IETF YANG Versioning Packages Update

draft-ietf-netmod-yang-packages-06

NETMOD WG IETF 124, Nov 2025

Presenting (on behalf of authors): Rob Wilton

Main contributors: Jason Sterne, Reshad Rahman, Joe Clarke

Solution Overview

Solution Overview draft: draft-ietf-netmod-yang-versioning-reqs-11

Proposed full solution is made up of 6 drafts (at different levels of maturity):

1. Updated YANG Module Revision Handling (draft-ietf-netmod-yang-module-versioning-15)
2. Module semantic version number scheme (draft-ietf-netmod-yang-semver-24)
3. Yang module filename convention (draft-ietf-netmod-yang-module-filename-03)

4. **Versioned YANG packages (draft-ietf-netmod-yang-packages-06)**
5. YANG schema comparison tooling (draft-ietf-netmod-yang-schema-comparison-05)
6. Protocol operations for package version selection (draft-ietf-netmod-yang-ver-selection-00)

Solution Overview

Solution Overview draft: draft-ietf-netmod-yang-versioning-reqs-11

Proposed full solution is made up of 6 drafts (at different levels of maturity):

1. Updated YANG Module Revision Handling (draft-ietf-netmod-yang-module-versioning-15)
2. Module semantic version number scheme (draft-ietf-netmod-yang-semver-24)
3. Yang module filename convention (draft-ietf-netmod-yang-module-filename-03)

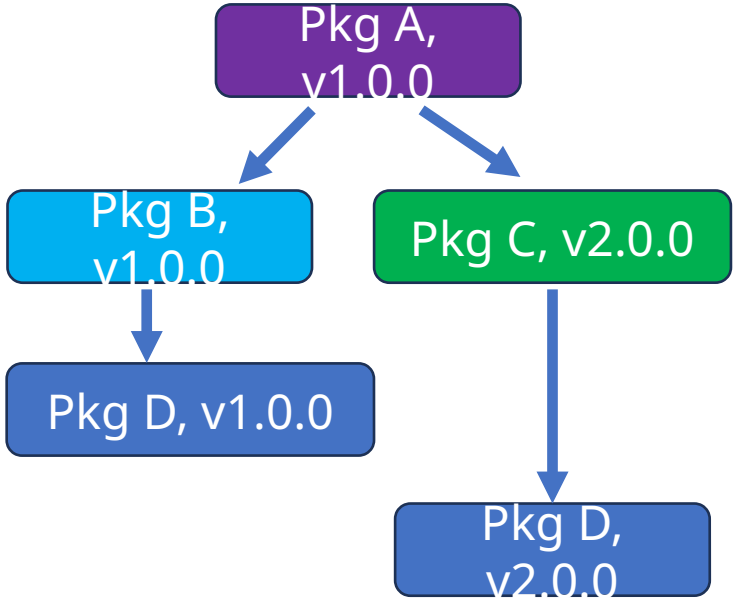
Potentially 1b/2b: Guidelines to IANA for versioning

4. **Versioned YANG**
5. YANG schema comparison tooling (draft-ietf-netmod-yang-s
6. Protocol operations for package version selection (draft-ietf-netmod-yang-ver-selection-00)

Potentially 4b: IETF Packages draft

YANG Packages Recap 101

- YANG Packages are an alternative mechanism for managing sets of YANG modules, features and mount
- They can be defined hierarchically
- A package logically resolves into a YANG schema – like the information provided by YANG library
- Packages are versioned, using YANG Semver
- Packages should be made available offline and referenced by a URL
- Tooling to allow package versions to be compared/queried



Recap of decisions made before/at IETF 121:

- Overriding idea: Keep it simple
- No differentiation between API vs implementation packages
- Packages just define what is there (like YANG library)
- No optionality in package definitions
- No extra flag for deviations
- The model supports mounted schema, needed for IETF packages
- Don’t include checksums

Recap of decisions made before/at IETF 121:

- Overriding idea: Keep it simple
- No differentiat… betwe… API… …plem…ntation packages
- Packages just def… …NC library)
- No optionality …
- No extra fla… for de…
- The model support… mo… …ed s… …ma, …ded for IETF packages
- Don’t include checksums

No new draft version since IETF 123

Recap of decisions made at IETF 122:

1. Write a simple package compiler
   - E.g., trial some IETF/OC/Vendor packages
2. Start separate draft(s) to guide IETF package design & lifecycle
   - Lifecycle management overlaps with NEMOPS discussion for modules
3. Test some packages to gain further experience

Recap of decisions made at IETF 122:

1. Write a simple package compiler
   - E.g., trial some IETF/OC/Vendor packages
2. Start separate draft(s) to guide IETF package design & lifecycle
   - Lifecycle management overlaps with NEMOPS discussion for modules
3. Test some packages to gain further experience

Continuing!
(Not published)

Started!
(Not published)

Still TODO

High Level Status

Slower progress since IETF 123 (WGLC issues + short on time)

- Still working on implementation
- Some issues still to resolve
- Doc is continuing to be restructured to improve readability:
  - Sections/text has been moved around
  - Trying to remove extraneous text and simplify what is there

Significant Changes (since 12

1. Removed support for tags
2. Finishing work on adding YANG mount support
3. Default automatic choice between included module versions
4. Added ability to exclude features or modules
   - Also considered excluding packages and what it means to “implement a package”
   - Lot of discussion related to package resolution
5. Considered relationship to library datastore schema + bugfixes
6. Mount path type - YPath

YANG Package Schema

```
grouping yang-pkg-instance:
    +-- name                    pkg-name
    +-- version                 pkg-version
    +-- timestamp?              yang:date-and-time
    +-- organization?           string
    +-- contact?                string
    +-- description?            string
    +-- reference?              string
    +-- complete?               boolean
    +-- includes
    |   +-- package* [name]
    |   |   +-- name          pkg-name
    |   |   +-- version       pkg-version
    |   |   +-- location*     inet:uri
    |   +-- module* [name]
    |   |   +-- name          yang:yang-identifier
    |   |   +-- version       version-or-rev-date
    |   |   +-- location*     inet:uri
    |   |   +-- submodule* [name]
    |   |      +-- name?         yang:yang-identifier
    |   |      +-- version       version-or-rev-date
    |   |      +-- location*     inet:uri
    |   +-- import-only-module* [name version]
    |      +-- name                  yang:yang-
identifier
    |      +-- version               version-or-rev-
date
    |      +-- replaces-version*     version-or-rev-
date
    |      +-- location*             inet:uri
    |      +-- submodule* [name]
    |         +-- name?         yang:yang-identifier
    |         +-- version       version-or-rev-date
    |         +-- location*     inet:uri
    +-- excludes
    |   +-- module*                  pkg-name
    |   +-- import-only-module*      pkg-name
    +-- mandatory-features
    |   +-- include*   scoped-feature
    |   +-- exclude*   scoped-feature
    +-- mounts* [mount-path]
        +-- mount-path?          mount-ypath
        +-- package* [name]
        |   +-- name                   pkg-name
        |   +-- version                pkg-version
        |   +-- location*              inet:uri
        |   +-- replaces-package*      pkg-name
        +-- parent-reference*    mount-ypath
```

1. Removes support for “tags”

4. Includes / excludes

2. New type: mount-ypath
Added: replaces-package
Added: parent-reference

Next Steps

Meeting weekly on Tuesdays, 9 am US Eastern Time.

On the packages draft:

1. Finish restructuring sections, finish examples
2. Continue working on checking/resolving outstanding issues (TODOs in doc & github issues)

Related work:

- Make progress on tooling – hoping that I can open source
- Make progress on IETF models drafts
- Do more testing with IETF, OC, vendors models

Any Questions/Comments?

Backup slides

Additional details of changes presented in IETF 123

1. Remove support for tags

Some discussions about:

1. What happens to tags from included packages - how do you merge?
2. What about tags at mount points?
3. How are tags going to be used?

Conclusion:

- **Remove unwanted & unnecessary complexity**
- Potential future enhancement - **if there is a concrete use case**

2. Finishing support for Schema Mount

Note:

- *Mount support is required to fully support IETF YANG models*
- *Partially allows for design time mounts (i.e., publish package alongside module)*

Added:

- “parent-reference” – allows access to nodes outside the ”mount jail”
- “replaces-package” – needed to ‘wrap package’ to remove modules or features
  - *TODO – Perhaps will rename to “wrap-package”*

3. Automatic resolution between module versions

When a package A includes two packages B and C, that both implement different versions of a module *M* then A must choose a single version of *M*

Previously:

- Package A must resolve by explicitly *including* the desired module version

Now:

- Will automatically resolve to latest module version. Uses YANG Semver if either have a YANG Semver version, or latest revision-date otherwise
- Package A can still choose by *explicitly including* the desired version of module *M*

4. Added ability to exclude modules or mandatory-features

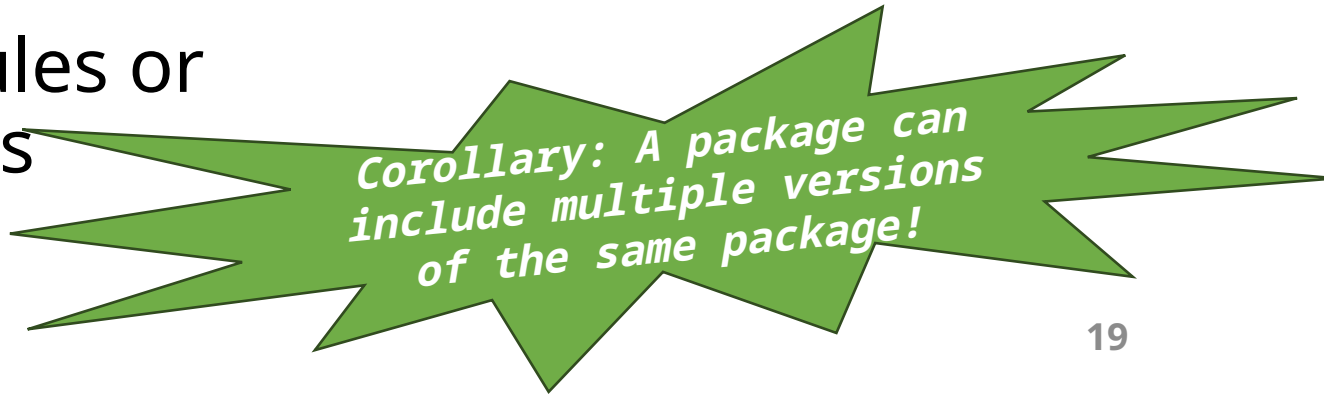
Package A can include Package B, but then exclude modules, import-only-modules, or mandatory-features

- Helpful for device conformance and advertising packages
- Clearer than implementing the module and deviating all data nodes
- Useful if implementing large standard/industry packages (e.g., OpenConfig or an IETF ‘router’ package)
  - Alternative implementation choices exist (e.g., don’t implement the top-level OC or IETF ’router’ package)
- *Can still use deviations* (i.e., add modules with deviation statements)*

4b). Package resolution

Basic steps:

- Recursively resolve each included package (to modules, features, & mount points), then
- Merge included resolved packages together using automatic resolution for conflicting module versions, then
- overwrite with any included modules/mounts/mandatory features, finally,
- filter out any excluded modules or excluded mandatory features



4c). Cannot exclude packages

We considered being able to exclude packages:

- This seems like an obvious thing to do, but …
  - It makes resolution far more complex
    - I.e., can no longer just make the choice on the recursively resolved included packages.
  - It makes “implements package foo” harder to understand
  - Note: the same issue occurs if you allow one package version to replace another package version during resolution

Solution:

- Don’t allow packages to be excluded
- Refine definition of “implements a package”
- Tooling will highlight ground truth on package implementation

4d). “**Implementing**” a package

Latest proposal (not documented in draft yet):

- Use structural typing (e.g., like Go or Rust Interfaces)
- I.e., a package is fully implemented if:
  - It has been included by a package
  - All modules are implemented, mandatory features supported, compatible mounts
  - Implementing a later *backwards-compatible** version of the module is also okay
- Tooling could report packages that are not fully implemented:
  - E.g., module removed or older version or non-backwards-compatible new version
- *TODO – Need to consider deviations …*
- *Note – could ”implement” multiple versions of a package …*

5. Considered relationship to device datastore schema + supporting bugfixes

YANG Library schema can be bound to a set of YANG Packages, e.g.,:

- a single base device version package (NOS-25_3_1 package)
- a set of bugfix packages (bgp-25_3_1-2.0.0 package, isis-25_3_1-1.1.0 package), with later module version numbers

These are resolved as if they were in a single top-level package definition that includes them:

- Alternatively, the device could generate this, but must generate a unique device specific name for the package

Relies on automatic resolution between module versions

6. Mount path type - YPath

- Like YANG JSON encoding for instance-identifier, section 6.11 of RFC 7951
- Also like the NACM XML based node-instance-identifier but more JSON like
- YANG Push Lite also looking to use this (or something very similar)

Hence key question is:

- *Should this be specified separately => would likely delay this draft*
- *Or just define what is required in this draft => duplicate similar definitions*