NETMOD
Internet-Draft
Intended status: Informational
Expires: 18 September 2026

R. Wilton
Cisco

17 March 2026

Guidance for Managing YANG Modules in RFCs and IANA Registries
draft-ietf-netmod-iana-yang-guidance-01

Abstract

This document provides guidance to the RFC Editor and IANA on managing YANG modules in RFCs and IANA registries, ensuring consistent application of YANG Semantic Versioning rules.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://rgwilton.github.io/iana-yang-guidance/draft-verdt-iana-yang-guidance.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-netmod-iana-yang-guidance/.

Discussion of this document takes place on the Network Modelling Working Group mailing list (mailto:netmod@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/netmod/. Subscribe at https://www.ietf.org/mailman/listinfo/netmod/.

Source for this draft and an issue tracker can be found at https://github.com/rgwilton/iana-yang-guidance.

Status of This Memo

This Internet-Draft will expire on 18 September 2026.

Table of Contents

1. Open Issues/Questions:

1. Do we need guidance to IANA in this document to list modules both by revision date and version? I.e., following the filename convention.

2. This document is informational, is it appropriate to use RFC 2119 language?

3. For the RFC Editor and ADs, do we want to allow the RFC Editor to apply errata to IETF YANG modules?

4. For Section 5, should we give examples of the rules, or just reference the module versioning draft [Reshad]?

2. For Reviewers of this document

RFC Editor - please delete this section before publication

This draft should be carefully reviewed by:

* IANA and RFC Editor to check that they agree with the workflows

* OPS ADS & IESG (if needed) that they agree that the IETF should delay publishing YANG modules in approved internet drafts until after the RFC Editor has had the opportunity to review and amend the text.

* YANG Doctors and NETMOD to ensure that they are happy with the requirements being placed upon them.

3. Introduction

YANG [RFC6020] [RFC7950] modules are used to model network management data and protocols. The IETF publishes YANG modules as part of RFCs, and the Internet Assigned Numbers Authority (IANA) maintains YANG modules that are derived from IANA registries. Both processes require careful attention to module versioning and the timing of publication to ensure that implementations can correctly assess module version compatibility when modules are updated.

This document provides informational guidance to both the RFC Editor and IANA for managing YANG modules in two distinct scenarios:

1. *Managing YANG Modules in RFCs*: When documents containing YANG modules are approved by the IESG and processed for publication as RFCs, both the RFC Editor and IANA have responsibilities to ensure that modules are correctly versioned and published.

2. *Managing IANA-Maintained YANG Modules*: When IANA registries are updated, any YANG modules derived from those registries must be updated accordingly with proper versioning.

The guidance in this document is informational rather than prescriptive. It describes recommended practices and procedures that reflect current consensus within the NETMOD working group and the IETF operations and management community. While following this guidance will help ensure consistent and correct handling of YANG modules, specific situations may require consultation with the YANG Doctors (as described in Section 8).

Note: In addition to the guidance detailed in this document, there is a broader, ongoing discussion within the IETF community around the processes and responsibilities for managing YANG modules in RFCs. For further information and the latest proposals, see [I-D.boucadair-veloce-yang]. The recommendations and operational practices described here may be revised in the future to reflect outcomes from that work.

The procedures and classifications in this document are drawn from text and general guidance on the following IETF specifications:

* [I-D.ietf-netmod-rfc8407bis] - Provides general guidelines for IETF YANG module authors

* [I-D.ietf-netmod-yang-module-versioning] - Defines updated YANG module revision handling, including rules for backwards-compatible and non-backwards-compatible changes

* [I-D.ietf-netmod-yang-semver] - Defines YANG Semantic Versioning (YANG Semver) for YANG modules

* [I-D.ietf-netmod-yang-module-filename] - Defines filename conventions for YANG modules versioned using YANG Semver

4. Conventions and Definitions

This document uses the following terminology from [I-D.ietf-netmod-yang-module-versioning]:

Backwards-Compatible (BC) Change A change to a YANG module that conforms to the backwards-compatible update rules defined in section 3.1.1 of [I-D.ietf-netmod-yang-module-versioning]. BC changes require incrementing the MINOR version number.

Non-Backwards-Compatible (NBC) Change A change to a YANG module that does not conform to the backwards-compatible update rules defined in section 3.1.2 of [I-D.ietf-netmod-yang-module-versioning]. NBC changes require incrementing the MAJOR version number and adding the rev:non-backwards-compatible extension statement within the revision statement in the YANG module.

This document uses the following terminology from [I-D.ietf-netmod-yang-semver]:

YANG Semver YANG Semantic Versioning - a version identifier in the

format _MAJOR.MINOR.PATCH_COMPAT_ that indicates the compatibility level of a YANG module, as defined in [I-D.ietf-netmod-yang-semver].

Editorial Change A change to a YANG module that does not affect the semantic meaning or functionality of the module. Editorial changes only require incrementing the PATCH version number, as described in section 4.4 of [I-D.ietf-netmod-yang-semver].

In addition, this document defines:

IANA-Maintained Module A YANG module maintained by IANA, typically derived from one or more IANA registries. These modules have names starting with "iana-" (e.g., iana-if-type, iana-routing-types).

5. Background on YANG Versioning

5.1. YANG Semantic Versioning

YANG Semantic Versioning (YANG Semver), defined in [I-D.ietf-netmod-yang-semver], uses a version identifier in the format MAJOR.MINOR.PATCH (with an optional _COMPAT suffix for branched development):

* *MAJOR* version increments indicate non-backwards-compatible (NBC) changes, with _MINOR_ and _PATCH_ fields reset to 0

* *MINOR* version increments indicate backwards-compatible (BC) feature additions, with the _PATCH_ field reset to 0

* *PATCH* version increments indicate editorial or documentation-only changes

* *_COMPAT* is used for branched development trees and is not applicable to modules published by the IETF or maintained by IANA.

If an update to a YANG module contains a mix of changes, then the version number is updated as per the most impactful change. E.g., if a change included both backwards-compatible feature additions and editorial changes then the _MINOR_ version field is incremented and the _PATCH_ version field is set to 0, e.g., as per the second example below. If in doubt as to which category a particular change fits into, it is always better to err on the side of caution and choose the more significant version change.

For example, if a published IETF YANG module is at version _1.2.3_:

* An editorial only change would update it to _1.2.4_

* A backwards-compatible feature addition would update it to _1.3.0_

* A non-backwards-compatible change would update it to _2.0.0_.

Pre-release versions (versions with MAJOR = 0, e.g., "0.2.0", or with a pre-release suffix, e.g., "1.3.0-draft-verdt-iana-yang-guidance") indicate modules that have not completed the IETF standardization process and whose revision content is subject to change in non-backwards-compatible ways without corresponding changes to the major version number. Published IETF and IANA YANG modules should always be at version "1.0.0" or later, and should never include a pre-release suffix. The initial published version should be "1.0.0".

5.2. Backwards Compatibility Rules

The rules that determine whether a change to a YANG module is backwards-compatible or non-backwards-compatible are defined in Section 3.1 of [I-D.ietf-netmod-yang-module-versioning]. These rules refine and extend the update rules specified in Section 11 of [RFC7950].

Section 3.1.1 of [I-D.ietf-netmod-yang-module-versioning] defines backwards-compatible changes, examples include:

* Adding new schema nodes (e.g., new enum values, identities, leafs, containers)

* Adding or updating "description" and "reference" statements (provided the semantic meaning is unchanged)

* Changing the status of a schema node from "current" to "deprecated" (e.g., by adding a status: "deprecated" statement)

Section 3.1.2 of [I-D.ietf-netmod-yang-module-versioning] defines non-backwards-compatible changes, examples include:

* Removing schema nodes (unless they already have status "obsolete")

* Changing the status of a schema node from "current" or "deprecated" to "obsolete"

* Renaming schema nodes or changing their identifiers

* Changing data types in ways that alter syntax or semantics

* Changing numeric values assigned to enumerations

* Modifying "description" statements in ways that change semantic meaning or behavior

In addition, section 4.4 of [I-D.ietf-netmod-yang-semver] defines editorial changes as the subset of backwards-compatible changes that have no impact on the semantics or syntax of a YANG module, examples include:

* Corrections to comments, descriptions, or references that do not change the semantic meaning

* Formatting improvements such as whitespace or indentation changes

* Updates to contact information or copyright statements

5.3. The rev:non-backwards-compatible Extension

The YANG module versioning framework [I-D.ietf-netmod-yang-module-versioning] defines the "rev:non-backwards-compatible" extension statement. This extension MUST be added as a substatement of a revision statement whenever that revision contains non-backwards-compatible changes relative to the previous revision.

The following example illustrates this extension in use. In the example, an identity 'foo' was added in version 1.3.0, but was subsequently renamed to 'bar' in version 2.0.0. Since renaming is a non-backwards-compatible change, the major version number is incremented and the rev:non-backwards-compatible extension is included in the revision statement in version 2.0.0 of the YANG module:

```
revision 2025-11-15 {
  ysv:version "2.0.0";
  rev:non-backwards-compatible;
  description
    "Renamed identity 'foo' to 'bar'.";
}
revision 2025-06-01 {
  ysv:version "1.3.0";
  description
    "Added identity 'foo'.";
}
...
```

Figure 1: Revision history example from a YANG module

5.4. Module Immutability

A fundamental principle of YANG module versioning is that once a module revision is published with a specific revision date and version number, its content is immutable (much like an RFC is). The published content of that revision MUST NOT change. Any change to the module content requires publishing a new revision with a new revision date and an updated YANG Semver.

This immutability principle has important implications:

* Modules in Internet-Drafts MUST use pre-release versions (e.g., 0.1.0 or 2.0.0-draft-name) to indicate that the content may still change.

* Once a document is approved by the IESG and has been processed by the RFC editor, (*TODO - Is IANA involved too?*) then the module version MUST be updated to the correct release version (e.g., 1.0.0, or 2.0.0) before publication in an RFC or made available in the IANA YANG Module Names registry [IANA-YANG-PARAMETERS].

* IANA-maintained modules MUST publish a new YANG module revision any time IANA registry changes require YANG module updates.

6. YANG Modules in Documents Being Published as RFCs

This section describes the workflow and responsibilities for managing YANG modules in documents that have been approved by the IESG and are being processed for publication as RFCs. Both the RFC Editor and IANA have roles in this process.

6.1. Core Requirements

All YANG modules published by the RFC Editor or maintained by IANA MUST meet the following requirements:

1. *YANG Semver Version*: Every module MUST include a semantic version number using the ysv:version statement in its most recent revision. The version MUST be correct relative to any previously version of the same module published either by the RFC editor or on the IANA website.

2. *NBC Extension for NBC Changes*: If the module contains non-backwards-compatible changes relative to the previously published version, the revision statement MUST include the rev:non-backwards-compatible extension.

3. *Revision Immutability*: A published YANG module with a specific revision date and version number is immutable. Its content MUST NOT change without also changing the revision date and version number. For this reason, modules in Internet-Drafts use pre-release versions (e.g., versions with MAJOR = 0 such as 0.1.0, or versions with a pre-release suffix such as 2.0.0-05, where the -05 is the Internet Draft number where the YANG module was updated) to indicate that content may still change before final publication.

4. *RFC Code Markers*: YANG modules in RFCs MUST be properly marked with <CODE BEGINS> and <CODE ENDS> markers (or equivalent in the source format) to enable automated extraction. The markers MUST include the filename following the conventions in [I-D.ietf-netmod-yang-module-filename].

6.2. Workflow Steps

The following steps describe the coordinated process between the RFC Editor and IANA for handling YANG modules during RFC publication:

6.2.1. Step 1: IESG Approval with Pre-Release Version

When a document is approved by the IESG, any YANG modules it contains typically have pre-release version numbers (e.g., 0.4.0, 1.1.0-03, 2.0.0-07). These pre-release versions indicate that the module content may still be subject to editorial changes during RFC Editor processing.

6.2.2. Step 2: RFC Editor Processing

During RFC Editor processing, the RFC Editor may make editorial changes to the YANG module, such as:

* Improving description text for clarity without changing semantic meaning

* Updating references to use final RFC numbers instead of draft names

* Correcting typographical errors

* Standardizing formatting and style

These editorial changes are appropriate and expected. The RFC Editor SHOULD:

* Coordinate with document authors regarding any substantive changes

* Ensure that only editorial changes (as defined in Section 5) are made without author consultation

* If more significant changes are needed that might be backwards-compatible or non-backwards-compatible, consult with the authors to determine the correct version number and whether the rev:non-backwards-compatible extension is required.

* Ensure that final module is correctly formatted (e.g., by running Appendix A.1.1.2)

6.2.3. Step 3: Finalizing the Module Version

Before publication, the module version MUST be updated from the pre-release version to a release version. The RFC Editor, in coordination with the document authors:

* Updates the version to remove pre-release indicators (e.g., 0.1.0 → 1.0.0, or 1.1.0-<draft-num> → 1.1.0)

* Uses pyang (Appendix A.1.1.3) to check that an appropriate new version has been choosen based on the relationship to any previously published version of the module. Tooling is not infallible, so if the suggested version by the tooling is unexpected then please reach out for additional guidance, as per Section 8.

* Adds the rev:non-backwards-compatible extension if NBC changes have occurred since the previous publication

* Updates the revision date to reflect the date of the final revision

6.2.4. Step 4: Validate the Module

YANG modules are expected to be provided to the RFC Editor for publication already passing validation (pyang and yanglint). However, it is possible that mistakes could be introduced when editing the YANG modules so validation should be re-run to ensure that IETF does not publish invalid YANG modules.

After all updates are completed, or as updates as made, and after any formatting, then validation tools MUST be run over the resultant module to ensure that there are no warnings or errors. pyang validation (Appendix A.1.1.1) MUST be performed, and it is RECOMMENDED that _yanglint_ (Appendix A.1.2) validation is also performed.

If the tools return any warnings or errors then the authors should help fix them, potentially seeking additional guidance if required, as per Section 8.

If further changes are made, that the step 3 versioning check MUST be re-run to ensure that the module version is still correct.

6.2.5. Step 5: IANA Delay of Publication

IANA SHOULD delay publishing the YANG module to the IANA YANG Parameters registry until the RFC Editor has completed editing the module. This coordination ensures that:

* The IANA-published version matches the RFC-published version exactly
* No discrepancies exist between the two authoritative sources
* The module reference to the RFC (if present) is correct

6.2.6. Step 6: Coordinated Publication

Once the RFC Editor has finalized the module:

* The RFC is published with the final module content
* IANA publishes the module to the IANA YANG Parameters registry at approximately the same time
* The module filename follows the conventions in [I-D.ietf-netmod-yang-module-filename]
* IANA registers the module in the "YANG Module Names" registry if it is not already registered

7. IANA-Maintained YANG Modules

This section describes the process for IANA to update and publish YANG modules that are maintained by IANA and derived from IANA registries.

7.1. Overview

Some IANA registries have corresponding YANG modules that represent registry contents in a machine-readable format, and that are published at [IANA-YANG-PARAMETERS]. Examples include:

* *iana-if-type.yang* - derived from the Interface Types (ifType) registry [iana-iftype-registry]

* *iana-routing-types.yang* - derived from Address Family Numbers [iana-afnum-registry] and SAFI Parameters [iana-safi-registry] registries

* *iana-bgp-types.yang* - derived from BGP Parameters registries [iana-bgp-parameters]

When these registries are updated, the corresponding YANG modules MUST be updated accordingly by IANA, following the same versioning rules described in Section 5.

7.2. Characteristics of IANA-Maintained Modules

IANA-maintained YANG modules typically:

* Have names starting with "iana-"

* Contain primarily enumeration typedefs or identity definitions that are derived from registry entries

* Are updated more frequently than IETF-defined modules

* Follow a linear version history without branching

* Have a much simpler structure than general-purpose YANG modules, which simplifies classification of changes

Because IANA-maintained YANG modules are always expected to follow a linear version history without branching, the __COMPAT_ modifier defined in [I-D.ietf-netmod-yang-semver] is not needed or used for these modules. The __COMPAT_ modifier is only required for non-linear branched histories of YANG module versions. Therefore, only the _MAJOR.MINOR.PATCH_ elements of YANG Semver need be considered for IANA-maintained modules.

7.3. Rules Applicable to IANA-Maintained Modules

IANA-maintained YANG modules typically contain only enumerations (enum) and identity definitions, as they represent simple registry mappings. Hence, the most relevant compatibility rules for these modules are:

Editorial Changes:

* Clarifying "description" statements without changing meaning

* Adding or updating "reference" statements

* Fixing typographical errors in description text

* Updating contact information

* Formatting improvements

Backwards-Compatible Changes:

* Adding a new enum value or identity

* Changing status from "current" to "deprecated"

* Removing schema nodes that already have status "obsolete"

Non-Backwards-Compatible Changes:

* Removing an enum value or identity (unless status is "obsolete")

* Changing status to from "current" or "deprecated" to "obsolete"

* Renaming an enum or identity

* Changing the numeric value assigned to an enum

* Modifying "description" statements in a way that changes the semantic meaning

Important: If multiple updates to the registry are made at the same time resulting in a single update to the IANA maintained YANG module then the new module version number is decided by the impact of the most significant change.

7.3.1. Special YANG Considerations for Deprecated Registry Entries

IANA registries and YANG modules use the term _deprecated_ differently:

* In IANA registries, deprecated generally means the value SHOULD NOT be used for new deployments.

* In YANG modules, status deprecated means the definition is still supported (including for new deployments) but it is expected to be obsoleted (or removed) in a future module version.

To avoid confusion, when an IANA registry entry is marked deprecated, the corresponding enum or identity description SHOULD include indicate that the base IANA registry entry is deprecated and therefore the entry SHOULD NOT be used. I.e., the following sentence SHOULD be added to the end of the enum/identity description: This value is deprecated in the base IANA registry which means that its use is NOT RECOMMENDED.

7.4. Process for Updating IANA-Maintained YANG Modules

When a change is made to an IANA registry that has a corresponding YANG module, it is recommended that IANA update the module following these steps:

7.4.1. Step 1: Follow RFC-Defined Rules

First, consult the RFC that defines the IANA registry and its associated YANG module. That RFC may specify:

* Specific rules for how registry entries map to YANG constructs
* Guidance on when and how to update the module
* Contact information for expert reviewers
* Special considerations for the particular registry

Always follow the specific guidance in the RFC that created the registry and module.

7.4.2. Step 2: Identify the Registry Change

Determine exactly what changed in the registry:

* Was a new entry added?
* Was an existing entry modified (description, reference, status)?
* Was an entry deprecated or obsoleted?
* Was an entry removed?
* Were multiple changes made simultaneously?

7.4.3. Step 3: Apply Equivalen Changes to the YANG Module

Update the YANG module to reflect the registry changes. For IANA-maintained modules, this typically involves:

* Adding a new enum value or identity for new registry entries

* Updating description or reference statements for modified entries

* Changing status statements for deprecated or obsoleted entries

* Removing entries only if they are obsolete or if the defining RFC specifies removal

* Add a revision statement (using the current date) describing the change, and a reference, if appropriate.

* _(Optional) include a version statement with the anticipated new version and an rev:non-backwards-compatible statement if it is a backwards-incompatible change._

* _(Optional) Use tooling to format the YANG module, as described in Appendix A.1.1.2._

7.4.4. Step 4: Use Pyang Tooling to Check/Recommend Next Version

Use the tools described in Appendix A.1.1.3 to recommend or check (if provided in step 3) the next module version. Be aware of the tooling limitations, as per Appendix A.2, and sanity check that the version recommeded by the tooling is what is expected based on the changes.

* Add or update the version statement with the correct version

* Add rev:non-backwards-compatible‘ extension if NBC changes have occurred

7.4.5. Step 5: Validate the Module

Use validation tools, as per Appendix A.1.1.1, to ensure the updated module is syntactically correct. Since these modules are simple, just checking with the _pyang_ tool is sufficent, but _yanglint_ (Appendix A.1.2) may be used as an alternative.

7.4.6. Step 6: Seek additional help if Needed

In most cases, the classification will be straightforward. However, if any of the following apply, IANA should seek additional guidance as described in Section 8:

* The change classification is unclear

* The tool output is unexpected or contradictory

* Description changes, where it is not obvious if they change semantic meaning

* Any situation not covered by the guidance above, or examples in Appendix B

7.4.7. Step 7: Publish the Updated Module

Once the module is validated and the version is confirmed:

* Publish the updated module to the IANA website

* The module name should use <module-name>#<version>.yang [I-D.ietf-netmod-yang-module-filename]

* Update any relevant registries or indexes

* Ensure the new version is discoverable and accessible

7.5. Examples

Detailed examples of common scenarios (adding entries, updating references, deprecating entries, etc.) are given in Appendix B.

8. Seeking Additional Guidance

8.1. When to Seek Guidance

The RFC Editor and IANA should contact the YANG Doctors in the following situations:

1. *Classification Uncertainty* - When it’s unclear whether a change is NBC, BC, or Editorial

2. *Tool Disagreement* - When validation tools give unexpected or contradictory results

3. *Description Changes* - When it is unclear whether a description update alters semantic meaning

4. *Unusual Situations* - Any scenario not clearly covered in this document

5. *Registry Restructuring* - Major changes to how a registry is organized

6. *RFC Editor Processing* - When RFC Editor changes may go beyond editorial scope

8.2. How to Seek Guidance

Email the YANG Doctors mailing list and the Operations and Management Area Directors (OPS ADS):

* *Email*: yang-doctors@ietf.org & ops-ads@ietf.org
* *Purpose*: Technical review and guidance on YANG module versioning.
* *Response Time*: Typically 1-2 weeks

When emailing, please include:

* Description of the change or situation
* The affected YANG module and relevant excerpts
* Your proposed classification and version change
* Specific questions or concerns
* Any relevant tool output
* An indication if an urgent reply is required.

The expectation is that the YANG Doctors should reply to the request within the time frame given above, but if a reply isn’t forthcoming then please escalate via the OPS ADS.

8.3. Example Request

Subject: YANG Versioning Question - iana-if-type Update

Dear YANG Doctors,

I need guidance on classifying a change to the iana-if-type module.

Change Description:
The Interface Types registry has updated the description for interface type 6 (ethernet) to clarify that it includes both 10BASE-T and 100BASE-T variants.

Proposed YANG Change:
Update the description statement for the "ethernet" enum to include the clarification.

Question:
Should this be classified as Editorial (PATCH increment) since it’s clarifying existing behavior, or as BC (MINOR increment) because it’s adding new information?

The old description said: "Ethernet interface"
The new description says: "Ethernet interface, including 10BASE-T and 100BASE-T variants"

Current module version: 1.5.0
Proposed version: 1.5.1 (if Editorial) or 1.6.0 (if BC)

Thank you for your guidance.

9. Operational Considerations

This entire document provides operational guidance for the RFC Editor and IANA on how to process and publish YANG modules. The procedures described in Section 6 and Section 7 are designed to ensure consistent and correct versioning of YANG modules across all IETF and IANA publications.

Correct versioning is critical because consumers of YANG modules rely on the semantic version number to understand the compatibility and risk associated with updating to a new module version:

* *PATCH version increments* signal that only editorial changes have been made, indicating very low risk for updates

* *MINOR version increments* signal backwards-compatible additions, indicating that existing implementations will continue to work but new features are available

* *MAJOR version increments* signal non-backwards-compatible changes, indicating that implementations must carefully evaluate the impact before updating

Following the guidance in this document helps ensure that version numbers accurately communicate these compatibility expectations to the YANG module consumer community.

When uncertain about the correct classification or version for a module, the operational recommendation is to choose the more conservative option:

* If uncertain between editorial and backwards-compatible, choose backwards-compatible (MINOR rather than PATCH)

* If uncertain between backwards-compatible and non-backwards-compatible, choose non-backwards-compatible (MAJOR rather than MINOR, and include the NBC extension)

This conservative approach ensures that consumers are appropriately warned about potential compatibility implications, even if the actual risk turns out to be lower than indicated.

10. Security Considerations

This document gives instructions to IANA on how to handle YANG modules that are published in RFCs and also YANG modules that are derived from IANA registries.

Incorrect interpretion of this document could cause incorrect handling or versioning of IANA maintained YANG modules.

This document recommends the usage of various tools. Bugs or attacks on these tools could cause the tools to give incorrect or misleading guidance. In all cases, secondary evaluation of output of the tools should be performed to confirm that they are giving the anticipated results. The _YANG Doctors_ or _Operations and Management Area Directors_ can also be contacted for further advice, if required.

11. IANA Considerations

This document provides operational guidance to IANA and the RFC Editor for managing YANG modules. It does not require IANA to create or modify any registries, nor does it define any new registration procedures.

The guidance in this document is intended to clarify and standardize how IANA processes YANG modules in the "YANG Module Names" registry [IANA-YANG-PARAMETERS] and how IANA maintains YANG modules derived from IANA registries.

IANA should follow the procedures described in Section 6 when processing YANG modules from RFCs and the procedures described in Section 7 when updating IANA-maintained YANG modules.

12. References

12.1. Normative References

[I-D.ietf-netmod-rfc8407bis]
Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-28, 5 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28>.

[I-D.ietf-netmod-yang-module-filename]
Andersson, P., "YANG module file name convention", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-filename-06, 18 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-filename-06>.

[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-15, 18 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-versioning-15>.

[I-D.ietf-netmod-yang-semver]
Clarke, J., Wilton, R., Rahman, R., Lengyel, B., Sterne, J., and B. Claise, "YANG Semantic Versioning", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-semver-24, 29 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-semver-24>.

[IANA-YANG-PARAMETERS]
IANA, "YANG Parameters", n.d., <https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml>.

[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/rfc/rfc6020>.

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

[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/rfc/rfc8340>.

12.2. Informative References

[I-D.boucadair-veloce-yang]
Boucadair, M., "YANG deVELpment PrOCEss & maintenance (VELOCE)", Work in Progress, Internet-Draft, draft-boucadair-veloce-yang-05, 18 September 2025, <https://datatracker.ietf.org/doc/html/draft-boucadair-veloce-yang-05>.

[I-D.ietf-netmod-yang-schema-comparison]
Andersson, P., Wilton, R., and M. Vako, "YANG Schema Comparison", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-schema-comparison-06, 15 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-schema-comparison-06>.

[iana-afnum-registry]
IANA, "Address Family Numbers", n.d., <https://www.iana.org/assignments/address-family-numbers>.

[iana-bgp-parameters]
IANA, "BGP Parameters", n.d., <https://www.iana.org/assignments/bgp-parameters>.

[iana-iftype-registry]
IANA, "Interface Types (ifType) Registry", n.d., <https://www.iana.org/assignments/smi-numbers>.

[iana-safi-registry]
IANA, "SAFI Parameters", n.d., <https://www.iana.org/assignments/safi-parameters>.

Appendix A. Available Tooling

This appendix describes tooling available to assist the RFC Editor and IANA in validating and versioning YANG modules. The tools and capabilities described here reflect the state of tooling as of the publication of this document. Tool capabilities are expected to evolve over time, and newer or improved tools may become available. The NETMOD working group and YANG Doctors can provide updated guidance on current best practices for tooling.

A.1. YANG Validation Tools

A.1.1. pyang

Purpose: pyang is a comprehensive YANG validator and converter tool that can validate syntax, check for backwards-compatible violations, and generate documentation.

Primary Use Cases:

1. Validating YANG module syntax and compliance with IETF conventions

2. Formatting YANG modules in a consistent way prior to publication

3. Suggesting proposed next version when updating a YANG module

4. Generating tree diagrams for documentation

Installation: pyang is available via PyPI (pip install pyang) or from https://github.com/mbj4668/pyang (https://github.com/mbj4668/pyang). It is recommended to periodically check and update the version to pick up bugfixes and new functionality (which could include stricter checks).

Note to RFC Editor and reviewers, some of the tooling enhancements documented here are not yet been merged into the master pyang repository, so depending on publication timing we need to add further references.

To ensure that pyang correctly processes the YANG files, then the correct versions of any YANG module dependencies must also be used, this is often the latest version of the published YANG modules, but if a draft contains a set of YANG modules, or if there are set of drafts with YANG modules being published together then all the YANG modules in those drafts MUST be extracted together and validated. These dependent YANG modules can either be stored in the same directory as the YANG module being validated/checked, or they can be stored in a separate directory and passed using the -p argument to provide a path to the directory.

A.1.1.1. Basic YANG Syntax Validation

This command below validates the module syntax and checks compliance with IETF-specific conventions. The output will show any errors or warnings. IETF and IANA modules SHOULD have no errors or warnings before publication.

```
pyang --ietf --strict --max-line-length=69 -Werror -p <dep-module-directory> ietf-modu
le-name.yang
```

A.1.1.2. Consistent formatting of YANG modules

Pyang can be used to reformat a YANG file, particularly fixing line length issues and correcting any indentation mistakes. Some complex expressions, e.g., must, when and path statements may not be automatically split to the correct line length and may need to be done manually. Running the basic syntax validation command on the output file will indicate whether any further manual line folding is required.

```
pyang -f yang --yang-line-length=69 --yang-canonical -Werror -p <dep-module-directory>
 -o <output-file> <treeOpts> ietf-module-name.yang
```

A.1.1.3. Suggesting proposed next version when updating a YANG module

Pyang can be used to compare the changes between two YANG module versions and either validate that a suitable next version number has been used, or to suggest what the appropriate next version should be, or if further manual checks should be performed, e.g., for changes to description statements.

If the new module version already includes a version statement for the latest version then it will check whether the version used matches the expected version that has been calculated based on the changes between the two module versions. Otherwise, if the latest version does not contain a version statement then it will suggest the new version that should be used.

```
pyang --check-update-semver --check-update-from module-name@old-version.yang module-na
me@new-version.yang
```

Interpreting Tool Output

The command output:

* will suggested the next YANG Semver, based on the changes.

* indicate whether the rev:non-backwards-compatible annotation is needed.

* highlight any non-backwards-compatible changes, which are reported as errors.

* indicate if there are changes to any statements, e.g., description, that require further analysis to decide whether a semantic change has occurrred and hence if the change is not-backwards-compatible rather than editorial.

Example Tool Output 1:

This example illustrates what the expected output would be for an update to a YANG module that is derived from an IANA registry update that defines a new enum or identity. This is a minor version change and hence the version change recommended by the tool is from 1.0.0 → 1.1.0.

```
SUGGESTED-NEXT-YANG-SEMVER: 1.1.0
```

Example Tool Output 2:

This example output below due to deleting an enum entry indicates an NBC change has occurred. Hence the tool recommends a major version number change from 1.0.0 → 2.0.0 and the addition of the rev:non-backwards-compatible statement.

```
SUGGESTED-NEXT-YANG-SEMVER: 2.0.0
NBC-CHANGE(S):
iana-ssh-mac-algs@2026-03-06.yang:45: error: rev:non-backwards-compatible is required for this revision
iana-ssh-mac-algs@2026-03-06.yang:62: error: the enum 'AEAD_AES_256_GCM', defined at iana-ssh-mac-algs@2024-10-16.yang:109 is illegally removed or marked obsolete
iana-ssh-mac-algs@2026-03-06.yang:62: error: the value for enum 'hmac-sha2-256', has changed from 7 to 6 (RFC 7950: sec. 11, p5, bullet 1)
iana-ssh-mac-algs@2026-03-06.yang:62: error: the value for enum 'hmac-sha2-512', has changed from 8 to 7 (RFC 7950: sec. 11, p5, bullet 1)
```

Example Tool Output 3:

This example output is for a change to a description statement. The tool output suggests a version change from 1.0.0 → 1.0.1, which is correct if there is no change in semantics. It is also highlights that it may be necessary to consult with the authors to determine if a semantic change has occurred (if it is not obvious). If a semantic change has occurred, then the version change should be from 1.0.0 → 2.0.0.

```
SUGGESTED-NEXT-YANG-SEMVER: 1.0.1
POSSIBLE-NBC-CHANGE(S):
Consult document authors and YANG Doctors.
iana-ssh-mac-algs@2025-03-17.yang:138: warning: the description change may have change
d the semantics of the node
```

This example output indicates an NBC change has occurred, which is indicated by a major version number change and the addition of the rev:non-backwards-compatible statement.

A.1.1.4. Generating tree diagrams for documentation

Pyang can be used to generate tree diagram output that conforms to [RFC8340]. The command below generates the tree diagram for ietf-module-name.yang, limited to a line length of 69 characters and writes it into the specified output-file. The tree-options is based on the options in pyang that are prefixed with --tree and can be seen by running pyang --help. Common options may include printing out grouping (--tree-print-groupings) or printing out structures (--tree-print-structures).

```
pyang -f tree --tree-line-length=69 -Werror -p <dep-module-directory> -o <output-file>
 <tree-options> ietf-module-name.yang
```

A.1.2. yanglint

Purpose: yanglint is a YANG validator and data manipulation tool from the libyang project, useful for validating modules and instance data.

Primary Use Cases:

* Validating YANG module syntax

* Checking cross-module dependencies

* Validating instance data against YANG schemas

Installation: yanglint is part of libyang, available from https://github.com/CESNET/libyang (https://github.com/CESNET/libyang)

Syntax Validation:

```
yanglint -p /path/to/yang/modules module-name.yang
```

The -p option specifies a directory containing imported modules.

This command validates the module syntax. No output indicates successful validation; errors will be displayed if found.

A.1.3. YANG Catalog Tools

Purpose: The YANG Catalog (https://www.yangcatalog.org (https://www.yangcatalog.org)) provides online tools for module validation, comparison, and discovery.

Primary Use Cases:

* Validating YANG modules without local tool installation
* Comparing different versions of modules
* Viewing module dependencies and impact analysis
* Searching for existing modules and versions

Usage:

Access the web interface at https://www.yangcatalog.org (https://www.yangcatalog.org) and use the "Validator" and "YANG Impact Analysis" tools.

Interpreting Results:

The online tools provide visual feedback on validation results and module comparisons. The impact analysis tool can show which other modules depend on a given module, helping assess the impact of changes.

A.2. Tool Limitations

While tools are valuable for YANG module validation and versioning, they have a couple of limitations relevant to their usage here:

Limitation 1: Cannot Always Distinguish Editorial from BC/NBC Changes

Current tools cannot determine whether a description change is purely editorial (clarifying existing meaning), backwards-incompatible (changing meaning). Human or AI judgment is required to make this distinction.

Example: Changing "Ethernet interface" to "Ethernet interface, includes all Ethernet interface speeds" could be editorial (if those variants were always included). But changing an "ip" type from a description saying "IPv4 address or IPv6 address" to just "IPv4 address" would be regarded an a NBC change because the scope of the type has clearly changed and may impact users of that type.

Limitation 2: May Produce False Positives or False Negatives

Tool implementations may have bugs or may not cover all edge cases in the YANG versioning rules. Generally, in any ambiguous cases, the tools are being designed to either explicitly flag this or choose the more impactful option. It is assumed to be better for clients to potentially flag an NBC change that is not really there than to mistakenly flag an NBC change as only being a BC change.

Hence the recommendation is to always review tool output critically and seek additional review (Section 8) when uncertainty remains.

Appendix B. Summary of IANA Registry Action Scenarios

This appendix provides a comprehensive reference of common scenarios encountered when updating IANA-maintained YANG module derived from IANA registries. Each scenario describes the registry action, the corresponding YANG module change, the classification (NBC/BC/Editorial), the version change required, and whether the rev:non-backwards-compatible extension must be added.

B.1. Quick Reference Table

The assumption is that the YANG module uses the registry entry name, numeric identifier, description, status, and any reference fields as part of the YANG entries. If additional fields from the registry are used in the YANG module (e.g., perhaps the YANG description is constucted from multiple registry fields) then any changes to those fields will require a new version of the YANG module to be published and an appropriate new version number, chosen based on the actual change to the YANG module.

Important Principle: The source or trigger of a change (errata, new RFC, registry update, expert review, etc.) does NOT determine whether it is NBC, BC, or Editorial. What matters is the resultant change made to the YANG module content.

Registry Action	YANG Change	Classification	Version	NBC Ext
Add new registration	Add enum/ identity	BC	MINOR	No
Update reference (Draft → RFC)	Update reference	Editorial	PATCH	No
Update reference (obsoleted RFC)	Update reference	Editorial	PATCH	No
Add additional reference	Update reference	Editorial	PATCH	No
Change or remove reference	Update reference	Editorial	PATCH	No
Update description (clarify)	Update description	Editorial	PATCH	No
Update description (change meaning)	Update description	NBC	MAJOR	Yes
Deprecate entry (keep name)	status deprecated	BC	MINOR	No
Obsolete entry	status obsolete	NBC	MAJOR	Yes
Rename entry	Change identifier	NBC	MAJOR	Yes
Remove entry completely	Remove enum/ identity	NBC	MAJOR	Yes
Change value number	Change value	NBC	MAJOR	Yes
Reuse old value	Same as	BC	Minor	No

(previously removed)	adding new entry			
Add footnote	Optionally update description	Editorial	PATCH	No
Non-YANG field changes	Depends on how the module is updated	Analyze	Varies	Maybe
Errata	Depends on content	Analyze	Varies	Maybe
Early alloc expired (left as-is)	No change	N/A	None	No
Early alloc expired (removed)	Follow removal rules	NBC	MAJOR	Yes
Revive expired (removed) allocation	Add enum/ identity	BC	MINOR	No

Table 1: Registry Action → YANG Module Update Reference Table

Key:

* *BC* = Backwards-Compatible; *NBC* = Non-Backwards-Compatible

* *MAJOR/MINOR/PATCH* refer to the YANG Semver version components

* *NBC Ext* = Whether rev:non-backwards-compatible extension is required

* *Varies* or *Maybe* indicates the specific change must be analyzed using the detailed scenarios below

B.2. Detailed Common Scenarios

Note: The following scenarios only contain snippets of YANG to illustrate the change being made and are not intended to represent complete example modules, or be able to compile or validate.

B.2.1. Scenario 1: Adding a New Registry Entry

Registry Action: A new entry is added to an IANA registry.

YANG Module Change: Add a new enum value or identity.

Classification: Backwards-Compatible (BC)

Version Change: Increment MINOR version (e.g., 1.0.0 → 1.1.0)

NBC Extension Required: No

Rationale: Adding a new type is backwards compatible because it cannot break backwards-compatability of existing implementations.

Example:

Previous version, _1.0.0_:

```
revision 2025-11-01 {
  ysv:version "1.0.0";
  description "Initial revision.";
}

typedef interface-type {
  type enumeration {
    enum ethernet {
      value 6;
      description "Ethernet interface";
    }
  }
}
```

New version, _1.1.0_, after addition of ’wifi’ enum type:

```
   revision 2025-11-15 {
     ysv:version "1.1.0";
     description "Added 'wifi' (71).";
   }
   revision 2025-11-01 {
     ysv:version "1.0.0";
     description "Initial revision.";
   }

   typedef interface-type {
     type enumeration {
       enum ethernet {
         value 6;
         description "Ethernet interface";
       }
       enum wifi {
         value 71;
         description "IEEE 802.11 wireless interface";
       }
     }
   }
```

B.2.2. Scenario 2: Updating References

Registry Action: A reference is updated (e.g., RFC obsoleted, additional reference added, draft reference changed to RFC number).

YANG Module Change: Update the "reference" statement.

Classification: Editorial

Version Change: Increment PATCH version (e.g., 2.3.0 → 2.3.1)

NBC Extension Required: No

Rationale: Reference statements may be added or updated without affecting compatibility of existing implementations.

Example:

Previous version, _2.3.0_:

```
revision 2025-11-01 {
  ysv:version "2.3.0";
  description "Added 'foo' (42).";
}

typedef interface-type {
  type enumeration {
    ...
    enum foo {
      value 42;
      description "Foo interface type.";
      reference "RFC 1234";
    }
  }
}
```

New version (_2.3.1_) after the reference for 'foo' is updated to RFC 5678:

```
revision 2025-11-15 {
  ysv:version "2.3.1";
  description "Updated reference for 'foo' (42).";
}
revision 2025-11-01 {
  ysv:version "2.3.0";
  description "Added 'foo' (42).";
}

typedef interface-type {
  type enumeration {
    ...
    enum foo {
      value 42;
      description "Foo interface type.";
      reference "RFC 5678 (obsoletes RFC 1234)";
    }
  }
}
```

B.2.3. Scenario 3: Deprecating a Registry Entry

Registry Action: A registry entry is marked as deprecated (name and description remain visible).

YANG Module Change: Change status from "current" to "deprecated".

Classification: Backwards-Compatible (BC)

Version Change: Increment MINOR version (e.g., 2.3.1 → 2.4.0)

NBC Extension Required: No

Rationale: Changing status to deprecated is backwards-compatible but the description must also be updated to highlight the difference in meaning between IANA registries and YANG modules.

Example:

Previous version, _2.3.1_:

```
revision 2025-11-15 {
  ysv:version "2.3.1";
  description "Updated reference 'foo' (42).";
}

typedef interface-type {
  type enumeration {
    ...
    enum oldtype {
      value 99;
      description "Old interface type";
    }
  }
}
```

New version (_2.4.0_) after deprecation:

```
   revision 2025-11-23 {
     ysv:version "2.4.0";
     description "Deprecated 'oldtype' (99).";
   }
   revision 2025-11-15 {
     ysv:version "2.3.1";
     description "Updated reference 'foo' (42).";
   }

   typedef interface-type {
     type enumeration {
       ...
       enum oldtype {
         value 99;
         status deprecated;
         description
           "Old interface type. This value is deprecated in the base IANA
           registry which means that its use is NOT RECOMMENDED.";
       }
     }
   }
```

B.2.4. Scenario 4: Obsoleting a Registry Entry

Registry Action: A registry entry is marked as obsolete.

YANG Module Change: Change status from "deprecated" (or "current") to "obsolete".

Classification: Non-Backwards-Compatible (NBC)

Version Change: Increment MAJOR version (e.g., 2.4.0 → 3.0.0)

NBC Extension Required: Yes

Rationale: Changing status to obsolete indicates the value MUST NOT be used, breaking compatibility. Note, the description comment about deprecation can also be removed.

Example:

Previous version (_2.4.0_):

```
   revision 2025-11-23 {
     ysv:version "2.4.0";
     description
       "Deprecated 'oldtype' (99).";
   }

   typedef interface-type {
     type enumeration {
       ...
       enum oldtype {
         value 99;
         status deprecated;
         description
           "Old interface type. This value is deprecated in the base IANA
           registry which means that its use is NOT RECOMMENDED.";
       }
     }
   }
```

New version (_3.0.0_) after obsoletion:

```
   revision 2025-11-30 {
     ysv:version "3.0.0";
     rev:non-backwards-compatible;
     description
       "Obsoleted 'oldtype' (99).";
   }
   revision 2025-11-23 {
     ysv:version "2.4.0";
     description
       "Deprecated 'oldtype' (99).";
   }

   typedef interface-type {
     type enumeration {
       ...
       enum oldtype {
         value 99;
         status obsolete;
         description
           "Old interface type.";
       }
     }
   }
```

B.2.5. Scenario 5: Removing a Registry Entry Completely

Registry Action: A registry entry is removed with no trace.

YANG Module Change: Remove the enum or identity from the module.

Classification: Non-Backwards-Compatible (NBC)

Version Change: Increment MAJOR version (e.g., 2.2.0 → 3.0.0)

NBC Extension Required: Yes

Rationale: Removing a schema node is NBC per Section 3.1.2.1 of [I-D.ietf-netmod-yang-module-versioning].

Example:

Previous version (_2.2.0_):

```
revision 2026-02-01 {
  ysv:version "2.2.0";
  description
    "Added 'legacy-wireless' identity.";
}

identity interface-type {
  description
    "Base identity for interface types.";
}

identity legacy-wireless {
  base interface-type;
  description
    "Legacy wireless interface.";
}
```

New version (_3.0.0_) after removal:

```
   revision 2026-03-15 {
     ysv:version "3.0.0";
     rev:non-backwards-compatible;
     description
       "Removed 'legacy-wireless' identity.";
   }
   revision 2026-02-01 {
     ysv:version "2.2.0";
     description
       "Added 'legacy-wireless' identity.";
   }

   identity interface-type {
     description "Base identity for interface types.";
   }
```

B.2.6. Scenario 6: Renaming a Registry Entry

Registry Action: The name of a registry entry is changed.

YANG Module Change: Change the enum or identity identifier.

Classification: Non-Backwards-Compatible (NBC)

Version Change: Increment MAJOR version (e.g., 3.1.0 → 4.0.0)

NBC Extension Required: Yes

Rationale: Renaming breaks programmatic references to the identifier.

Example:

Previous version (_3.1.0_):

```
   revision 2026-04-01 {
     ysv:version "3.1.0";
     description "Added 'old-gre' identity.";
   }

   identity tunnel-type {
     description "Base identity for tunnel types.";
   }

   identity old-gre {
     base tunnel-type;
     description "GRE tunnel (legacy name).";
   }
```

New version (_4.0.0_) after rename:

```
revision 2026-05-01 {
  ysv:version "4.0.0";
  rev:non-backwards-compatible;
  description "Renamed 'old-gre' identity to 'gre'.";
}
revision 2026-04-01 {
  ysv:version "3.1.0";
  description "Added old-gre identity.";
}

identity tunnel-type {
  description "Base identity for tunnel types.";
}

identity gre {
  base tunnel-type;
  description "GRE tunnel.";
}
```

B.2.7. Scenario 7: Changing a Value Number

Registry Action: The numeric value assigned to a registry entry is changed.

YANG Module Change: Change the value assigned to an enum.

Classification: Non-Backwards-Compatible (NBC)

Version Change: Increment MAJOR version (e.g., 2.3.0 → 3.0.0)

NBC Extension Required: Yes

Rationale: Changing values breaks compatibility for implementations using those values.

Example:

Previous version (_2.3.0_):

```
   revision 2026-01-10 {
     ysv:version "2.3.0";
     description "Added multiple identities.";
   }

   typedef interface-type {
     type enumeration {
       enum fastether {
         value 210;
         description "Fast Ethernet interface.";
       }
       enum atm {
         value 211;
         description "ATM making a surprising comeback";
       }
     }
   }
```

New version (_3.0.0_) after value change:

```
   revision 2026-02-20 {
     ysv:version "3.0.0";
     rev:non-backwards-compatible;
     description "Changed 'fastether' value to 215.";
   }
   revision 2026-01-10 {
     ysv:version "2.3.0";
     description "Added multiple identiies.";
   }

   typedef interface-type {
     type enumeration {
       enum fastether {
         value 215;
         description "Fast Ethernet interface.";
       }
     }
   }
```

B.2.8. Scenario 8: Updating Description (Clarification)

Registry Action: Description is updated to clarify existing behavior without changing meaning.

YANG Module Change: Update the description statement.

Classification: Editorial

Version Change: Increment PATCH version (e.g., 2.1.3 → 2.1.4)

NBC Extension Required: No

Example:

Previous version (_2.1.3_):

```
revision 2026-02-05 {
  ysv:version "2.1.3";
  description "Clarified description for 'foo'.";
}

identity ethernet {
  base interface-type;
  description "Ethernet interface.";
}
```

New version (_2.1.4_) after clarification:

```
revision 2026-02-12 {
  ysv:version "2.1.4";
  description "Clarified description for 'ethernet'.";
}
revision 2026-02-05 {
  ysv:version "2.1.3";
  description "Clarified description for 'foo'.";
}

identity ethernet {
  base interface-type;
  description
    "Ethernet interface, includes 10BASE-T, 100BASE-T, and
     1000BASE-T variants.";
}
```

B.2.9. Scenario 9: Updating Description (Semantic Change)

Registry Action: Description is updated in a way that changes the semantic meaning or behavior.

YANG Module Change: Update the description statement.

Classification: Non-Backwards-Compatible (NBC)

Version Change: Increment MAJOR version (e.g., 2.2.0 → 3.0.0)

NBC Extension Required: Yes

Example:

Previous version (_2.2.0_):

```
revision 2026-03-01 {
  ysv:version "2.2.0";
  description "Defined ’ip’ identity.";
}

identity ip {
  base interface-type;
  description "Interface supports IPv4.";
}
```

New version (_3.0.0_) after semantic change:

```
revision 2026-04-01 {
  ysv:version "3.0.0";
  rev:non-backwards-compatible;
  description "Changed description for ’ip’ identity";
}
revision 2026-03-01 {
  ysv:version "2.2.0";
  description "Defined ’ip’ identity.";
}

identity ipv {
  base interface-type;
  description "Interface supports IPv4 and IPv6.";
}
```

B.2.10. Scenario 10: Handling Errata

Registry Action: An errata report is filed for the registry or module.

YANG Module Change: Depends on the specific errata content.

Classification: Analyze the actual change, not the source (errata vs. new RFC does not determine classification).

Version Change: Follow the rules based on the actual change being made to the IANA registry entry.

NBC Extension Required: May be required depending on the change.

Examples:

* Errata fixes typo in description → Editorial / PATCH

* Errata adds missing enum → BC / MINOR

* Errata corrects wrong value assignment → NBC / MAJOR

Acknowledgments

The creation of this document was motivated by questions from IANA team members Amanda Baber and Sabrina Tanamal regarding the correct handling of YANG module versioning. This was followed by a productive in-person discussion at IETF 124 between members of the YANG versioning design team, the IANA team, NETMOD working group chairs, RFC Editor representatives, and the OPS Area Director.

Participants in that meeting included: Amanda Baber, Jason Sterne, Joe Clarke, Kent Watsen, Lou Berger, Mahesh Jethanandani, Per Andersson, Reshad Rahman, Rob Wilton, and Sabrina Tanamal.

Special thanks to Joe Clarke for his presentation on YANG versioning tooling at IETF 124, which informed the tooling guidance in Appendix A.

The authors thank the RFC Editor and IANA teams for their collaboration in refining the operational procedures described in this document.

The authors also thank the NETMOD working group for their extensive work on YANG versioning specifications that form the foundation of this guidance, including the module versioning framework, semantic versioning, and associated tooling.

The initial substantive revision of this document used Claude Sonnet 4.5 to create prose and examples, which have been subsequently reviewed and refined by the YANG Versioning design team and the NETMOD working group.

Author’s Address

Robert Wilton
Cisco
Email: rwilton@cisco.com