draft-clacla-netmod-yang-model-update-03

Netmod WG

March 20, 2018

Benoît Claise, Cisco
Joe Clarke, Cisco
Balazs Lengyel, Ericsson
Kevin D'Souza, AT&T

Tracking Module Changes

ietf-interfaces@2013-12-23

```
        }
      ],
      "derived-semantic-version": "1.0.0",
      "implementations": {
        "implementation": [
```

ietf-interfaces@2017-08-17

```
        }
      ],
      "derived-semantic-version": "2.0.0",
      "implementations": {
        "implementation": [
          {
```

- YANG 1.1 stipulates module revisions should not have backward-incompatible changes
  - Module has to be perfect
  - Causes more delay in publishing ratified versions
- This document proposes an alternative mechanism that allow for backward-incompatible changes
- Reflect modules that have such changes using a *semantic version*
  - MAJOR.MINOR.PATCH

Changes Since -02

- Another justification use case around errors in modules that then require backward-incompatible changes to fix
- Clarify when a new module name may be required
  - Splitting a module
  - Removing an overwhelming majority of schema
  - Organizational moves
- Updated YANG module to include augments for ietf-yang-library module and submodule that provide a `module-version` leaf

Fixing “Deprecated” and “Obsolete”

- The definitions of deprecated and obsolete in RFC7950 are problematic
- They provide for a “may or may not” situation whereby the client needs to try a read/write on each node to determine its implementation status
- This does not make for a reliable API contract
- **NEW** Deprecated : Nodes MUST still work as defined. Deprecated serves as a warning that schema nodes will be remove in the future
- **NEW** Obsolete : Schema nodes MUST be removed from the *implementation*. Requests for them must result in errors:
  - `error-tag: operation-failed`
  - `error-app-tag: obsolete element`

Import By Semantic Version

- Import by revision is seldom used and too specific
- With semantic version, import can be more granular
  - Import version X.0.0 or newer of module foo
  - Import version X.*.* of module foo
  - Import version X.Y.Z of module foo
  - ⚌ Operators can better understand module dependencies and whether or not applications may break upon upgrade
- The `module-version` statement SHOULD be a substatement of the `import` statement, and an import statement MUST NOT contain both `revision` and `module-version`
- Import by revision should be discouraged

Open Issues

- Do we need include-by-semver?
- Should IETF/IANA officially generate derived semantic versions for convention for their own modules? As they are the owner of the modules it should be their responsibility, but how to document it?
- There are cases where the module's name should be changed but we still want to formally document the connection between the old and the new module names. For these cases shall we introduce a new YANG extension statement? (e.g., `"replaces-module ietf-vlan;`”)
- Feature and release bundling is currently out-of-scope for this document
- Do we define a new notation (similar to ‘@’) for module file naming (e.g., module%semver.yang)?

Asks

- Who sees this as being important?
- Who wants to work on this?
- Does the WG want to work on and adopt this work?