IETF YANG Versioning Packages Update

draft-ietf-netmod-yang-packages-05

NETMOD WG IETF 122, March 2025

Presenting (on behalf of authors): Rob Wilton

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-13)
2. Module semantic version number scheme (draft-ietf-netmod-yang-semver-20)
3. Yang module filename convention (draft-ietf-netmod-yang-module-filename-01)

---

4. **Versioned YANG packages (draft-ietf-netmod-yang-packages-05)**
5. Protocol operations for package version selection (draft-ietf-netmod-yang-ver-selection-00)
6. YANG schema comparison tooling (draft-ietf-netmod-yang-schema-comparison-02)

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

Current Status – YANG Packages

Not spent that much time during last cycle:

- Most time focused on drafts in WGLC
- Picked up YANG packages again about a month back

We have made some minor updates:

- Removed “replaces-version” in a couple of places
- Allow multiple mounted packages

YANG Package Schema

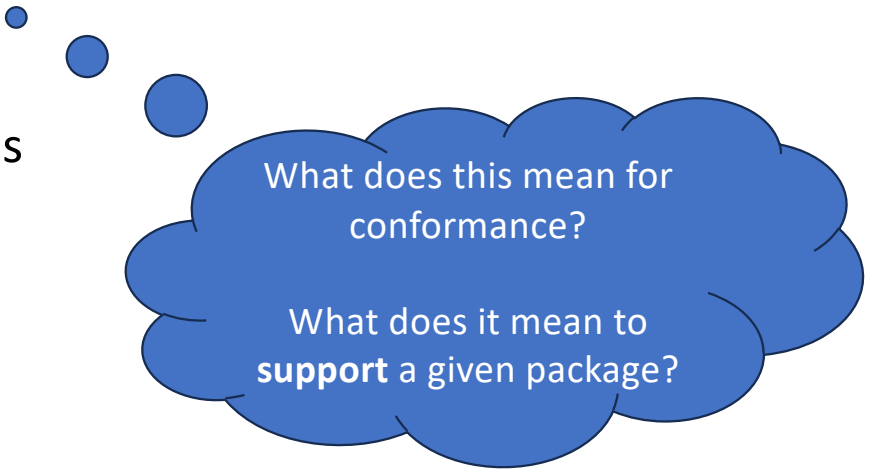
```
grouping yang-pkg-instance:
   +-- name                    pkg-name
   +-- version                 pkg-version
   +-- timestamp?              yang:date-and-time
   +-- organization?           string
   +-- contact?                string
   +-- description?            string
   +-- reference?              string
   +-- complete?               boolean
   +-- tag*                    tags:tag
   +-- supported-feature*      scoped-feature
   +-- included-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
   +-- mounted* [mount-path]
      +-- mount-path?   string
      +-- packages* [name]
         +-- name          pkg-name
         +-- version       pkg-version
         +-- location*     inet:uri
```

Removed “replaces-version”

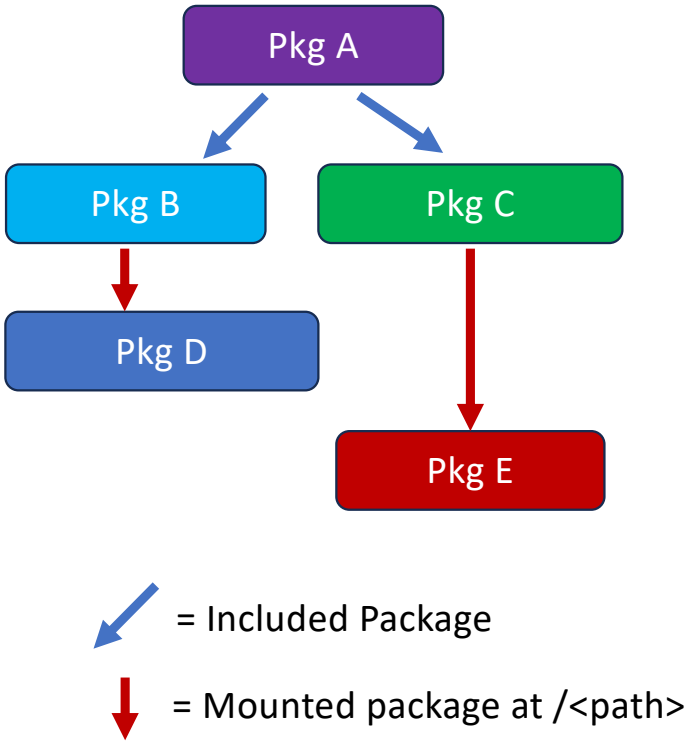
Allow multiple packages (merge not replace)

Current focus is on packages being additive

- E.g., define smaller packages, which are then combined (included) into larger packages
- Is this too restrictive?
  - E.g., if OpenConfig ends up being 1 package, but implementation doesn't support all modules:
  - Could force the use of deviations
  - Or allow a package to remove modules



Mounted Schema

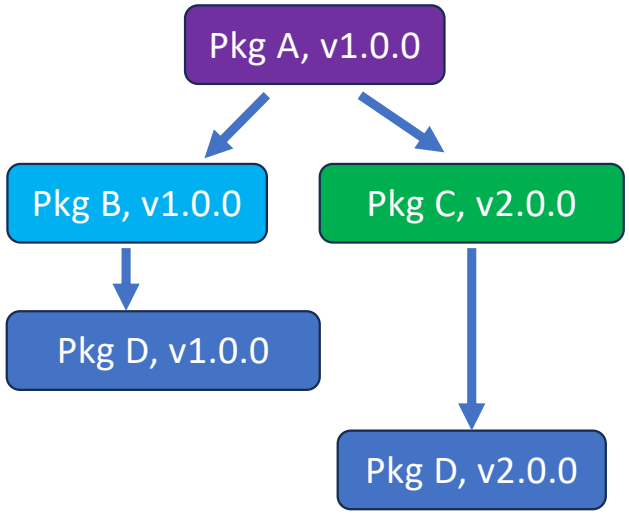


- Latest model allows for multiple packages at a given mount point:
  - E.g., package A includes B and C,
  - B mounts package D at /x,
  - C also mounts package E at /x

  Mounted packages are additive, so package A mounts D and E at /x

Resolving conflicts in included module/package versions

- Draft currently enforces explicit resolution
- Considering whether automatic resolution could work:
  - E.g., choose latest by semver or date
  - Potential complexity from removed modules:
    - E.g., if Pkg D, v 2.0.0 removes a module present in Pkg D, v 1.0.0
    - Problem recurses, e.g., updated package removes an included package.



Plan for next steps

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

Does the WG agree with this direction, any comments?