Network Working Group R. Wilton, Ed.
Internet-Draft R. Rahman
Intended status: Standards Track J. Clarke
Expires: May 6, 2021 Cisco Systems, Inc.
J. Sterne
Nokia
B. Wu, Ed.
Huawei
November 2, 2020

YANG Packages
draft-ietf-netmod-yang-packages-01

Abstract

This document defines YANG packages, a versioned organizational structure holding a set of related YANG modules that collectively define a YANG schema. It describes how packages: are represented on a server, can be defined in offline YANG instance data files, and can be used to define the schema associated with YANG instance data files.

Status of This Memo

Copyright Notice

Table of Contents

1. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terminology introduced in the YANG versioning requirements draft [I-D.ietf-netmod-yang-versioning-reqs].

This document also makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

- datastore schema

This document also makes of the following terminology introduced in the YANG 1.1 Data Modeling Language [RFC7950]:

- data node

In addition, this document defines the following terminology:

- YANG schema: A datastore schema, not bound to any particular datastore.

- YANG package: An organizational structure containing a collection of YANG modules, normally defined in a YANG instance data file. A YANG package defines a YANG schema by specifying a set of YANG modules and their revisions, other packages and their revisions, mandatory features, and deviations. YANG packages are defined in Section 5.

- backwards-compatible (BC) change: When used in the context of a YANG module, it follows the definition in Section 3.1.1 of [I-D.ietf-netmod-yang-module-versioning]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.2.

- non-backwards-compatible (NBC) change: When used in the context of a YANG module, it follows the definition in Section 3.1.2 of [I-D.ietf-netmod-yang-module-versioning]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.2.

- editorial change: When used in the context of a YANG module, it follows the definition of an 'editorial change' in 3.2 of [I-D.ietf-netmod-yang-module-versioning]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.3.

2. Introduction

This document defines and describes the YANG [RFC7950] constructs that are used to define and use YANG packages.

A YANG package is an organizational structure that groups a set of YANG modules together into a consistent versioned definition. For example, a YANG package could define the set of YANG modules required to implement an L2VPN service on a network device. YANG packages can themselves refer to, and reuse, other package definitions.

Non-normative examples of YANG packages are provided in the appendices.

3. Background on YANG packages

It has long been acknowledged within the YANG community that network management using YANG requires a unit of organization and conformance that is broader in scope than individual YANG modules.

'The YANG Package Statement' [I-D.bierman-netmod-yang-package] proposed a YANG package mechanism based on new YANG language statements, where a YANG package is defined in a file similar to how YANG modules are defined, and would require enhancements to YANG compilers to understand the new statements used to define packages.

OpenConfig [openconfigsemver] describes an approach to versioning 'bundle releases' based on git tags. I.e. a set of modules, at particular versions, can be marked with the same release tag to indicate that they are known to interoperate together.

The NETMOD WG in general, and the YANG versioning design team in particular, are exploring solutions [I-D.ietf-netmod-yang-solutions] to the YANG versioning requirements, [I-D.ietf-netmod-yang-versioning-reqs]. Solutions to the versioning requirements can be split into several distinct areas. [I-D.ietf-netmod-yang-module-versioning] is focused on YANG versioning scoped to individual modules. The overall solution must also consider YANG versioning and conformance scoped to YANG schema. YANG packages provide part of the solution for versioning YANG schema.

4. Objectives

The main goals of YANG package definitions include, but are not restricted to:

o To provide an alternative, simplified, YANG conformance mechanism. Rather than conformance being performed against a set of individual YANG module revisions, features, and deviations, conformance can be more simply stated in terms of YANG packages, with a set of modifications (e.g. additional modules, deviations, or features).

o To allow YANG schema to be specified in a concise way rather than having each server explicitly list all modules, revisions, and features. YANG package definitions can be defined in documents that are available offline, and accessible via a URL, rather than requiring explicit lists of modules to be shared between client and server. Hence, a YANG package must contain sufficient information to allow a client or server to precisely construct the schema associated with the package.

o To define a mainly linear versioned history of sets of modules versions that are known to work together. I.e. to help mitigate the problem where a client must manage devices from multiple vendors, and vendor A implements version 1.0.0 of module foo and version 2.0.0 of module bar, and vendor B implements version 2.0.0 of module foo and version 1.0.0 of module bar. For a client, trying to interoperate with multiple vendors, and many YANG modules, finding a consistent lowest common denominator set of YANG module versions may be difficult, if not impossible.

Protocol mechanisms of how clients can negotiate which packages or package versions are to be used for NETCONF/RESTCONF communications are outside the scope of this document, and are defined in [I-D.ietf-netmod-yang-ver-selection].

Finally, the package definitions proposed by this document are intended to be relatively basic in their definition and the functionality that they support. As industry gains experience using

YANG packages, the standard YANG mechanisms of updating, or augmenting YANG modules could also be used to extend the functionality supported by YANG packages, if required.

5. YANG Package Definition

This document specifies an approach to defining YANG packages that is different to either of the approaches described in the background.

A YANG package is a versioned organizational structure defining a set of related YANG modules, packages, features, and deviations. A YANG package collectively defines a YANG schema.

Each YANG package has a name that SHOULD end with the suffix "-pkg". Package names are normally expected to be globally unique, but in some cases the package name may be locally scoped to a server or device, as described in Section 5.5.

YANG packages are versioned using the same approaches described in [I-D.ietf-netmod-yang-module-versioning] and [I-D.ietf-netmod-yang-semver]. This is described in further detail in Section 5.2.

Each YANG package version, defines:

- o some metadata about the package, e.g., description, tags, scoping, referential completeness, location information.
- o a set of YANG modules, at particular revisions, that are implemented by servers that implement the package. The modules may contain deviations.
- o a set of import-only YANG modules, at particular revisions, that are used 'import-only' by the servers that implement the package.
- o a set of included YANG packages, at particular revisions, that are also implemented by servers that implement the package.
- o a set of YANG module features that must be supported by servers that implement the package.

The structure for YANG package definitions uses existing YANG language statements, YANG Data Structure Extensions [I-D.ietf-netmod-yang-data-ext], and YANG Instance Data File Format [I-D.ietf-netmod-yang-instance-file-format].

YANG package definitions are available offline in YANG instance data files. Client applications can be designed to support particular package versions that they expect to interoperate with.

YANG package definitions are available from the server via augmentations to YANG Library [RFC8525]. Rather than client applications downloading the entire contents of YANG library to confirm that the server schema is compatible with the client, they can check, or download, a much shorter YANG package definition, and validate that it conforms to the expected schema.

YANG package definitions can also be used to define the schema associated with YANG instance data files holding other, e.g., non packages related, instance data.

5.1. Package definition rules

Packages are defined using the following rules:

1. A YANG package MAY represent a complete YANG schema or only part of a YANG schema with some module import dependencies missing, as described in Section 5.4.

2. Packages definitions are hierarchical. A package can include other packages. Only a single version of a package can be included, and conflicting package includes (e.g. from descendant package includes) MUST be explicitly resolved by indicating which version takes precedence, and which versions are being replaced.

3. For each module implemented by a package, only a single revision of that module MUST be implemented. Multiple revisions of a module MAY be listed as import-only dependencies.

4. The revision of a module listed in the package 'module' list supersedes any 'implemented' revision of the module listed in an included package module list. The 'replaces-revision' leaf-list is used to indicate which 'implemented' or 'import-only' module revisions are replaces by this module revision. This allows a package to explicitly resolve conflicts between implemented module revisions in included packages.

5. The 'replaces-revision' leaf-list in the 'import-only-module' list can be used to exclude duplicate revisions of import-only modules from included packages. Otherwise, the import-only-modules for a package are the import-only-modules from all included packages combined with any modules listed in the packages import-only-module list.

6. YANG packages definitions MAY include modules containing deviation statements, but those deviation statements MUST only be used in an RFC 7950 compatible way to indicate where a server, or class of servers, deviates from a published standard. Deviations MUST NOT be included in a package definition that is part of a published standard. See section 5.8.1 for further guidance on the use of deviations in YANG packages.

5.2. Package versioning

Individual versions of a YANG package are versioned using the "revision-label" scheme defined in section 3.3 of [I-D.ietf-netmod-yang-module-versioning].

5.2.1. Updating a package with a new version

Package compatibility is fundamentally defined by how the YANG schema between two package versions has changed.

When a package definition is updated, the version associated with the package MUST be updated appropriately, taking into consideration the scope of the changes as defined by the rules below.

A package definition SHOULD define the previous version of the package in the 'previous-version' leaf unless it is the initial version of the package. If the 'previous-version' leaf is provided then the package definition MUST set the 'nbc-changes' leaf if the new version is non-backwards-compatible with respect to the package version defined in the 'previous-version' leaf.

5.2.1.1. Non-Backwards-compatible changes

The following changes classify as non-backwards-compatible changes to a package definition:

o Changing an 'included-package' list entry to select a package version that is non-backwards-compatible to the prior package version, or removing a previously included package.

o Changing a 'module' or 'import-only-module' list entry to select a module revision that is non-backwards-compatible to the prior module revision, or removing a previously implemented module.

o Removing a feature from the 'mandatory-feature' leaf-list.

o Adding, changing, or removing a deviation that is considered a non-backwards-compatible change to the affected data node in the schema associated with the prior package version.

5.2.1.2. Backwards-compatible changes

The following changes classify as backwards-compatible changes to a package definition:

- o Changing an 'included-package' list entry to select a package version that is backwards-compatible to the prior package version, or including a new package that does not conflict with any existing included package or module.

- o Changing a 'module' or 'import-only-module' list entry to select a module revision that is backwards-compatible to the prior module revision, or including a new module to the package definition.

- o Adding a feature to the 'mandatory-feature' leaf-list.

- o Adding, changing, or removing a deviation that is considered a backwards-compatible change to the affected data node in the schema associated with the prior package version.

5.2.1.3. Editorial changes

The following changes classify as editorial changes to a package definition:

- o Changing a 'included-package' list entry to select a package version that is classified as an editorial change relative to the prior package version.

- o Changing a 'module' or 'import-only-module' list entry to select a module revision that is classified as an editorial change relative to the prior module revision.

- o Any change to any metadata associated with a package definition that causes it to have a different checksum value.

5.2.2. YANG Semantic Versioning for packages

YANG Semantic Versioning [I-D.ietf-netmod-yang-semver] MAY be used as an appropriate type of revision-label for the package version leaf.

If the format of the leaf matches the 'yangver:version' type specified in ietf-yang-semver.yang, then the package version leaf MUST be interpreted as a YANG semantic version number.

For YANG packages defined by the IETF, YANG semantic version numbers MUST be used as the version scheme for YANG packages.

The rules for incrementing the YANG package version number are equivalent to the semantic versioning rules used to version individual YANG modules, defined in section 3.2 of [I-D.ietf-netmod-yang-semver], but use the rules defined previously in Section 5.2.1 to determine whether a change is classified as non-backwards-compatible, backwards-compatible, or editorial. Where available, the semantic version number of the referenced elements in the package (included packages or modules) can be used to help determine the scope of changes being made.

5.2.3. Revision history

YANG packages do not contain a revision history. This is because packages may have many revisions and a long revision history would bloat the package definition. By recursively examining the 'previous-version' leaf of a package definition, a full revision history (including where non-backwards-compatible changes have occurred) can be dynamically constructed, if all package versions are available.

5.3. Package conformance

YANG packages allows for conformance to be checked at a package level rather than requiring a client to download all modules, revisions, and deviations from the server to ensure that the datastore schema used by the server is compatible with the client.

YANG package conformance is analogous to how YANG [RFC7950] requires that servers either implement a module faithfully, or otherwise use deviations to indicate areas of non-conformance.

For a top level package representing a datastore schema, servers MUST implement the package definition faithfully, including all mandatory features.

Package definitions MAY modify the schema for directly or hierarchically included packages through the use of different module revisions or module deviations. If the schema of any included package is modified in a non-backwards-compatible way then it MUST be indicated by setting the 'nbc-modified' leaf to true.

5.3.1. Use of YANG semantic versioning

Using the YANG semantic versioning scheme for package version numbers and module revision labels can help with conformance. In the general case, clients should be able to determine the nature of changes between two package versions by comparing the version number.

This usually means that a client does not have to be restricted to working only with servers that advertise exactly the same version of a package in YANG library. Instead, reasonable clients should be able to interoperate with any server that supports a package version that is backwards compatible to version that the client is designed for, assuming that the client is designed to ignore operational values for unknown data nodes.

For example, a client coded to support 'foo' package at version 1.0.0 should interoperate with a server implementing 'foo' package at version 1.3.5, because the YANG semantic versioning rules require that package version 1.3.5 is backwards compatible to version 1.0.0.

This also has a relevance on servers that are capable of supporting version selection because they need not support every version of a YANG package to ensure good client compatibility. Choosing suitable minor versions within each major version number should generally be sufficient, particular if they can avoid non-backwards-compatible patch level changes.

5.3.2. Package checksums

Each YANG package definition may have a checksum associated with it to allow a client to validate that the package definition of the server matches the expected package definition without downloading the full package definition from the server.

The checksum for a package is calculated using the SHA-256 hash (XXX, reference) of the full file contents of the YANG package instance data file. This means that the checksum includes all whitespace and formatting, encoding, and all meta-data fields associated with the package and the instance data file).

The checksum for a module is calculated using the SHA-256 hash of the YANG module file definition. This means that the checksum includes all whitespace, formatting, and comments within the YANG module.

Packages that are locally scoped to a server may not have an offline instance data file available, and hence MAY not have a checksum.

The package definition allows URLs and checksums to be specified for all included packages, modules and submodules within the package definition. Checksums SHOULD be included in package definitions to validate the full integrity of the package.

On a server, package checksums SHOULD also be provided for the top level packages associated with the datastore schema.

5.3.3. The relationship between packages and datastores

As defined by NMDA [RFC8342], each datastore has an associated datastore schema. Sections 5.1 and 5.3 of NMDA defines further constraints on the schema associated with datastores. These constraints can be summarized thus:

- o The schema for all conventional datastores is the same.
- o The schema for non conventional configuration datastores (e.g., dynamic datastores) may completely differ (i.e. no overlap at all) from the schema associated with the conventional configuration datastores, or may partially or fully overlap with the schema of the conventional configuration datastores. A dynamic datastore, for example, may support different modules than conventional datastores, or may support a subset or superset of modules, features, or data nodes supported in the conventional configuration datastores. Where a data node exists in multiple datastore schema it has the same type, properties and semantics.
- o The schema for the operational datastore is intended to be a superset of all the configuration datastores (i.e. includes all the schema nodes from the conventional configuration datastores), but data nodes can be omitted if they cannot be accurately reported. The operational datastore schema can include additional modules containing only config false data nodes, but there is no harm in including those modules in the configuration datastore schema as well.

Given that YANG packages represent a YANG schema, it follows that each datastore schema can be represented using packages. In addition, the schema for most datastores on a server are often closely related. Given that there are many ways that a datastore schema could be represented using packages, the following guidance provides a consistent approach to help clients understand the relationship between the different datastore schema supported by a device (e.g., which parts of the schema are common and which parts have differences):

- o Any datastores (e.g., conventional configuration datastores) that have exactly the same datastore schema MUST use the same package definitions. This is to avoid, for example, the creation of a 'running-cfg' package and a separate 'intended-cfg' package that have identical schema.
- o Common package definitions SHOULD be used for those parts of the datastore schema that are common between datastores, when those datastores do not share exactly the same datastore schema. E.g.,

if a substantial part of the schema is common between the conventional, dynamic, and operational datastores then a single common package can be used to describe the common parts, along with other packages to describe the unique parts of each datastore schema.

- o YANG modules that do not contain any configuration data nodes SHOULD be included in the package for configuration datastores if that helps unify the package definitions.

- o The packages for the operational datastore schema MUST include all packages for all configuration datastores, along with any required modules defining deviations to mark unsupported data nodes. The deviations MAY be defined directly in the packages defining the operational datastore schema, or in separate non referentially complete packages.

- o The schema for a datastore MAY be represented using a single package or as the union of a set of compatible packages, i.e., equivalently to a set of non-conflicting packages being included together in an overarching package definition.

5.4. Schema referential completeness

A YANG package may represent a schema that is 'referentially complete', or 'referentially incomplete', indicated in the package definition by the 'complete' flag.

If all import statements in all YANG modules included in the package (either directly, or through included packages) can be resolved to a module revision defined with the YANG package definition, then the package is classified as referentially complete. Conversely, if one or more import statements cannot be resolved to a module specified as part of the package definition, then the package is classified as referentially incomplete.

A package that represents the exact contents of a datastore schema MUST always be referentially complete.

Referentially incomplete packages can be used, along with locally scoped packages, to represent an update to a device's datastore schema as part of an optional software hot fix. E.g., the base software is made available as a complete globally scoped package. The hot fix is made available as an incomplete globally scoped package. A device's datastore schema can define a local package that implements the base software package updated with the hot fix package.

Referentially incomplete packages could also be used to group sets of logically related modules together, but without requiring a fixed dependency on all imported 'types' modules (e.g., iana-if-types.yang), instead leaving the choice of specific revisions of 'types' modules to be resolved when the package definition is used.

5.5. Package name scoping and uniqueness

YANG package names can be globally unique, or locally scoped to a particular server or device.

5.5.1. Globally scoped packages

The name given to a package MUST be globally unique, and it MUST include an appropriate organization prefix in the name, equivalent to YANG module naming conventions.

Ideally a YANG instance data file defining a particular package version would be publicly available at one or more URLs.

5.5.2. Server scoped packages

Package definitions may be scoped to a particular server by setting the 'is-local' leaf to true in the package definition.

Locally scoped packages MAY have a package name that is not globally unique.

Locally scoped packages MAY have a definition that is not available offline from the server in a YANG instance data file.

5.6. Submodules packages considerations

As defined in [RFC7950] and [I-D.ietf-netmod-yang-semver], YANG conformance and versioning is specified in terms of particular revisions of YANG modules rather than for individual submodules.

However, YANG package definitions also include the list of submodules included by a module, primarily to provide a location of where the submodule definition can be obtained from, allowing a YANG schema to be fully constructed from a YANG package instance data file definition.

5.7. Package tags

[I-D.ietf-netmod-module-tags] defines YANG module tags as a mechanism to annotate a module definition with additional metadata. Tags MAY also be associated to a package definition via the 'tags' leaf-list.

The tags use the same registry and definitions used by YANG module tags.

5.8. YANG Package Usage Guidance

It is RECOMMENDED that organizations that publish YANG modules also publish YANG package definition that group and version those modules into units of related functionality. This increases interoperability, by encouraging implementations to use the same collections of YANG modules versions. Using packages also makes it easier to understand relationship between modules, and enables functionality to be described on a more abstract level than individual modules.

5.8.1. Use of deviations in YANG packages

[RFC7950] section 5.6.3 defines deviations as the mechanism to allow servers to indicate where they do not conform to a published YANG module that is being implemented.

In cases where implementations contain deviations from published packages, then those implementations SHOULD define a package that includes both the published packages and all modules containing deviations. This implementation specific package accurately reflects the schema used by the device and allows clients to determine how the implementation differs from the published package schema in an offline consumable way, e.g., when published in an instance data file (see section 6).

Organizations may wish to reuse YANG modules and YANG packages published by other organizations for new functionality. Sometimes, they may desire to modify the published YANG modules. However, they MUST NOT use deviations in an attempt to achieve this because such deviations cause two problems:

They prevent implementations from reporting their own deviations for the same nodes.

They fracture the ecosystem by preventing implementations from conforming to the standards specified by both organizations. This hurts the interoperability in the YANG community, promotes development of disconnected functional silos, and hurts creativity in the market.

5.8.2. Use of features in YANG modules and YANG packages

The YANG language supports feature statements as the mechanism to make parts of a schema optional. Published standard YANG modules SHOULD make use of appropriate feature statements to provide flexibility in how YANG modules may be used by implementations and used by YANG modules published by other organizations.

YANG packages support 'mandatory features' which allow a package to specify features that MUST be implemented by any conformant implementation of the package as a mechanism to simplify and manage the schema represented by a YANG package.

5.9. YANG package core definition

The ietf-yang-package-types.yang module defines a grouping to specify the core elements of the YANG package structure that is used within YANG package instance data files (ietf-yang-package-instance.yang) and also on the server (ietf-yang-packages.yang).

The "ietf-yang-package-types" YANG module has the following structure:

```
module: ietf-yang-package-types

  grouping yang-pkg-identification-leafs
    +-- name       pkg-name
    +-- version    pkg-version

  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
    +-- local?                 boolean
    +-- previous-version?      pkg-version
    +-- nbc-changes?           boolean
    +-- tag*                   tags:tag
    +-- mandatory-feature*     scoped-feature
    +-- included-package* [name version]
    |  +-- name                 pkg-name
    |  +-- version              pkg-version
    |  +-- replaces-version*    pkg-version
```

```
       |  +-- nbc-modified?       boolean
       |  +-- location*           inet:uri
       |  +-- checksum?           pkg-types:sha-256-hash
       +-- module* [name]
       |  +-- name                 yang:yang-identifier
       |  +-- revision?            rev:revision-date-or-label
       |  +-- replaces-revision*   rev:revision-date-or-label
       |  +-- namespace?           inet:uri
       |  +-- location*            inet:uri
       |  +-- checksum?            pkg-types:sha-256-hash
       |  +-- submodule* [name]
       |     +-- name?       yang:yang-identifier
       |     +-- revision    yang:revision-identifier
       |     +-- location*   inet:uri
       |     +-- checksum?   pkg-types:sha-256-hash
       +-- import-only-module* [name revision]
          +-- name?                yang:yang-identifier
          +-- revision?            rev:revision-date-or-label
          +-- replaces-revision*   rev:revision-date-or-label
          +-- namespace?           inet:uri
          +-- location*            inet:uri
          +-- checksum?            pkg-types:sha-256-hash
          +-- submodule* [name]
             +-- name?       yang:yang-identifier
             +-- revision    yang:revision-identifier
             +-- location*   inet:uri
             +-- checksum?   pkg-types:sha-256-hash
```

6. Package Instance Data Files

YANG packages SHOULD be available offline from the server, defined as YANG instance data files [I-D.ietf-netmod-yang-instance-file-format] using the YANG schema below to define the package data.

The following rules apply to the format of the YANG package instance files:

1. The file SHOULD be encoded in JSON.

2. The name of the file SHOULD follow the format "<package-name>@<version>.json".

3. The package name MUST be specified in both the instance-data-set 'name' and package 'name' leafs.

4. The 'description' field of the instance-data-set SHOULD be "YANG package definition".

5. The ’timestamp’, "organization’, ’contact’ fields are defined in both the instance-data-set metadata and the YANG package metadata. Package definitions SHOULD only define these fields as part of the package definition. If any of these fields are populated in the instance-data-set metadata then they MUST contain the same value as the corresponding leaves in the package definition.

6. The ’revision’ list in the instance data file SHOULD NOT be used, since versioning is handled by the package definition.

7. The instance data file for each version of a YANG package SHOULD be made available at one of more locations accessible via URLs. If one of the listed locations defines a definitive reference implementation for the package definition then it MUST be listed as the first entry in the list.

The "ietf-yang-package" YANG module has the following structure:

```
module: ietf-yang-package

  structure package:
    // Uses the yang-package-instance grouping defined in
    // ietf-yang-package-types.yang
    +-- name                  pkg-name
    +-- version               pkg-version
    ... remainder of yang-package-instance grouping ...
```

7. Package Definitions on a Server

7.1. Package List

A top level ’packages’ container holds the list of all versions of all packages known to the server. Each list entry uses the common package definition, but with the addition of package location and checksum information that cannot be contained within a offline package definition contained in an instance data file.

The ’/packages/package’ list MAY include multiple versions of a particular package. E.g. if the server is capable of allowing clients to select which package versions should be used by the server.

7.2. Tree diagram

The "ietf-yang-packages" YANG module has the following structure:

```
module: ietf-yang-packages
  +--ro packages
     +--ro package* [name version]
        // Uses the yang-package-instance grouping defined in
        // ietf-yang-package-types.yang, with location and checksum:
        +--ro name                   pkg-name
        +--ro version                pkg-version
        ... remainder of yang-package-instance grouping ...
        +--ro location*              inet:uri
        +--ro checksum?              pkg-types:sha-256-hash
```

8. YANG Library Package Bindings

The YANG packages module also augments YANG library to allow a server to optionally indicate that a datastore schema is defined by a package, or a union of compatible packages. Since packages can generally be made available offline in instance data files, it may be sufficient for a client to only check that a compatible version of the package is implemented by the server without fetching either the package definition, or downloading and comparing the full list of modules and enabled features.

If a server indicates that a datastore schema maps to a particular package, then it MUST exactly match the schema defined by that package, taking into account enabled features and any deviations.

If a server cannot faithfully implement a package then it can define a new package to accurately report what it does implement. The new package can include the original package as an included package, and the new package can define additional modules containing deviations to the modules in the original package, allowing the new package to accurately describe the server's behavior. There is no specific mechanism provided to indicate that a mandatory-feature in package definition is not supported on a server, but deviations MAY be used to disable functionality predicated by an if-feature statement.

The "ietf-yl-packages" YANG module has the following structure:

```
module: ietf-yl-packages
  augment /yanglib:yang-library/yanglib:schema:
    +--ro package* [name version]
       +--ro name        -> /pkgs:packages/package/name
       +--ro version     leafref
       +--ro checksum?   leafref
```

9. YANG packages as schema for YANG instance data document

YANG package definitions can be used as the schema definition for YANG instance data files. When using a package schema, the name and version of the package MUST be specified, a package checksum and/or URL to the package definition MAY also be provided.

The "ietf-yang-inst-data-pkg" YANG module has the following structure:

```
module: ietf-yang-inst-data-pkg

  augment-structure /yid:instance-data-set/yid:content-schema-spec:
    +--:(pkg-schema)
      +-- pkg-schema
        +-- name        pkg-name
        +-- version     pkg-version
        +-- location*   inet:uri
        +-- checksum?   pkg-types:sha-256-hash
```

10. YANG Modules

The YANG module definitions for the modules described in the previous sections.

```
<CODE BEGINS> file "ietf-yang-package-types@2020-01-21.yang"
module ietf-yang-package-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package-types";
  prefix "pkg-types";

  import ietf-yang-revisions {
    prefix rev;
    reference "XXXX: Updated YANG Module Revision Handling";
```

```
     }

     import ietf-yang-types {
       prefix yang;
       rev:revision-or-derived 2019-07-21;
       reference "RFC 6991bis: Common YANG Data Types.";
     }

     import ietf-inet-types {
       prefix inet;
       rev:revision-or-derived 2013-07-15;
       reference "RFC 6991: Common YANG Data Types.";
     }

     import ietf-module-tags {
       prefix tags;
       // RFC Ed. Fix revision once revision date of
       // ietf-module-tags.yang is known.
       reference "RFC XXX: YANG Module Tags.";
     }

     organization
       "IETF NETMOD (Network Modeling) Working Group";

     contact
       "WG Web:   <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>

        Author:   Rob Wilton
                  <mailto:rwilton@cisco.com>";

     description
       "This module provides type and grouping definitions for YANG
        packages.

        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.
```

```
           The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
           NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
           'MAY', and 'OPTIONAL' in this document are to be interpreted as
           described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
           they appear in all capitals, as shown here.";

        // RFC Ed.: update the date below with the date of RFC publication
        // and remove this note.
        // RFC Ed.: replace XXXX with actual RFC number and remove this
        // note.
        revision 2020-01-21 {
          rev:revision-label 0.2.0;
          description
            "Initial revision";
          reference
            "RFC XXXX: YANG Packages";
        }


        /*
         * Typedefs
         */

        typedef pkg-name {
          type yang:yang-identifier;
          description
            "Package names are typed as YANG identifiers.";
        }

        typedef pkg-version {
          type rev:revision-date-or-label;
          description
            "Package versions SHOULD be a revision-label (e.g. perhaps a
             YANG Semver version string).  Package versions MAY also be a
             revision-date";

        }

        typedef pkg-identifier {
          type rev:name-revision;
          description
            "Package identifiers combine a pkg-name and a pkg-version";
        }

        typedef scoped-feature {
          type string {
            pattern '[a-zA-Z_][a-zA-Z0-9\-_.]*:[a-zA-Z_][a-zA-Z0-9\-_.]*';
          }
```

```
       description
         "Represents a feature name scoped to a particular module,
          identified as the '<module-name>:<feature-name>', where both
          <module-name> and <feature-name> are YANG identifier strings,
          as defiend by Section 12 or RFC 6020.";
       reference
         "RFC XXXX, YANG Packages.";
     }

     typedef sha-256-hash {
       type string {
         length "64";
         pattern "[0-9a-fA-F]*";
       }
       description
         "A SHA-256 hash represented as a hexadecimal string.

          Used as the checksum for modules, submodules and packages in a
          YANG package definition.

          For modules and submodules the SHA-256 hash is calculated on
          the contents of the YANG file defining the module/submodule.

          For packages the SHA-256 hash is calculated on the file
          containing the YANG instance data document holding the package
          definition";
     }


     /*
      * Groupings
      */
     grouping yang-pkg-identification-leafs {
       description
         "Parameters for identifying a specific version of a YANG
          package";

       leaf name {
         type pkg-name;
         mandatory true;
         description
           "The YANG package name.";
       }

       leaf version {
         type pkg-version;
         mandatory true;
         description
```

```
           "Uniquely identies a particular version of a YANG package.

            Follows the definition for revision labels defined in
            draft-verdt-nemod-yang-module-versioning, section XXX";
       }
     }

     grouping yang-pkg-instance {
       description
         "Specifies the data node for a full YANG package instance
          represented either on a server or as a YANG instance data
          document.";
       uses yang-pkg-identification-leafs;

       leaf timestamp {
         type yang:date-and-time;

         description
           "An optional timestamp for when this package was created.
            This does not need to be unique across all versions of a
            package.";
       }

       leaf organization {
         type string;

         description "Organization responsible for this package";
       }

       leaf contact {
         type string;

         description
           "Contact information for the person or organization to whom
            queries concerning this package should be sent.";
       }

       leaf description {
         type string;

         description "Provides a description of the package";
       }

       leaf reference {
         type string;

         description "Allows for a reference for the package";
       }
       leaf complete {
         type boolean;
         default true;
         description
           "Indicates whether the schema defined by this package is
            referentially complete.  I.e. all module imports can be
            resolved to a module explicitly defined in this package or
            one of the included packages.";
       }

       leaf local {
         type boolean;
         default false;
         description
           "Defines that the package definition is local to the server,
            and the name of the package MAY not be unique, and the
            package definition MAY not be available in an offline file.

            Local packages can be used when the schema for the device
            can be changed at runtime through the addition or removal of
            software packages, or hot fixes.";
       }

       leaf previous-version {
         type pkg-version;
         description
           "The previous package version that this version has been
            derived from.  This leaf allows a full version history graph
            to be constructed if required.";
       }

       leaf nbc-changes {
         type boolean;
         default false;
         description
           "Indicates whether the defined package version contains
            non-backwards-compatible changes relative to the package
            version defined in the 'previous-version' leaf.";
       }

       leaf-list tag {
         type tags:tag;
         description
           "Tags associated with a YANG package.  Module tags defined in
            XXX, ietf-netmod-module-tags can be used here but with the
            modification that the tag applies to the entire package
            rather than a specific module.  See the IANA 'YANG Module
            Tag Prefix' registry for reserved prefixes and the IANA
```

```
             'YANG Module IETF Tag' registry for IETF standard tags.";
        }

        leaf-list mandatory-feature {
          type scoped-feature;
          description
            "Lists features from any modules included in the package that
             MUST be supported by any server implementing the package.

             Features already specified in a 'mandatory-feature' list of
             any included package MUST also be supported by server
             implementations and do not need to be repeated in this list.

             All other features defined in modules included in the
             package are OPTIONAL to implement.

             Features are identified using <module-name>:<feature-name>";
        }

        list included-package {
          key "name version";
          description
            "An entry in this list represents a package that is included
             as part of the package definition, or an indirectly included
             package that is changed in a non backwards compatible way.

             It can be used to resolve inclusion of conflicting package
             versions by explicitly specifying which package version is
             used.

             If included packages implement different revisions or
             versions of the same module, then an explicit entry in the
             module list MUST be provided to select the specific module
             version 'implemented' by this package definition.

             If the schema for any packages that are included, either
             directly or indirectly via another package include, are
             changed in any non-backwards-compatible way then they MUST
             be explicitly listed in the included-packages list with the
             'nbc-modified' leaf set to true.

             For import-only modules, the 'replaces-revision' leaf-list
             can be used to select the specific module versions used by
             this package.";
          reference
            "XXX";

          uses yang-pkg-identification-leafs;
```

```
         leaf-list replaces-version {
           type pkg-version;
           description
             "Gives the version of an included package version that
              is replaced by this included package revision.";
         }

         leaf nbc-modified {
           type boolean;
           default false;
           description
             "Set to true if any data nodes in this package are modified
              in a non backwards compatible way, either through the use
              of deviations, or because one of the modules has been
              replaced by an incompatible revision.  This could also
              occur if a module's revision was replaced by an earlier
              revision that had the effect of removing some data
              nodes.";
         }

         leaf-list location {
           type inet:uri;
           description
             "Contains a URL that represents where an instance data file
              for this YANG package can be found.

              This leaf will only be present if there is a URL available
              for retrieval of the schema for this entry.

              If multiple locations are provided, then the first
              location in the leaf-list MUST be the definitive location
              that uniquely identifies this package";
         }

         leaf checksum {
           type pkg-types:sha-256-hash;
           description
             "The SHA-256 hash calculated on the textual package
              definition, represented as a hexadecimal string.";
         }
       }

       list module {
         key "name";
         description
           "An entry in this list represents a module that must be
            implemented by a server implementing this package, as per
            RFC 7950 section 5.6.5, with a particular set of supported
```

```
             features and deviations.

             A entry in this list overrides any module revision
             'implemented' by an included package.  Any replaced module
             revision SHOULD also be listed in the 'replaces-revision'
             list.";
          reference
            "RFC 7950: The YANG 1.1 Data Modeling Language.";

          leaf name {
            type yang:yang-identifier;
            mandatory true;
            description
              "The YANG module name.";
          }

          leaf revision {
            type rev:revision-date-or-label;
            description
              "The YANG module revision date or revision-label.

               If no revision statement is present in the YANG module,
               this leaf is not instantiated.";
          }

          leaf-list replaces-revision {
            type rev:revision-date-or-label;
            description
              "Gives the revision of an module (implemented or
               import-only) defined in an included package that is
               replaced by this implemented module revision.";
          }

          leaf namespace {
            type inet:uri;
            description
              "The XML namespace identifier for this module.";
          }

          leaf-list location {
            type inet:uri;
            description
              "Contains a URL that represents the YANG schema resource
               for this module.

               This leaf will only be present if there is a URL available
               for retrieval of the schema for this entry.";
          }
```

```
         leaf checksum {
           type pkg-types:sha-256-hash;
           description
             "The SHA-256 hash calculated on the textual module
              definition, represented as a hexadecimal string.";
         }

         list submodule {
           key "name";
           description
             "Each entry represents one submodule within the
              parent module.";

           leaf name {
             type yang:yang-identifier;
             description
               "The YANG submodule name.";
           }

           leaf revision {
             type yang:revision-identifier;
             mandatory true;
             description
               "The YANG submodule revision date.  If the parent module
                include statement for this submodule includes a revision
                date then it MUST match this leaf's value.";
           }

           leaf-list location {
             type inet:uri;
             description
               "Contains a URL that represents the YANG schema resource
                for this submodule.

                This leaf will only be present if there is a URL
                available for retrieval of the schema for this entry.";
           }

           leaf checksum {
             type pkg-types:sha-256-hash;
             description
               "The SHA-256 hash calculated on the textual submodule
                definition, represented as a hexadecimal string.";
           }
         }
       }

       list import-only-module {
         key "name revision";
         description
           "An entry in this list indicates that the server imports
            reusable definitions from the specified revision of the
            module, but does not implement any protocol accessible
            objects from this revision.

            Multiple entries for the same module name MAY exist.  This
            can occur if multiple modules import the same module, but
            specify different revision-dates in the import statements.";

         leaf name {
           type yang:yang-identifier;
           description
             "The YANG module name.";
         }

         leaf revision {
           type rev:revision-date-or-label;
           description
             "The YANG module revision date or revision-label.

              If no revision statement is present in the YANG module,
              this leaf is not instantiated.";
         }

         leaf-list replaces-revision {
           type rev:revision-date-or-label;
           description
             "Gives the revision of an import-only-module defined in an
              included package that is replaced by this
              import-only-module revision.";
         }

         leaf namespace {
           type inet:uri;
           description
             "The XML namespace identifier for this module.";
         }

         leaf-list location {
           type inet:uri;
           description
             "Contains a URL that represents the YANG schema resource
              for this module.

              This leaf will only be present if there is a URL available
              for retrieval of the schema for this entry.";
         }

         leaf checksum {
           type pkg-types:sha-256-hash;
           description
             "The SHA-256 hash calculated on the textual submodule
              definition, represented as a hexadecimal string.";
         }

         list submodule {
           key "name";
           description
             "Each entry represents one submodule within the
              parent module.";

           leaf name {
             type yang:yang-identifier;
             description
               "The YANG submodule name.";
           }

           leaf revision {
             type yang:revision-identifier;
             mandatory true;
             description
               "The YANG submodule revision date.  If the parent module
                include statement for this submodule includes a revision
                date then it MUST match this leaf's value.";
           }

           leaf-list location {
             type inet:uri;
             description
               "Contains a URL that represents the YANG schema resource
                for this submodule.

                This leaf will only be present if there is a URL
                available for retrieval of the schema for this entry.";
           }

           leaf checksum {
             type pkg-types:sha-256-hash;
             description
               "The SHA-256 hash calculated on the textual submodule
                definition, represented as a hexadecimal string.";
           }
         }
       }
```

```
     }
   }
   <CODE ENDS>



   <CODE BEGINS> file "ietf-yang-package-instance@2020-01-21.yang"
   module ietf-yang-package-instance {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package-instance";
     prefix pkg-inst;

     import ietf-yang-revisions {
       prefix rev;
       reference "XXXX: Updated YANG Module Revision Handling";
     }

     import ietf-yang-package-types {
       prefix pkg-types;
       rev:revision-or-derived 0.2.0;
       reference "RFC XXX: YANG Schema Versioning.";
     }

     import ietf-yang-structure-ext {
       prefix sx;
       reference "RFC XXX: YANG Data Structure Extensions.";
     }

     organization
       "IETF NETMOD (Network Modeling) Working Group";

     contact
       "WG Web:   <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>

        Author:   Rob Wilton
                  <mailto:rwilton@cisco.com>";

     description
       "This module provides a definition of a YANG package, which is
        used as the schema for an YANG instance data document specifying
        a YANG package.

        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
        'MAY', and 'OPTIONAL' in this document are to be interpreted as
        described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     revision 2020-01-21 {
       rev:revision-label 0.2.0;
       description
         "Initial revision";
       reference
         "RFC XXXX: YANG Packages";
     }


     /*
      * Top-level structure
      */

     sx:structure package {
       description
         "Defines the YANG package structure for use in a YANG instance
          data document.";

       uses pkg-types:yang-pkg-instance;
     }
   }
   <CODE ENDS>



   <CODE BEGINS> file "ietf-yang-package@2020-01-21.yang"
   module ietf-yang-packages {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-yang-packages";
     prefix pkgs;
     import ietf-yang-revisions {
       prefix rev;
       reference "XXXX: Updated YANG Module Revision Handling";
     }

     import ietf-yang-package-types {
       prefix pkg-types;
       rev:revision-or-derived 0.2.0;
       reference "RFC XXX: YANG Packages.";
     }

     import ietf-inet-types {
       prefix inet;
       rev:revision-or-derived 2013-07-15;
       reference "RFC 6991: Common YANG Data Types.";
     }

     organization
       "IETF NETMOD (Network Modeling) Working Group";

     contact
       "WG Web:   <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>

        Author:   Rob Wilton
                  <mailto:rwilton@cisco.com>";

     description
       "This module defines YANG packages on a server implementation.

        Copyright (c) 2018 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
        'MAY', and 'OPTIONAL' in this document are to be interpreted as
        described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";
     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     revision 2020-01-21 {
       rev:revision-label 0.2.0;
       description
         "Initial revision";
       reference
         "RFC XXXX: YANG Packages";
     }


     /*
      * Groupings
      */

     grouping yang-pkg-ref {
       description
         "Defines the leaves used to reference a single YANG package";

       leaf name {
         type leafref {
           path '/pkgs:packages/pkgs:package/pkgs:name';
         }
         description
           "The name of the references package.";
       }

       leaf version {
         type leafref {
           path '/pkgs:packages'
             + '/pkgs:package[pkgs:name = current()/../name]'
             + '/pkgs:version';
         }

         description
           "The version of the referenced package.";
       }

       leaf checksum {
         type leafref {
           path '/pkgs:packages'
             + '/pkgs:package[pkgs:name = current()/../name]'
             + '[pkgs:version = current()/../version]/pkgs:checksum';
         }

         description
```

```
           "The checksum of the referenced package.";
         }
       }

       grouping yang-ds-pkg-ref {
         description
           "Defines the list used to reference a set of YANG packages that
            collectively represent a datastore schema.";

         list package {
           key "name version";

           description
             "Identifies the YANG packages that collectively defines the
              schema for the associated datastore.

              The datastore schema is defined as the union of all
              referenced packages, that MUST represent a referentially
              complete schema.

              All of the referenced packages must be compatible with no
              conflicting module versions or dependencies.";

           uses yang-pkg-ref;
         }
       }


       /*
        * Top level data nodes.
        */

       container packages {
         config false;
         description "All YANG package definitions";

         list package {
           key "name version";

           description
             "YANG package instance";

           uses pkg-types:yang-pkg-instance;

           leaf-list location {
             type inet:uri;
             description
               "Contains a URL that represents where an instance data file
              for this YANG package can be found.

              This leaf will only be present if there is a URL available
              for retrieval of the schema for this entry.

              If multiple locations are provided, then the first
              location in the leaf-list MUST be the definitive location
              that uniquely identifies this package";
         }

         leaf checksum {
           type pkg-types:sha-256-hash;
           description
             "The checksum of the package this schema relates to,
              calculated on the 'YANG instance data file' package
              definition available in the 'location' leaf list.

              This leaf MAY be omitted if the referenced package is
              locally scoped without an associated checksum.";
         }
       }
     }
   }
   <CODE ENDS>



   <CODE BEGINS> file "ietf-yl-package@2020-01-21.yang"
   module ietf-yl-packages {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-yl-packages";
     prefix yl-pkgs;

     import ietf-yang-revisions {
       prefix rev;
       reference "XXXX: Updated YANG Module Revision Handling";
     }

     import ietf-yang-packages {
       prefix pkgs;
       rev:revision-or-derived 0.2.0;
       reference "RFC XXX: YANG Packages.";
     }

     import ietf-yang-library {
       prefix yanglib;
       rev:revision-or-derived 2019-01-04;
       reference "RFC 8525: YANG Library";
     }

     organization
       "IETF NETMOD (Network Modeling) Working Group";

     contact
       "WG Web:   <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>

        Author:   Rob Wilton
                  <mailto:rwilton@cisco.com>";

     description
       "This module provides defined augmentations to YANG library to
        allow a server to report YANG package information.

        Copyright (c) 2018 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
        'MAY', and 'OPTIONAL' in this document are to be interpreted as
        described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";


     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     revision 2020-01-21 {
       rev:revision-label 0.2.0;
       description
         "Initial revision";
       reference
         "RFC XXXX: YANG Packages";
     }
     /*
      * Augmentations
      */

     augment "/yanglib:yang-library/yanglib:schema" {
       description
         "Allow datastore schema to be related to a set of YANG
          packages";

       uses pkgs:yang-ds-pkg-ref;
     }
   }
   <CODE ENDS>



   <CODE BEGINS> file "ietf-yang-inst-data-pkg@2020-01-21.yang"
   module ietf-yang-inst-data-pkg {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-yang-inst-data-pkg";
     prefix yid-pkg;

     import ietf-yang-revisions {
       prefix rev;
       reference "XXXX: Updated YANG Module Revision Handling";
     }

     import ietf-yang-package-types {
       prefix pkg-types;
       rev:revision-or-derived 0.2.0;
       reference "RFC XXX: YANG Schema Versioning.";
     }

     import ietf-yang-structure-ext {
       prefix sx;
       reference "RFC XXX: YANG Data Structure Extensions.";
     }

     import ietf-yang-instance-data {
       prefix yid;
       reference "RFC XXX: YANG Instance Data File Format.";
     }

     import ietf-inet-types {
       prefix inet;
       reference "RFC 6991: Common YANG Data Types.";
     }
     organization
       "IETF NETMOD (Network Modeling) Working Group";

     contact
       "WG Web:   <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>

        Author:   Rob Wilton
                  <mailto:rwilton@cisco.com>";

     description
       "The module augments ietf-yang-instance-data to allow package
        definitions to be used to define schema in YANG instance data
        documents.

        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
        'MAY', and 'OPTIONAL' in this document are to be interpreted as
        described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     revision 2020-01-21 {
       rev:revision-label 0.2.0;
       description
         "Initial revision";
       reference
         "RFC XXXX: YANG Packages";
     }

     /*
      * Augmentations
      */

     sx:augment-structure
       "/yid:instance-data-set/yid:content-schema-spec" {
       description
         "Add package reference to instance data set schema
          specification";
       case pkg-schema {
         container pkg-schema {
           uses pkg-types:yang-pkg-identification-leafs;

           leaf checksum {
             type pkg-types:sha-256-hash;
             description
               "The SHA-256 hash of the package, calculated on
                the textual package definition, represented as a
                hexadecimal string.";
           }

           leaf-list location {
             type inet:uri;
             description
               "Contains a URL that represents where an instance data
                file for this YANG package can be found.

                This leaf will only be present if there is a URL
                available for retrieval of the schema for this entry.

                If multiple locations are provided, then the first
                location in the leaf-list MUST be the definitive
                location that uniquely identifies this package";
           }
         }
       }
     }
   }
   <CODE ENDS>
```

11. Security Considerations

The YANG modules specified in this document defines a schema for data that is accessed by network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

Similarly to YANG library [I-D.ietf-netconf-rfc7895bis], some of the readable data nodes in these YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes.

One additional key different to YANG library, is that the 'ietf-yang-package' YANG module defines a schema to allow YANG packages to be defined in YANG instance data files, that are outside the security controls of the network management protocols. Hence, it is important to also consider controlling access to these package instance data files to restrict access to sensitive information. SHA-256 checksums are used to ensure the integrity of YANG package definitions, imported modules, and sub-modules.

As per the YANG library security considerations, the module, revision and version information in YANG packages may help an attacker identify the server capabilities and server implementations with known bugs since the set of YANG modules supported by a server may reveal the kind of device and the manufacturer of the device. Server vulnerabilities may be specific to particular modules, module revisions, module features, or even module deviations. For example, if a particular operation on a particular data node is known to cause a server to crash or significantly degrade device performance, then the YANG packages information will help an attacker identify server implementations with such a defect, in order to launch a denial-of-service attack on the device.

12. IANA Considerations

It is expected that a central registry of standard YANG package definitions is required to support this solution.

It is unclear whether an IANA registry is also required to manage specific package versions. It is highly desirable to have a specific canonical location, under IETF control, where the definitive YANG package versions can be obtained from.

This document requests IANA to registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations are requested.

```
URI: urn:ietf:params:xml:ns:yang:ietf-yang-package-types.yang
```

Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-package-instance.yang
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-packages.yang
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-yl-packages.yang
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-inst-data-pkg.yang
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document requests that the following YANG modules are added in the "YANG Module Names" registry [RFC6020]:

Name: ietf-yang-package-types.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-package-types.yang
Prefix: pkg-types
Reference: RFC XXXX

Name: ietf-yang-package-instance.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-package-instance.yang
Prefix: pkg-inst
Reference: RFC XXXX

Name: ietf-yang-packages.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-packages.yang
Prefix: pkgs
Reference: RFC XXXX

Name: ietf-yl-packages.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yl-packages.yang
Prefix: yl-pkgs
Reference: RFC XXXX

Name: ietf-yang-inst-data-pkg.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-inst-data-pkg.yang
Prefix: yid-pkg

Reference: RFC XXXX

13. Open Questions/Issues

All issues, along with the draft text, are currently being tracked at https://github.com/rgwilton/YANG-Packages-Draft/issues/

14. Acknowledgements

Feedback helping shape this document has kindly been provided by Andy Bierman, James Cumming, Mahesh Jethanandani, Balazs Lengyel, Ladislav Lhotka,and Jan Lindblad.

15. References

15.1. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.

[I-D.ietf-netmod-module-tags]
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", draft-ietf-netmod-module-tags-10 (work in progress), February 2020.

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Structure Extensions", draft-ietf-netmod-yang-data-ext-05 (work in progress), December 2019.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-12 (work in progress), April 2020.

[I-D.ietf-netmod-yang-module-versioning]
Wilton, R., Rahman, R., Lengyel, B., Clarke, J., Sterne, J., Claise, B., and K. D'Souza, "Updated YANG Module Revision Handling", draft-ietf-netmod-yang-module-versioning-01 (work in progress), July 2020.

[I-D.ietf-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-ietf-netmod-yang-semver-01 (work in progress), July 2020.

[I-D.ietf-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-ietf-netmod-yang-solutions-00 (work in progress), March 2020.

[I-D.ietf-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Schema Selection", draft-ietf-netmod-yang-ver-selection-00 (work in progress), March 2020.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-ietf-netmod-yang-versioning-reqs-03 (work in progress), June 2020.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

15.2. Informative References

[I-D.bierman-netmod-yang-package]
Bierman, A., "The YANG Package Statement", draft-bierman-netmod-yang-package-00 (work in progress), July 2015.

[I-D.ietf-netmod-artwork-folding]
Watsen, K., Auerswald, E., Farrel, A., and Q. WU, "Handling Long Lines in Inclusions in Internet-Drafts and RFCs", draft-ietf-netmod-artwork-folding-12 (work in progress), January 2020.

[openconfigsemver]
"Semantic Versioning for OpenConfig Models", <http://www.openconfig.net/docs/semver/>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

Appendix A. Examples

This section provides various examples of YANG packages, and as such this text is non-normative. The purpose of the examples is to only illustrate the file format of YANG packages, and how package dependencies work. It does not imply that such packages will be defined by IETF, or which modules would be included in those packages even if they were defined. For brevity, the examples exclude namespace declarations, and use a shortened URL of "tiny.cc/ietf-yang" as a replacement for "https://raw.githubusercontent.com/YangModels/yang/master/standard/ietf/RFC".

A.1. Example IETF Network Device YANG package

This section provides an instance data file example of an IETF Network Device YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, to implement a basic network device without any dynamic routing or layer 2 services. E.g., it includes functionality such as system information, interface and basic IP configuration.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, they modules are referenced by revision date rather than version number.

```
<CODE BEGINS> file "example-ietf-network-device-pkg.json"
========= NOTE: '\' line wrapping per BCP XX (RFC XXXX) ===========

 {
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ietf-network-device-pkg",
    "pkg-schema": {
       package: "ietf-yang-package-defn-pkg@0.1.0.json"
    },
    "description": "YANG package definition",
    "content-data": {
      "ietf-yang-package-instance:yang-package": {
        "name": "example-ietf-network-device-pkg",
        "version": "1.1.2",
        "timestamp": "2018-12-13T17:00:00Z",
        "organization": "IETF NETMOD Working Group",
        "contact" : "WG Web:   <http://tools.ietf.org/wg/netmod/>, \
                     WG List:  <mailto:netmod@ietf.org>",
        "description": "Example IETF network device YANG package.\
           \
           This package defines a small sample set of \
           YANG modules that could represent the basic set of \
           modules that a standard network device might be expected \
           to support.",
          "reference": "XXX, draft-rwilton-netmod-yang-packages",
          "location": [ "file://example.org/yang/packages/\
                           ietf-network-device@v1.1.2.json" ],
          "module": [
            {
              "name": "iana-crypt-hash",
              "revision": "2014-08-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                         iana-crypt-hash%402014-08-06.yang" ],
              "checksum": "fa9fde408ddec2c16bf2c6b9e4c2f80b\
                      813a2f9e48c127016f3fa96da346e02d"
            },
            {
              "name": "ietf-system",
              "revision": "2014-08-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                             ietf-system%402014-08-06.yang" ],
              "checksum": "8a692ee2521b4ffe87a88303a61a1038\
                      79ee26bff050c1b05a2027ae23205d3f"
            },
            {
              "name": "ietf-interfaces",
              "revision": "2018-02-20",
              "location": [ "https://tiny.cc/ietf-yang/\
                         ietf-interfaces%402018-02-20.yang" ],
              "checksum": "f6faea9938f0341ed48fda93dba9a69a\
                      a32ee7142c463342efec3d38f4eb3621"
            },
            {
              "name": "ietf-netconf-acm",
              "revision": "2018-02-14",
              "location": [ "https://tiny.cc/ietf-yang/\
                        ietf-netconf-acm%402018-02-14.yang" ],
              "checksum": "e03f91317f9538a89296e99df3ff0c40\
                      03cdfea70bf517407643b3ec13c1ed25"
            },
            {
              "name": "ietf-key-chain",
              "revision": "2017-06-15",
              "location": [ "https://tiny.cc/ietf-yang/\
                            ietf-key-chain@2017-06-15.yang" ],
              "checksum": "6250705f59fc9ad786e8d74172ce90d5\
                      8deec437982cbca7922af40b3ae8107c"
            },
            {
              "name": "ietf-ip",
              "revision": "2018-02-22",
              "location": [ "https://tiny.cc/ietf-yang/\
                                          ietf-ip%402018-02-22.yang" ],
               "checksum": "b624c84a66c128ae69ab107a5179ca8e\
                            20e693fb57dbe5cb56c3db2ebb18c894"
           }
           ],
           "import-only-module": [
             {
               "name": "ietf-yang-types",
               "revision": "2013-07-15",
               "location": [ "https://tiny.cc/ietf-yang/\
                                   ietf-yang-types%402013-07-15.yang" ],
               "checksum": "a04cdcc875764a76e89b7a0200c6b9d8\
                            00b10713978093acda7840c7c2907c3f"
             },
             {
               "name": "ietf-inet-types",
               "revision": "2013-07-15",
               "location": [ "https://tiny.cc/ietf-yang/\
                                   ietf-inet-types%402013-07-15.yang" ],
               "checksum": "12d98b0143a5ca5095b36420f9ebc1ff\
                            a61cfd2eaa850080244cadf01b86ddf9"
             }
           ]
         }
       }
     }
   }
   <CODE ENDS>
```

A.2. Example IETF Basic Routing YANG package

This section provides an instance data file example of a basic IETF Routing YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, that builds upon the example-ietf-network-device YANG package to add support for basic dynamic routing and ACLs.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, they modules are referenced by revision date rather than version number. Locations have been excluded where they are not currently known, e.g., for YANG modules defined in IETF drafts. In a normal YANG package, locations would be expected to be provided for all YANG modules.

```
   <CODE BEGINS> file "example-ietf-routing-pkg.json"
   ========== NOTE: '\' line wrapping per BCP XX (RFC XXXX) ===========

   {
     "ietf-yang-instance-data:instance-data-set": {
       "name": "example-ietf-routing-pkg",
       "module": [ "ietf-yang-package@2019-09-11.yang" ],
       "description": "YANG package definition",
       "content-data": {
         "ietf-yang-package-instance:yang-package": {
           "name": "example-ietf-routing",
           "version": "1.3.1",
           "timestamp": "2018-12-13T17:00:00Z",
           "description": "This package defines a small sample set of \
             IETF routing YANG modules that could represent the set of \
             IETF routing functionality that a basic IP network device \
             might be expected to support.",
           "reference": "XXX, draft-rwilton-netmod-yang-packages",
           "imported-packages": [
             {
               "name": "ietf-network-device",
               "version": "1.1.2",
               "location": [ "http://example.org/yang/packages/\
                             ietf-network-device@v1.1.2.json" ],
               "checksum": ""
             }
           ],
           "module": [
             {
               "name": "ietf-routing",
               "revision": "2018-03-13",
               "location": [ "https://tiny.cc/ietf-yang/\
                                ietf-routing@2018-03-13.yang" ],
               "checksum": ""
             },
             {
               "name": "ietf-ipv4-unicast-routing",
               "revision": "2018-03-13",
               "location": [ "https://tiny.cc/ietf-yang/\
                         ietf-ipv4-unicast-routing@2018-03-13.yang" ],
               "checksum": ""
             },
             {
               "name": "ietf-ipv6-unicast-routing",
               "revision": "2018-03-13",
               "location": [ "https://tiny.cc/ietf-yang/\
                         ietf-ipv6-unicast-routing@2018-03-13.yang" ],
               "checksum": ""
```

```
            },
            {
              "name": "ietf-isis",
              "revision": "2018-12-11",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-interfaces-common",
              "revision": "2018-07-02",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-if-l3-vlan",
              "revision": "2017-10-30",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-routing-policy",
              "revision": "2018-10-19",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-bgp",
              "revision": "2018-05-09",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-access-control-list",
              "revision": "2018-11-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            }
          ],
          "import-only-module": [
            {
              "name": "ietf-routing-types",
              "revision": "2017-12-04",
              "location": [ "https://tiny.cc/ietf-yang/\
                            ietf-routing-types@2017-12-04.yang" ],
              "checksum": ""
            },
            {
              "name": "iana-routing-types",
              "revision": "2017-12-04",
              "location": [ "https://tiny.cc/ietf-yang/\
                            iana-routing-types@2017-12-04.yang" ],
              "checksum": ""
            },
            {
              "name": "ietf-bgp-types",
              "revision": "2018-05-09",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-packet-fields",
              "revision": "2018-11-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-ethertypes",
              "revision": "2018-11-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            }
          ]
        }
      }
    }
  }
  <CODE ENDS>
```

A.3. Package import conflict resolution example

This section provides an example of how a package can resolve conflicting module versions from imported packages.

In this example, YANG package 'example-3-pkg' imports both 'example-import-1' and 'example-import-2' packages. However, the two imported packages implement different versions of 'example-module-A' so the 'example-3-pkg' package selects version '1.2.3' to resolve the conflict. Similarly, for import-only modules, the 'example-3-pkg' package does not require both versions of example-types-module-C to be imported, so it indicates that it only imports revision '2018-11-26' and not '2018-01-01'.

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-import-1-pkg",
    "description": "First imported example package",
    "content-data": {
      "ietf-yang-package-instance:yang-package": {
        "name": "example-import-1",
        "version": "1.0.0",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-01-01",
        "module": [
          {
            "name": "example-module-A",
            "version": "1.0.0"
          },
          {
            "name": "example-module-B",
            "version": "1.0.0"
          }
        ],
        "import-only-module": [
          {
            "name": "example-types-module-C",
            "revision": "2018-01-01"
          },
          {
            "name": "example-types-module-D",
            "revision": "2018-01-01"
          }
        ]
      }
    }
  }
}

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-import-2-pkg",
    "description": "Second imported example package",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-import-2",
        "version": "2.0.0",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-11-26",
        "module": [
          {
            "name": "example-module-A",
            "version": "1.2.3"
          },
          {
            "name": "example-module-E",
            "version": "1.1.0"
          }
        ],
        "import-only-module": [
          {
            "name": "example-types-module-C",
            "revision": "2018-11-26"
          },
          {
            "name": "example-types-module-D",
            "revision": "2018-11-26"
          }
        ]
      }
    }
  }
}

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-3-pkg",
    "description": "Importing example package",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-3",
        "version": "1.0.0",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-11-26",
        "included-package": [
          {
            "name": "example-import-1",
            "version": "1.0.0"
          },
          {
            "name": "example-import-2",
            "version": "2.0.0"
          }
        ],
        "module": [
          {
            "name": "example-module-A",
            "version": "1.2.3"
          }
        ],
        "import-only-module": [
          {
            "name": "example-types-module-C",
            "revision": "2018-11-26",
            "replaces-revision": [ "2018-01-01 "]
          }
        ]
      }
    }
  }
}
```

Appendix B. Possible alternative solutions

This section briefly describes some alternative solutions. It can be removed if this document is adopted as a WG draft.

B.1. Using module tags

Module tags have been suggested as an alternative solution, and indeed that can address some of the same requirements as YANG packages but not all of them.

Module tags can be used to group or organize YANG modules. However, this raises the question of where this tag information is stored. Module tags either require that the YANG module files themselves are updated with the module tag information (creating another versioning problem), or for the module tag information to be hosted elsewhere, perhaps in a centralize YANG Catalog, or in instance data files similar to how YANG packages have been defined in this draft.

One of the principle aims of YANG packages is to be a versioned object that defines a precise set of YANG modules versions that work together. Module tags cannot meet this aim without an explosion of module tags definitions (i.e. a separate module tag must be defined for each package version).

Module tags cannot support the hierachical scheme to construct YANG schema that is proposed in this draft.

B.2. Using YANG library

Another question is whether it is necessary to define new YANG modules to define YANG packages, and whether YANG library could just be reused in an instance data file. The use of YANG packages offers several benefits over just using YANG library:

1. Packages allow schema to be built in a hierarchical fashion. [I-D.ietf-netconf-rfc7895bis] only allows one layer of hierarchy (using module sets), and there must be no conflicts between module revisions in different module-sets.

2. Packages can be made available off the box, with a well defined unique name, avoiding the need for clients to download, and construct/check the entire YANG schema for each device. Instead they can rely on the named packages with secure checksums. YANG library's use of a 'content-id' is unique only to the device that generated them.

3. Packages may be versioned using a semantic versioning scheme, YANG library does not provide a schema level semantic version number.

4. For a YANG library instance data file to contain the necessary information, it probably needs both YANG library and various augmentations (e.g. to include each module's semantic version number), unless a new version of YANG library is defined containing this information. The module definition for a YANG package is specified to contain all of the ncessary information to solve the problem without augmentations

5. YANG library is designed to publish information about the modules, datastores, and datastore schema used by a server. The information required to construct an off box schema is not precisely the same, and hence the definitions might deviate from each other over time.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Joe Clarke
Cisco Systems, Inc.

Email: jclarke@cisco.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Bo Wu (editor)
Huawei

Email: lana.wubo@huawei.com