YANG packages - Versioned YANG schema

draft-rwilton-netmod-yang-packages-02

NETMOD WG

Nov 19, 2019

Presenting: Rob Wilton

IETF

YANG Package - Overview

Defines a versioned YANG schema as the set of YANG module revisions

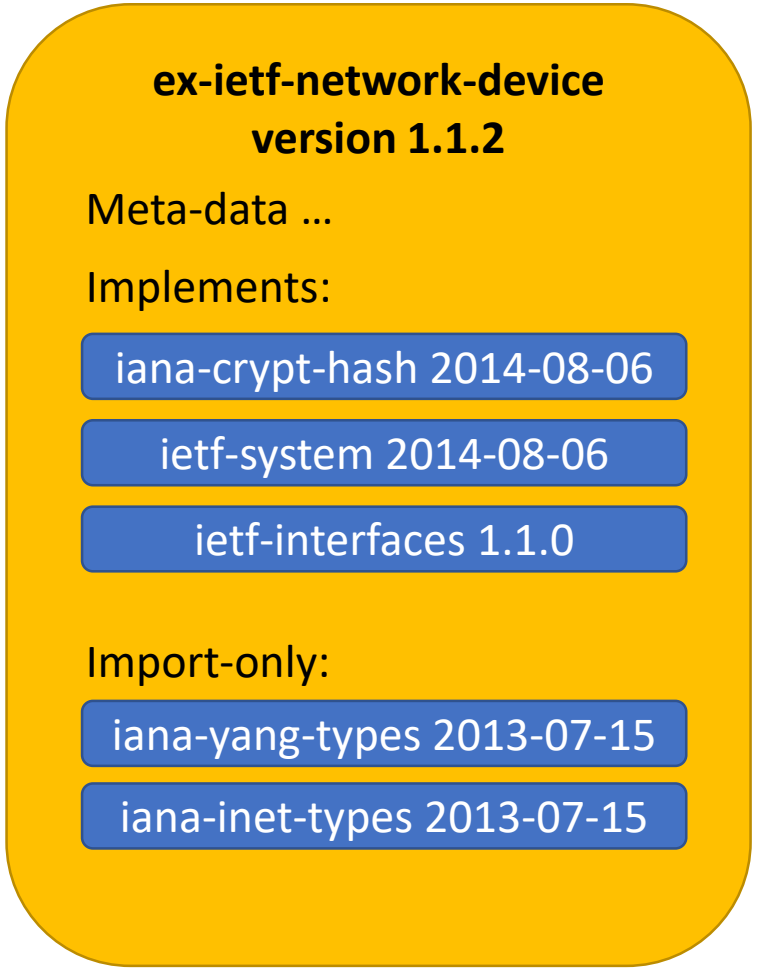
- Versioned using revision-labels - YANG semver can also be used
- Hierarchical - packages can import other packages
- Available offline in a YANG instance data file
  - Clients program against the schema at design time rather than runtime
- Available on the device
- Allows checksums for integrity checks, avoids needing to download complete set of modules from the device.

YANG Package – Why?

Aim to solve several problems:

- Need to version sets of modules instead of just individual modules
- Encourage more consistency in implementations
- Some schema contain 100+ modules, managing these as a flat list is unwieldy
- To avoid downloading/checking the full module list from a device. Making the schema available offline, and then check that the device is using (or just compatible with) the expected schema
- Schema version selection

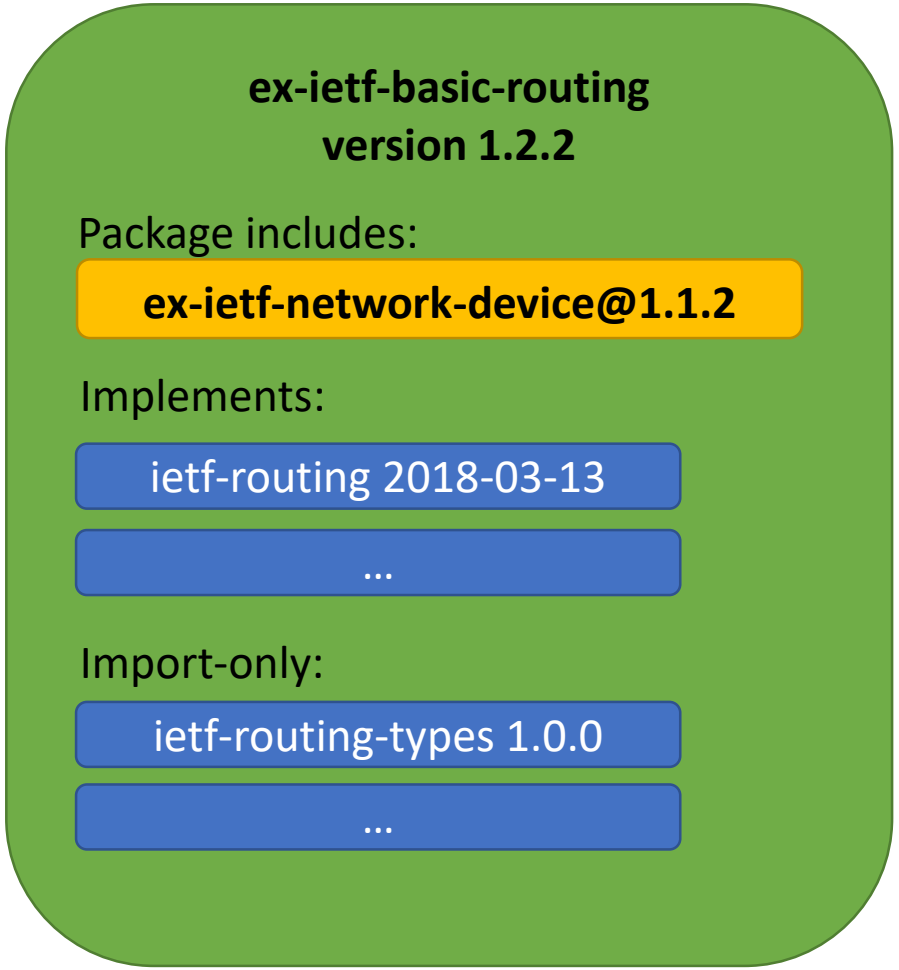
Example Package - **ex-ietf-network-device**



Definition includes:

- Metadata:
  - URLs to find package/module definitions
  - Mandatory features
- Imported packages
- Implemented module version/revisions
- Import-only module versions/revisions
- **Checksums**
- **Import conflict resolution**

Example Package 2 - **example-ietf-basic-routing**



- Ex-ietf-basic-routing imports ex-ex-ietf-network-device and defines more module versions
- **Any version conflict/change must be explicitly resolved**
- Package version indicates nature of changes in the modules or package import

YANG Package – Changes since -01

- Use revision-labels for versioning (packages and modules)
- Add support for checksums*
- Support local scoped packages*
- Conformance improvements*
- Use packages as definition of instance data file schema*
- Lots of minor changes & general draft/model cleanup

(*) More details to follow

Package & Module Checksums

- SHA-256 checksums added to imported module and packages
  - ➢ Allows a client to avoid downloading & comparing package/module
- For modules, checksum is calculated on the .yang file
  - This includes whitespace
- For packages, checksum is calculated on the YANG instance data file
  - Includes whitespace and metadata information

Relationship between packages and schema – **Local packages**

- The aim is for each datastore schema to be defined by one package
- Package definition *should* be available offline (e.g. design time)
- But device schema might be affected by installed optional software components, or affected by hot fixes
- For this scenario, “local packages” can be used:
  - Package name is scoped to the device (rather than globally)
  - Offline definition might not be available

Conformance improvements

- Packages can use ‘revision-labels’ or ‘YANG semver’
- More explicit conformance in places
- Package inclusions define:
  1. Which included package versions they replace (if any)
  2. Whether the included package is nbc modified, i.e. can clients rely on the definition
- Module inclusion define:
  1. Which included module revision they replace (if any) – primarily to allow an implemented module revision to replace an import only module revision

Packages as schema definition for instance data docs

- Packages are intended to be the best way to define a YANG schema
- YANG instance data documents have an associated schema
- Hence allowing YANG packages to be used as the schema definition makes sense
  - Need a clean solution for the bootstrap scenario (i.e. what is the schema for a YANG package)

Main open issues

1. Same or different structure for file vs device
2. Module namespaces
3. Checksum prefixes
4. Use of tags
5. Uber-packages
6. IANA registry for packages

Should packages use different structures for the file vs the device?

- Current approach aims to **optimize for readability** in the file and **minimize data transfer** off the device (i.e. by reusing YANG library module-sets)
- An alternative approach to **use the same structure for both**, with a duplication on module metadata information on the server by not reusing the YANG library module-sets.
  - I.e. the lists of modules comprising a package would be redefined rather than reusing the module-sets from YANG library
  - **Clients shouldn’t generally need module-sets information if using packages**
  - **Currently leaning towards changing this**

Do we need module namespaces in the package definition?

- YANG library **require** module namespace to be specified
- YANG packages **allow** module namespace to be specified
- Probably module name, revision-label, path, and checksum are sufficient
- **But keeping namespace definition allows it to be specified if required/useful**

Require full SHA-256 checksum or allow prefixes?

- The new version of the packages draft uses SHA-256 checksums on module, sub-module, and included package definitions.
- Normally, a SHA-256 checksum is 64 characters long, but we could **allow** a prefix of the checksum to optionally be used in the files instead (i.e. similarly to how git commit hashes are handled).
- E.g. allow a hash reference could perhaps be

  "checksum": " e03f91317f9538a89296e99df3ff0c40”

  Instead of:

  "checksum": "e03f91317f9538a89296e99df3ff0c4003cdfea70bf517407643b3ec13c1ed25“
- **Proposal: Require full 64 char SHA-256**

Use of module tags

- Packages reuse module tags
- The draft doesn’t currently define any mechanism to add, remove, modify the tags associated with a package on a device.
- Should this be added, or can this work reasonably be deferred?
- **Proposal is to defer this additional work at this time**

Packages for uber-schema

- Each package represents a [potentially incomplete] YANG schema
- For NMDA compliant devices, RFC 8342 implies the existence of an uber-schema that represents the common parent schema across all datastores
- Similar uber-schema can exist for particular schema families (e.g. IETF, OpenConfig, Native)
- It may be useful to refer to these uber-schema using packages when advertising the schema of a device, or during schema version selection
- **Proposal: Still in open DT discussion**

IANA registry for packages

- An IANA registry (or similar) for YANG package definitions is useful
- But is IANA the right choice:
  - Should these just cover IETF standardized YANG packages, or all YANG packages?
  - Does IANA manage the package revisions?
  - Or should we try and just use something like Github with some expert review/release process?
- **Proposal: Not sure**