NETMOD Versioning Solution Overview

NETMOD WG

November 2020

Presenting on behalf of the authors:
Jan Lindblad

Agenda

This presentation:

Solution Overview – Jan Lindblad

Following presentations on individual drafts:

YANG Module Versioning – Jan Lindblad

YANG Semantic Versioning – Jan Lindblad

YANG Packages – Bo Wu

Recap - YANG Versioning Solution Overview

Complete solution consists of five drafts:

1. Updated YANG Module Revision Handling:

   Can notify of nbc changes between module revisions, allows branched revision history, revision-labels

2. Module semantic version number scheme:

   Allows use of YANG semver for module revision-labels and package versioning

3. Versioned YANG packages:

   Versioning at the schema level rather than individual modules

4. Protocol operations for package version selection:

   Devices can support multiple schema versions, clients can select for session

5. YANG schema comparison tooling:

   Tooling to algorithmically compare module or schema revisions

General update on weekly Versioning calls

- Tried to progress Versioning, but a bit stuck
  - Working on packages may resolve issues
- Bumped the comparison & versioning selection drafts
  - Updated to newest RFC format xml2rfc v3
- Spent lots of time on Packages
  - Lots left to go
- Focus remains on Packages, likely for the next while

Next Steps

- Authors + interested parties continue to meet on a weekly call to progress this work
  - Meetings are open to all
  - key issues are brought back to WG mailing list
  - Weekly meeting is currently on Tues @2pm UK time / 9am Eastern.
    Thanks to the authors and contributors for their regular attendance
- Issues tracked in github, discussed on WG mail list
  https://github.com/netmod-wg/yang-ver-dt
- Phase work on the documents:
  - module versioning, yang semver and packages? first (get to last call)
  - version selection and schema comparison to follow

YANG Module Versioning

draft-ietf-netmod-yang-module-versioning

NETMOD WG

November 2020

Presenting on behalf of authors:
Jan Lindblad

Main issues discussed since IETF108
(still open)

Insignificant whitespace changes

https://github.com/netmod-wg/yang-ver-dt/issues/8

Many of the authors/contributors wanted insignificant whitespace changes to result in a new revision-label. Makes validation of a module with a checksum easier (no normalization needed).

WG feedback: insignificant changes should not require a new revision-label since this is currently allowed.

Email thread: https://mailarchive.ietf.org/arch/msg/netmod/1OWNbTb1riSzRupBei-T9FQprzI/

Next step: further discussions needed

How NBC changes can break imports

https://github.com/netmod-wg/yang-ver-dt/issues/11

Extension *revision-or-derived* reduces set of importable revisions to those which are derived from a particular revision:

1) Consider module A (1.0.0) which imports module B using “2.0.0 or derived” and that there is no revision-label with MAJOR version > 2. This means A will be importing rev 2.Y.Z of module B.
2) If new revision 3.0.0 of module B is created (NBC changes), module A may end up importing 3.0.0 and this **could** break clients using module A

Email thread: https://mailarchive.ietf.org/arch/msg/netmod/dGQX4jeQWjPT1TqPjk8_yjVJhFM/

Next step: decide whether we need extension *revision-or-compatible-derived*

Impact of changing import statement

https://github.com/netmod-wg/yang-ver-dt/issues/4

1) Consider module A (1.0.0) which imports module B using “2.0.0 or derived” and that there is no revision-label with MAJOR version > 2. This means A will be importing rev 2.Y.Z of module B.
2) If new revision 3.0.0 of module B is created, module A will end up with 2.Y.Z or 3.0.0. If we have *revision-or-compatible-derived*, 3.0.0 can not be used
3) If module A is modified to importing module B using “3.0.0 or derived”, is this a BC or NBC change?

Option A: depending on the impact on module A, the change is deemed BC or NBC.

Pros: addresses change to module A in step 3 appropriately depending on impact.

Cons: Is this harder on tooling?

Option B: the change to import is always considered to be a BC change to the importing module. The revision-label of the corresponding YANG package is updated according to the impact on the package’s schema.

Pros: consistent way of handling changes to import statements and to imported modules

Cons: revision-label of a module may not correctly indicate the impact of a change to import-stmt, i.e. Looking at revision label of a module in isolation may not give the whole picture

Impact of changing import statement (contd)

- Authors/contributors split on this topic
- Rough consensus is option B (consider all import-stmt changes as BC)
- Another consideration is whether a module's revision-label should represent the module's file contents only or the module's schema (i.e. imports also)
- **We would like to hear from the WG, further discussions needed**
- Email thread: https://mailarchive.ietf.org/arch/msg/netmod/_lvHJzsFWAM3dJxq_COHz7CG5Tg/

Next Steps

Next steps

- Resolve last outstanding issues before IETF110

YANG Semantic Versioning

draft-ietf-netmod-yang-semver

NETMOD WG

November 2020

Presenting on behalf of the authors:
Jan Lindblad

Current Status

- No substantive changes since IETF108
- Authors have been focusing on packages in these past few months
- Open Issues:
  - Pre-release version precedence
  - What to advise IANA
  - YANG Semver mandatory for IETF modules

Pre-release version precedence

GitHub Issue #60: If the MAJOR, MINOR, PATCH of a pre-release version doesn't change or jumps forward, the pre-release precedence rules are well-established based on semver.org. However, if a module is initially going to be a MAJOR version ahead but during development it's reverted to only a MINOR version increment, the precedence rules break. For example:

1.0.0 < 2.0.0-pre-release <broken!> 1.1.0-pre-release < 1.1.0

The above could be fixed simply by saying pre-releases don't have implied precedence and the module lineage should be used instead. But then we lose any semantic meaning in the revision-labels. Though, with the draft names in the pre-release strings, there is a direct link back to the draft, which has history in Data Tracker.

Another thought is to say that revision-labels can never go backwards. That if you decide on 2.0.0 as the next string, that the final product will be 2.0.0. This can work because it will most likely trigger authors to use the next MINOR (or even PATCH) version throughout development until the final release. Then, and only then is a true YANG semver assigned based on the totality of the development.

Recommendations to IANA

- GitHub Issue #59 (also applies to overall versioning)
- What guidance should be given to IANA around adding “nbc-change” and any revision-label to a document?
- **Recommendation: Authors with the help of YANG Doc would be best suited for this. IANA should not be adding labels or deciding on nbc-changes**

YANG Semver mandatory for IETF modules

GitHub Issue #45: Should all newly published IETF YANG modules include a yang-semver revision-label?

Authors’ proposal: yes

- helps readers understand “at a glance” if two revisions of a module are backwards compatible