin by using 0 as the MAJOR version component. If the Internet Draft containing the module or submodule is an adopted document, pre-release metadata MUST not be used. Instead, the Y.Z (i.e., MINOR.PATCH) version components MUST be incremented with new revisions in order to maintain version uniqueness.

6.1.2. YANG Semver when updating IETF Modules

When revising an existing IETF module or submodule, the version MUST use the target (i.e., intended) MAJOR and MINOR version components with a 0 PATCH version number. If the intended RFC release will be non-backwards-compatible with the current RFC release, the MINOR version number MUST be 0.

6.1.2.1. Guidelines for IETF Modules Previously Published without YANG Semvers

For IETF YANG modules and submodules that have already been published, versions MUST be retroactively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver version rules specified in Section 4.5. For example, if a module or submodule 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).

6.1.3. YANG Semver when there are multiple parallel competing IETF drafts

Whether a new IETF module is being defined, or a previously published IETF module is being updated, special considerations are required when multiple parallel competing drafts are in progress for a new module or a module update.

The following recommendations apply to the 2nd, 3rd, etc. parallel drafts that develop a module or submodule with the same name. The first draft published can simply use the guidelines above without any extra metadata described below. The YANG Semver in subsequent drafts SHOULD include the full document name as a pre-release version identifier, including the current document revision. For example, if a module or submodule 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 versions SHOULD 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 or submodule. When using the 0 MAJOR version for initial module or submodule development in an individual Internet Draft (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 or submodule.

Other approaches are also possible to ensure uniqueness of YANG Semver identifiers across all parallel competing modules and revisions before IETF adoption including:

1. Using unique module names (i.e., module names with unique prefixes such as the author's name, e.g. johnsmith-interfaces.yang), or

2. Using unique strings within the YANG Semver pre-release metadata (e.g. 2.0.0-johnsmith-02).

Once an Internet Draft containing the module or submodule is adopted as an IETF document, the need for pre-release metadata changes. For new modules and submodules, the pre-release metadata MUST NOT be used. Instead, the Y.Z (i.e., MINOR.PATCH) version components are incremented as desired while leaving the X (i.e., MAJOR) component at 0. For module or submodule revisions, only the revision of the Internet Draft is required. For example, if the module ietf-foo.yang is being revised by draft-ietf-netmod-rfcABCDbis to include backwards-compatible changes, its versions would be 1.1.0-01, 1.1.0-02, etc. as the draft progresses.

Some draft revisions may not include an update to the YANG modules or submodules contained in the draft. In that case, those modules or submodules that are not updated do not require a change to their versions. Updates to the YANG Semver version MUST only be done when the revision of the module changes.

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

7. Updates to ietf-yang-library

This document updates YANG 1.1 [RFC7950] and YANG library [RFC8525] to clarify how ambiguous module imports are resolved. It also defines the YANG module, ietf-yang-library-semver, that augments YANG library [RFC8525] with a version leaf for modules and submodules.

7.1. YANG library versioning augmentations

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

```
module: ietf-yang-library-semver

  augment /yanglib:yang-library/yanglib:module-set/yanglib:module:
    +--ro version?   ysv:version
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module
            /yanglib:submodule:
    +--ro version?   ysv:version
  augment /yanglib:yang-library/yanglib:module-set
            /yanglib:import-only-module:
    +--ro version?   ysv:version
  augment /yanglib:yang-library/yanglib:module-set
            /yanglib:import-only-module/yanglib:submodule:
    +--ro version?   ysv:version
```

Figure 3

7.1.1. Advertising version

The ietf-yang-library-semver YANG module augments the "module" and "submodule" lists in ietf-yang-library with "version" leafs to optionally declare the version identifier associated with each module and submodule.

8. YANG Modules

This YANG module contains the typedef for the YANG semantic version and the identity to signal its use.

```
<CODE BEGINS> file "ietf-yang-semver@2025-01-21.yang"
module ietf-yang-semver {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-semver";
  prefix ysv;

  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>
     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>
     Author:   Reshad Rahman
               <mailto:reshad@yahoo.com>
     Author:   Balazs Lengyel
               <mailto:balazs.lengyel@ericsson.com>
     Author:   Jason Sterne
               <mailto:jason.sterne@nokia.com>
     Author:   Benoit Claise
               <mailto:benoit.claise@huawei.com>";
  description
    "This module provides type and grouping definitions for YANG
     packages.

     Copyright (c) 2025 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Revised BSD License
```

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

      The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
      NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
      'MAY', and 'OPTIONAL' in 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.

      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.
  // RFC Ed. update the ysv:version to "1.0.0".

  revision 2025-01-21 {
    ysv:version "0.20.0";
    description
      "Initial revision";
    reference
      "RFC XXXX: YANG Semantic Versioning.";
  }

  /*
   * Extensions
   */

  extension version {
    argument yang-semantic-version;
    description
      "The version extension can be used to assign a version number
       to a module or submodule revision.

       The format of the version extension argument MUST conform
       to the 'version' typedef defined in this module.

       The statement MUST only be a substatement of the revision
       statement.  Zero or one version statements per parent
       statement are allowed.  No substatements for this extension
       have been standardized.

       Versions MUST be unique amongst all revisions of a
       module or submodule.
```

```
          Adding a version is a backwards-compatible
          change.  Changing or removing an existing version in
          the revision history is a non-backwards-compatible
          change, because it could impact any references to that
          version.";
        reference
          "RFC XXXX: YANG Semantic Versioning;
          Section 4.2, YANG Semantic Version Extension";
      }

      extension recommended-min-version {
        argument yang-semantic-version;
        description
          "Recommends the versions of the module that may be imported to
          one that is greater than or equal to the specified version.

          The format of the recommended-min-version extension argument
          must be the MAJOR.MINOR.PATCH components from the
          'version' typedef defined in this module (i.e., excluding
          the '_compatible' and '_non_compatible' modifiers as
          well as any metadata).

          The statement MUST only be a substatement of the import
          statement.  Zero or one 'recommended-min-version'
          statements per parent statement are allowed.  No
          substatements for this extension have been
          standardized.

          A module to be imported is considered as meeting the
          recommended minimum version criteria if it meets one of
          the following conditions:

          * Has the exact same MAJOR, MINOR, PATCH as in the
             recommend-min-version value.
          * Has the same MAJOR and MINOR version numbers and a
            greater PATCH number.
          * Has the same MAJOR version number and greater MINOR number.
            In this case the PATCH number is ignored.
          * Has a greater MAJOR version number.  In this case MINOR
            and PATCH numbers are ignored.

          Adding, removing or updating a 'recommended-min-version'
          statement to an import is a backwards-compatible change.";
        reference
          "RFC XXXX: YANG Semantic Versioning; Section 5,
          Import Module by Semantic Version";
      }
```

```
    /*
     * Typedefs
     */

    typedef version {
      type string {
        length "5..128";
        pattern '[0-9]+[.][0-9]+[.][0-9]+(_(non_)?compatible)?'
              + '(-[A-Za-z0-9.-]+[.-][0-9]+)?([+][A-Za-z0-9.-]+)?';
      }
      description
        "Represents a YANG semantic version.  The rules governing the
         use of this version identifier are defined in the
         reference for this typedef.";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }
  }
  <CODE ENDS>

  This YANG module contains the augmentations to the ietf-yang-library.

  <CODE BEGINS> file "ietf-yang-library-semver@2025-07-13.yang"
  module ietf-yang-library-semver {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-library-semver";
    prefix yl-semver;

    import ietf-yang-semver {
      prefix ysv;
      reference
        "XXXX: YANG Semantic Versioning";
    }
    import ietf-yang-library {
      prefix yanglib;
      reference
        "RFC 8525: YANG Library";
    }

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

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

```
        Author:   Reshad Rahman
                  <mailto:reshad@yahoo.com>

        Author:   Robert Wilton
                  <mailto:rwilton@cisco.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
         and submodule level version identifiers.

         Copyright (c) 2025 IETF Trust and the persons identified as
         authors of the code.  All rights reserved.

         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and subject
         to the license terms contained in, the Revised BSD License
         set forth in Section 4.c of the IETF Trust's Legal Provisions
         Relating to IETF Documents
         (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 ysv:version with 1.0.0 and
      // remove this note.

      revision 2025-07-13 {
        ysv:version "0.21.0";
        description
          "Initial revision";
        reference
          "RFC XXXX: YANG Semantic Versioning";
      }
```

```
    grouping module-version {
      description
        "Grouping for module version information.";
      leaf version {
        type ysv:version;
        description
          "The version of the module or submodule.
           The value MUST match the version value in the specific
           revision of the module or submodule loaded in this
           module-set.";
        reference
          "RFC XXXX: YANG Semantic Versioning;
           Section 7.1.1, Advertising version";
      }
    }

    // library 1.0 modules-state is not augmented with version

    augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
      description
        "Add a version to module information";
      uses module-version;
    }

    augment "/yanglib:yang-library/yanglib:module-set/yanglib:module/"
          + "yanglib:submodule" {
      description
        "Add a version to submodule information";
      uses module-version;
    }

    augment "/yanglib:yang-library/yanglib:module-set/"
          + "yanglib:import-only-module" {
      description
        "Add a version to module information";
      uses module-version;
    }

    augment "/yanglib:yang-library/yanglib:module-set/"
          + "yanglib:import-only-module/yanglib:submodule" {
      description
        "Add a version to submodule information";
      uses module-version;
    }
  }
  <CODE ENDS>
```

9. Contributors

The following people made substantial contributions to this document:

Bo Wu
lana.wubo@huawei.com

Jan Lindblad
jlindbla@cisco.com

Figure 4

10. Acknowledgments

This document grew out of the YANG module versioning design team that started after IETF 101. The team consists of the following members whom have worked on the YANG versioning project: Balazs Lengyel, Benoit Claise, Bo Wu, Ebben Aries, Jan Lindblad, Jason Sterne, Joe Clarke, Juergen Schoenwaelder, Mahesh Jethanandani, Michael (Wangzitao), Per Andersson, Qin Wu, Reshad Rahman, Tom Hill, and Rob Wilton.

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update]. We would like to thank Kevin D'Souza for his initial work in this problem space.

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 to thank both Anees Shaikh and Rob Shakir for their input into this problem space.

We would also like to thank Joseph Donahue from the SemVer.org project for his input on SemVer use and overall document readability.

11. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

That said, the YANG module in this document does not define any writeable nodes. The extensions defined are only used to document YANG artifacts.

12. IANA Considerations

12.1. YANG Module Registrations

This document requests IANA to register URIs in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations are requested.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-semver

Registrant Contact: The IESG.

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

URI: urn:ietf:params:xml:ns:yang:ietf-yang-library-semver

Registrant Contact: The IESG.

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

The following YANG modules are requested to be registered in the "IANA Module Names" [RFC6020]. Following the format in RFC 6020, the following registrations are requested:

The ietf-yang-semver module:

Name: ietf-yang-semver

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

Prefix: ys

Reference: [RFCXXXX]

The ietf-yang-library-semver module:

Name: ietf-yang-library-semver

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-library-semver

Prefix: yl-semver

Reference: [RFCXXXX]

12.2. Guidance for YANG Semver in IANA maintained YANG modules and submodules

Note for IANA (to be removed by the RFC editor): Please check that the registries and IANA YANG modules and submodules are referenced in the appropriate way.

IANA is responsible for maintaining and versioning some YANG modules and submodules, e.g., iana-if-types.yang [IfTypeYang] and iana-routing-types.yang [RoutingTypesYang].

In addition to following the rules specified in the IANA Considerations section of [I-D.ietf-netmod-yang-module-versioning], IANA maintained YANG modules and submodules MUST also include a YANG Semver version identifier for all new revisions, as defined in Section 4.

The YANG Semver version associated with the new revision MUST follow the rules defined in Section 4.5.

Note: For IANA maintained YANG modules and submodules that have already been published, versions MUST be retroactively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver rules specified in Section 4.5.

Most changes to IANA maintained YANG modules and submodules are expected to be backwards-compatible changes and classified as MINOR version changes. The PATCH version may be incremented instead when only editorial changes are made, and the MAJOR version would be incremented if non-backwards-compatible changes are made.

Given that IANA maintained YANG modules are versioned with a linear history, it is anticipated that it should not be necessary to use the "_compatible" or "_non_compatible" modifiers to the "Z_COMPAT" version element.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[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>.

[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>.

[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>.

[I-D.ietf-netmod-yang-module-versioning] Wilton, R., Rahman, R., Lengyel, B., Clarke, J., and J. Sterne, "Updated YANG Module Revision Handling", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-13, 6 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-versioning-13>.

13.2. Informative References

[I-D.clacla-netmod-yang-model-update] Claise, B., Clarke, J., Lengyel, B., and K. D’Souza, "New YANG Module Update Procedure", Work in Progress, Internet-Draft, draft-clacla-netmod-yang-model-update-06, 2 July 2018, <https://datatracker.ietf.org/doc/html/draft-clacla-netmod-yang-model-update-06>.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., and J. Sterne, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-06, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-packages-06>.

[I-D.ietf-netmod-yang-schema-comparison]
Andersson, P. and R. Wilton, "YANG Schema Comparison", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-schema-comparison-02, 14 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-schema-comparison-02>.

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

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

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

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

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

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

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

[SemVer] "Semantic Versioning 2.0.0 (text from June 19, 2020)", <https://github.com/semver/semver/blob/8b2e8eec394948632957639dfa99fc7ec6286911/semver.md>.

[IfTypeYang] "iana-if-type YANG Module", <https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml>.

[RoutingTypesYang] "iana-routing-types YANG Module", <https://www.iana.org/assignments/iana-routing-types/iana-routing-types.xhtml>.

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 ysv:version) of the module as it's being initially developed.

Version lineage for initial module development:

```
0.1.0-draft-jdoe-netmod-example-module-00
  |
0.2.0-draft-jdoe-netmod-example-module-01
  |
0.3.0-draft-jdoe-netmod-example-module-02
  |
0.3.1-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 pre-release metadata is no longer required.

Continued version progression after adoption:

```
0.4.0
  |
0.4.1
  |
0.5.0
```

At this point, the draft is standardized and becomes RFC12345 and the YANG module version 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-asmith-netmod-exmod-changes. These are initially developed in parallel with the following versions.

Parallel development for next module revision (track 1):

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

In parallel with (track 2):

```
1.1.0-draft-asmith-netmod-exmod-changes-00
  |
1.1.0-draft-asmith-netmod-exmod-changes-01
```

At this point, the WG decides to merge some aspects of both and adopt the work in asmith's draft as draft-ietf-netmod-exmod-changes. Since the draft is now an adopted IETF document, the pre-release metadata changes to simply include the adopted draft's revision. A single version progression continues.

```
1.1.0-00
  |
1.1.0-01
  |
1.1.0-02
  |
1.1.0-03
```

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

Appendix B. Examples of _COMPAT Modifier Use

The following scenarios illustrate some situations where the _COMPAT modifier (i.e., _compatible and _non_compatible) might be used in the YANG Semver of a YANG module. While module development is considered here, this is appliacable to any YANG artifact.

In the scenarios below, the X axis represents relative time (date & time) that an module is being published. Module versions to the right are more recent than module versions to the left. In scenario 1, for example, the time ordered publishing of module versions is: 2.0.0, 2.1.0, 3.0.0, A, B.

It is also useful to consider the leftmost sequence (diagonal column) of versions (going mostly downwards and slightly right) as the "main branch" in a software development branching scheme, and the sequence of versions moving rightwards (rows) as bug fixes in released branches (i.e., maintenance releases in a single major release stream).

These scenarios are generally more applicable for vendor modules (which require modifications that are not at the head of the main branch) and not expected to occur much in modules published by standard bodies (which tend to be linear and only update the head of the single branch).

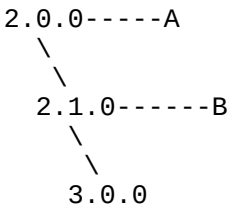


Figure 5: Scenario 1

For each revision A and B, the following are possible YANG Semver versions when the revision is backwards comptible (BC) or non backwards compatible (NBC):

Revision A (BC): 2.0.1_compatible

Revision A (NBC): 2.0.1_non_compatible

2.2.0 is not allowed for A (see Section 4.4.3), and 4.0.0 is not recommended (likely to cause confusion since it does not have 3.0.0 as an ancestor).

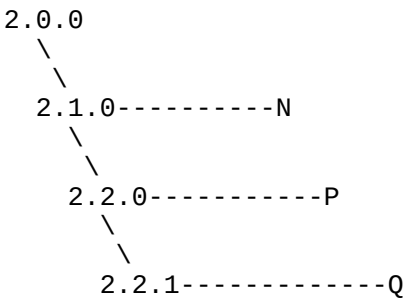


Figure 6: Scenario 2

For each revisions N, P and Q, the following are possible YANG Semver versions when the revision is backwards comptible (BC) or non backwards compatible (NBC):

Revision N (BC): 2.1.1_compatible

Revision N (NBC): 2.1.1_non_compatible

Revision P: This is a situation that illustrates the limits of branching in YANG Semver. It is not recommended to create any new BC or NBC version off of 2.2.0. One approach to avoid this situation is to avoid only incrementing the PATCH digit in the main branch (i.e., increment at least the MINOR digit).

For revision Q -- given that it is at the head of the branch -- the following are possible YANG Semver versions when the revision is backwards compatible (BC) or non backwards compatible (NBC):

Revision Q (BC): 2.3.0

Revision Q (NBC): 3.0.0

Authors' Addresses

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

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Reshad Rahman
Equinix
Email: reshad@yahoo.com

Balazs Lengyel
Ericsson
1117 Budapest
Magyar Tudosok Korutja
Hungary
Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Jason Sterne
Nokia
Email: jason.sterne@nokia.com

Benoit Claise
Huawei
Email: benoit.claise@huawei.com