Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman
Equinix
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia

2 March 2026

YANG Packages
draft-ietf-netmod-yang-packages-07

Abstract

This document defines YANG packages; a versioned organizational structure used to manage schema and conformance of YANG modules as a cohesive set instead of individually.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 September 2026.

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 the following terminology introduced in YANG Semantic Versioning [I-D.ietf-netmod-yang-semver]:

* YANG Semver

This document uses the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

* datastore schema

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

* data node

* schema node

In addition, this document defines the following terminology:

* version (module): In the context of a YANG module, it is used as a short hand to reference a particular revision of a YANG module, potentially identified by a YANG Semver version label or a revision-date.

* version (package): In the context of a YANG package, it refers to a particular version of a YANG package definition, identified by the package version field.

* YANG schema: The combined set of schema nodes for a set of modules taking into account implemented modules, import-only modules, deviations, features and mount-points. A more complete definition is provided in Section 4.

* YANG package: a versioned organizational structure used to manage a set of YANG modules that collectively define a package schema. YANG packages are defined in Section 3. Depending on context, the term 'YANG package' is often used to refer to a specific version of a YANG package rather than all versions of a package definition.

* package: An alternative term for 'YANG package'.

* 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 6.1.1.2.

* 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 6.1.1.1.

* editorial change: When used in the context of a YANG module, it follows the definition of an 'editorial change' in 4.4 of [I-D.ietf-netmod-yang-semver]. When used in the context of a YANG package, it follows the definition in Section 6.1.1.3.

* resolved package schema: The resolved package schema is the YANG schema defined by a package after package resolution has been performed, as defined in Section 4. The resolved package schema identifies the implemented modules (with any deviations applied), import-only modules, enabled features, and schema mount points.

* package schema: An alternative term for 'resolved package schema'.

* mandatory-feature: A YANG module feature, declared by a package definition as being mandatory to implement for all implementations that conform to the package definition.

2. Introduction

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

A YANG package is a versioned hierarchical organizational structure used to manage a set of YANG modules that collectively define a package schema. 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 be exported from a server, accessed as instance data files [RFC9195], augmented into YANG library [RFC8525], and used by tooling to compare and manage YANG schema.

Examples of YANG packages are provided in the appendices.

2.1. Main Objectives

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

* Providing 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).

* Allowing datastore 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 may be 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.

* YANG Packages should be able to represent the equivalent structure as YANG library, but making use of a hierarchical resolution mechanism.

* YANG Packages should be flexible enough to provide usable definitions representing collections of IETF YANG modules, OpenConfig YANG modules, and other bespoke sets of YANG modules, e.g., covering sets of vendor native YANG models.

* YANG packages should be flexible enough to represent the conformance and server implementations of standard or industry defined YANG package definitions. E.g., it should be possible for a server implementation to indicate that it does not faithfully implement a package schema, e.g., by excluding modules, implementing different module versions/revisions, and/or having deviations applied.

* Tooling should be able to easily work with YANG package definitions to compare YANG package versions and to compare server conformance against expected package definitions.

Protocol mechanisms of how clients can negotiate which YANG packages or YANG package versions are to be used for NETCONF/RESTCONF communications are outside the scope of this document. One potential mechanism is 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 the 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.

2.2. Potential Alternative Mechanisms

There are several alternative approaches to managing YANG schema. These include:

* Using YANG library, along with YANG Instance Data files [RFC9195],

* Using git tags and version labels for modules maintained on github,

* As collections of YANG modules in a zip file or at a directory folder. E.g., at time of publication, this method is used to represent the set of YANG modules associated with particular vendor release at the github repository at https://github.com/YangModels/yang

Although these methods are quite simple, there are some disadvantages with various aspects of these methods: They are verbose, they don't advertise supported features or support mounts, and they can be awkward to compare, particularly if YANG modules haven't been versioned correctly.

2.3. Open Questions/Issues

RFC Editor, please remove this section before publication.

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

3. The YANG Package Definition

A YANG package is a versioned organizational structure used to manage a set of YANG modules that collectively define a package schema.

Each YANG package defines:

* YANG package meta-data, such as "name", "version", "organization", "description", "complete" flag, etc.

* An "includes" container holding a list of included packages. It also contains lists of any implemented modules and import-only modules that are used in addition to, or with different version/ revisions to, the modules defined in the included packages. Finally, it lists required features in addition to those defined in included packages.

* An "excludes" container comprising modules and features that are included via the resolved packages of entries in the "includes/packages" list, but that are excluded from this package definition. It is not possible to exclude packages.

* Lists of YANG packages that will be found at particular mount points by any server implementing this package, used in conjunction with mount points defined by any included packages.

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 "yang-pkg-instance" grouping in the "ietf-yang-package-types" YANG module has the following structure:

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

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

3.1. Package definition rules

For a YANG package to be valid, it MUST conform to all the following rules:

1. Each (package name, version) pairing MUST define a globally unique version of that package definition.

2. Each YANG package has a name that SHOULD end with the suffix "-pkg". The name MUST be globally unique to avoid issues with tools and caching, e.g., using the mechanisms specified in Section 5.1.

3. YANG packages MUST be versioned using YANG Semver, [I-D.ietf-netmod-yang-semver]. Versioning YANG packages is further described in Section 6.1.

4. A YANG package MAY represent a referentially complete set of modules or MAY represent a set of modules with some module import dependencies missing, as described in Section 3.2.

5. Packages definitions are hierarchical because a package can include other packages. There MUST NOT be any circular package include dependencies, i.e., packages cannot, directly or indirectly, include themselves.

6. For each module implemented by a package, only a single version/ revision of that module MUST be implemented. Conflicting module versions (e.g. from package includes) MAY be resolved explicitly (via "includes/module") or using automatic module version resolution, as described in Section 4.1.

7. Multiple versions/revisions of an import-only module MAY be listed, but any extraneous import-only module versions SHOULD be removed.

8. A package definition MUST NOT include the same module name in both the ’includes/module’ and ’excludes/module’ lists.

9. A package definition MUST NOT include the same module name in both the ’includes/import-only-module’ and ’excludes/import-only-module’ lists.

10. A package definition MUST NOT include the same feature name in both the ’includes/feature’ and ’excludes/feature’ lists.

11. A package definition MUST NOT exclude a module and list features defined by that module in the ’includes/feature’ list.

12. A package definition MAY include redundant information, e.g., including a module or package version that is already present by an included package, or excluding a module that is not included by any included package. However, such redundant information might be confusing to readers. Although the resolved package definition is unambiguous, it is best to minimize redundant information where possible.

13. Finally, standard rules for YANG instance data apply. E.g., entries in the various lists MUST be unique by any list key.

3.2. 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 being 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 being referentially incomplete.

Also see Section 5.3 for details on cases when referentially incomplete packages are helpful.

3.3. 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 schema to be fully constructed from a YANG package instance data file definition.

3.4. Package Mount Points

YANG Schema Mount [RFC8528] defines a mechanism for YANG modules to be mounted at specific mount points in the schema tree. This mechanism is required to instantiate the full schema for some common networking use cases, e.g., [RFC8529] defines a YANG Model for Network Instances, that uses YANG mount points to mount IETF routing protocol YANG modules within the network instance list.

[RFC8528] declares that it provides support for mounted schema at "Implementation time" and "Run time", but does not cover mounting schema at "Design time". YANG Package definitions do not give YANG language level "Design time" guarantees, but they are able to give a stronger "Implementation time" guarantee through the use of offline YANG package definitions. They can also be used to report "Run time" mounted schema behaviour, if the server is able to report the packages implemented by the device.

Each YANG package definition defines a set of packages that will be found at a particular mount point, via two mechanisms:

* All mounted packages that are exported from any included packages are also exported at the same mount point for this specific package, unless explicitly overridden by a mount definition in this package with a ’replaces-package’ statement.

* A package definition CANNOT remove a package from a mount point, but it MAY add another different package version, which SHOULD be newer version of the package. A package definition MAY also remove modules or mandatory-features. In all cases, this is achieved by listing the replacement package in the ’mount/package’ list, which MUST include a version of the package to be replaced. Examples of this are given in Appendix A.6.2.

3.4.1. Exclusions

A package definition may exclude modules or mandatory-features. The mechanism for this is described in Section 4, and the conformance implications are described in Section 7.

4. Package Resolution

Package resolution is taking a YANG package definition and converting it to a specification for a YANG schema, e.g., as may be implemented by a device for a particular datastore. A YANG schema can be thought of as comprising:

A set of implemented modules at specific versions,

A set of import-only modules, potentially with multiple versions of a given module,

A set of mandatory-features that are supported,

A set of mount points, each with its own YANG schema,

that can be collectively compiled into a YANG schema tree.

The YANG schema generated by a package definition can be converted into an equivalent instance in YANG library [RFC8525], with Schema Mount [RFC8528] if mount points are used. E.g., see Section 5.4.3.

The following process defines how a YANG a package definition is resolved, using two steps:

1. The list of included packages are each recursively resolved using these steps and then the resolved package schema is combined using the following merge resolution rules:

    * The set of implemented modules is the union of all implemented modules in the resolved included packages. Conflicting module versions can be resolved automatically as per Section 4.1, then overwritten (including submodule and location information) by any entries in the "includes/module" list, and finally filtered by any entries in "excludes/module" list.

    * The set of import-only modules is the union of all import-only modules in the resolved included packages, then updated by any entries in the "includes/import-only-module" list, and finally filtered by any entries in "excludes/import-only-module" list.

    * The set of mandatory-features is the union of the mandatory features in the resolved included packages, with any "includes/feature" entries added, and any "excludes/feature" entries removed. If a module is excluded by "excludes/module" then all features associated with that module are also implicitly removed.

    * The set of mounts is the union of the mounts in the resolved included packages, where for a given mount-path that is present in more than one included package (same path and same keys) then it takes the union of the mounted packages and mount parent-references. The mounts are then updated by combining any entries in the package's "mount" list, with any entries listed in "mount/package/wraps-package" list replaced by a package that wraps the original package at the mount point.

    * If the same module version is being resolved from multiple included modules, then the union of the module locations is used.

    * Submodules are ignored for resolution purposes, only the module version is considered and compared.

See https://github.com/netmod-wg/yang-ver-dt/issues/258. How to handle union of meta-data, locations, and override behaviours.

4.1. Automatic Module Version Resolution

A resolved package definition can only advertise a single version of a module, and hence when conflicting module versions arise though included packages that resolve to different versions of a given module then a single version has to be chosen. The following rules are used, in the order given, to automatically select the chosen version by performing a pairwise comparison of the module versions from the resolved included packages, by comparing the version leaf:

1. If both the version leaf matches the YANG Semver format, then considering section 4.4 or [I-D.ietf-netmod-yang-semver]:

   i The module with the highest MAJOR version component is chosen.

   ii Otherwise, if the MAJOR version components are the same, then the module with the highest MINOR version component is chosen.

   iii Otherwise, the module with the highest PATCH version component is chosen.

   iv Note, the _COMPAT modifier is ignored for comparison purposes.

2. If one module has a version statement that matches the YANG Semver format but the other does not, then the module with the version statement matching the YANG Semver format is chosen.

3. If neither module has a version statement that matches the YANG Semver format, then the module with the most recent revision-date is chosen.

4. If there is no difference in version/revision dates to distinguish between two module then:

   i Submodule information, if present, for both module definitions is assumed to be equivalent and hence either can be used.

   ii the location leaf-list for both modules definitions is combined into a single list of locations without duplicates.

5. YANG Package Usage

This section provides information and guidance on how YANG packages can be used.

YANG packages can be defined and used for different purposes:

* By standards development organizations and industry organizations - to specify common sets of YANG data models that can be used together to manage network devices, or even just particular functionality on network devices (e.g., L3VPN services). Since package definitions can be defined hierarchically, packages defining different functionalities can be combined together into larger package definitions that define more complex and complete behavior and YANG schema. These package definitions may be published by the organizations as package files.

* For devices:

  - to describe the YANG schema associated with the device or a datastore schema on the device. These package schema can be made available both from the device and also in offline package files.

  - to define different optional YANG schema that can be used by the device and where clients can select which YANG schema can be used via configuration.

  - to refine standards based and industry packages to accurately report how the device does not fully conform to the package schema definition.

* To manage and report the schema available at YANG schema mounts points.

It is RECOMMENDED that organizations publishing YANG modules also publish YANG package definition that group and version those modules into units of related functionality. This increases interoperability by encouraging different implementations to coalesce around 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.

Where possible, package definitions SHOULD be made available offline in Package Instance Data files, see Section 5.5, but also on the device as a list of known packages and relationships between YANG library datastore schema and equivalent YANG package definitions, e.g., see Section 5.4.

5.1. Achieving package name uniqueness

As per Section 3.1, YANG package names are globally unique, since two different package definitions with the same name, but different content, cannot both be used together within the same package definition.

There are a couple of ways of achieving this uniqueness requirement:

* For package definitions that define an public API, or that could apply to multiple servers exposing the same management API, then an organization prefix, and perhaps device family name, should be included in the package name, i.e., following a similar naming convention as for modules.

* For package definitions that are entirely local to a particular server or device, then the sysName of the device, a MAC address, or a UUID should be used as a suffix to the package name to ensure uniqueness.

5.2. Specifying modules as implemented or import-only

Some YANG modules do not define any implementable data nodes, RPCs, Actions, or Notifications. These YANG modules often may include 'types' in the name of the YANG Module. For YANG package definitions, there is a choice as to whether to include these types modules in the packages list of implemented modules, or as import-only modules. This document does not specify how these should be declared, but instead gives some points of consideration that may be helpful when choosing. These are:

Listing a types only module as implemented allows for simpler automatic module version selection between different packages, as per Section 4.1.

As per [RFC7950] section 9.10.2, identities are only available for use by the server for implemented modules.

If a module defines data nodes and types and the server only wants to use the types but not implement any data nodes from the module then listing the module as import-only is clearer and simpler than marking it as implemented with a separate deviation file that deviates all data nodes as not-supported.

If a module imports a module at an exact revision (which, as per [I-D.ietf-netmod-yang-module-versioning], is not recommended) then it may be helpful to list that module in the import-only module list (even if implemented) to ensure that the import dependency is always satisfied.

5.3. Referential completeness and YANG packages

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. An example is provided in Appendix A.5.

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.

5.4. Providing Package Definitions on a Server

5.4.1. Package List

A top level 'packages' container holds the list of all versions of all packages known to the server. Entries in this list do not necessarily mean that the package is implemented by the server or currently active for any datastore. Instead the YANG Library package bindings in Section 5.4.3 are used to indicate which of the advertised packages are supported by each datastore schema.

Each list entry in '/packages/package' SHOULD list one or more URLs pointing to an offline location where the package definition can be obtained as a YANG 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, or if package versions have been changed via applying different software packages or hot fixes.

5.4.2. 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:
        +--ro name                  pkg-name
        +--ro version               pkg-version
        ... remainder of yang-package-instance grouping ...
        +--ro location*             inet:uri
```

5.4.3. YANG Library Package Bindings

The ietf-yl-packages module augments YANG library to allow a server to indicate that a datastore schema is defined by a package, or a union of compatible packages. Since packages can 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 (if previously cached), or downloading and comparing the full list of modules and enabled features.

If multiple packages are listed for a datastore schema then they are resolved as if the packages were all directly included in a single package definition, following the standard package resolution rules in Section 4.

This means that conflicting module versions between the listed packages are resolved using the automatic module version resolution rules in Section 4.1. As a result, the version that wins those rules is selected in the resolved package schema. This allows an extra ’bugfix’ package to be added to the list of packages defining a schema to provide a higher/newer module version, but it cannot be used to select a lower/older module version. Instead, a wrapper package would need to be defined to explicitly select the older module version (see Section 7.1).

If populated, the set of packages listed for a datastore schema MUST resolve to a schema that exactly matches the schema defined by the YANG library 'schema/module-set' data node, and the resolved schema MUST be referentially complete. Using YANG packages offers an alternative hierarchical definition of the same schema.

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 supported-feature*   pkg-types:scoped-feature
```

5.5. Package Instance Data Files

YANG packages SHOULD be made available offline from the server, defined as YANG instance data files [RFC9195] using the schema below to define the package data.

Package instance data files use the ".ypkg" file extension and the "application/ypkg" media type as defined in Section 10.3.

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>.ypkg".

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 meta-data and the YANG package meta-data. 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 meta-data 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-instance" YANG module has the following structure:

```
module: ietf-yang-package-instance

  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 ...
```

5.6. YANG packages as schema for YANG instance data document

YANG package definitions can be used as the content schema definition for YANG instance data files. When using a package-based content schema, the name and version of the package MUST be specified, a package 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/yid:content-schema-spec:
    +--:(pkg-schema)
      +-- pkg-schema
        +-- name        pkg-name
        +-- version     pkg-version
        +-- location*   inet:uri
```

6. Package Evolution and Versioning

6.1. Package versioning

As defined in Section 3.1, YANG packages are versioning using [I-D.ietf-netmod-yang-semver]. This section describes how those rules apply to YANG package definitions.

6.1.1. Updating a package with a new version

Package compatibility is fundamentally defined by how the package 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. See section 4.5 of [I-D.ietf-netmod-yang-semver] for guidance on choosing the next version number based on the type of change being made.

It is important to note that a non-backwards-compatible (NBC) change to a package definition (generally requiring a major version number increment) may not always result in an NBC change to the resolved schema. For example, if a package replaces a module version 1.0.0 with version 3.0.0, but the content of version 3.0.0 has been reverted to be functionally identical to version 1.0.0 (effectively backing out previous changes between the two), the schema difference between the package versions may be functionally compatible. Nevertheless, such changes must still follow the versioning rules defined above based on the package definition changes, not the effective change before resolved schema.

6.1.1.1. Non-Backwards-compatible changes

Non-backwards-compatible changes to a package are those that may cause the resolved package schema to change in a way that can negatively impact clients. E.g., the removal or change of a data node that was present in the previous package version, which may be used by client applications.

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

* Changing an ’includes/package’ list entry to select a package version that is non-backwards-compatible to the prior package version, or removing a previously included package.

* Changing an ’includes/module’ or ’includes/import-only-module’ list entry to select a module version that is non-backwards-compatible to the prior module version, or removing a previously implemented module.

* Adding an entry to the ’excludes/module’ list or the ’excludes/import-only-module’ list, which in either case causes a module to be removed from an included package and could affect the conformance reporting of whether the included package is deemed as being implemented.

* Removing a feature from the ’includes/feature’ list unless the feature was not mandatory in any included packages.

* Adding a feature to the ’excludes/feature’ list unless the feature was not mandatory in any included package.

* Adding, changing, or removing a module containing one or more deviations, that when applied to the target module would create a change that is considered a non-backwards-compatible change to the affected data node in the schema associated with the prior package version.

* Removing a package from a mount point, or changing a mounted package to a version that is non-backwards-compatible to the prior package version.

6.1.1.2. Backwards-compatible changes

Backwards-compatible changes to a package are those that may cause the resolved package schema to change in a way that should not impact clients. E.g., the addition of a new data node that was not present in the previous package version.

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

* 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.

* 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.

* Removing an entry to the ’excludes/module’ list or the ’excludes/import-only-module’ list. [TODO - Do we need to consider deviations here, e.g., removing a deviation may change the schema.]

* Adding a feature to the ’mandatory-feature/include’ leaf-list.

* Removing a feature to the ’mandatory-feature/exclude’ leaf-list.

* Adding, changing, or removing a module containing one or more deviations, that when applied to the target module would create a change that is considered a backwards-compatible change to the affected data node in the schema associated with the prior package version.

6.1.1.3. Editorial changes

Editorial changes to a package are those that do not change the resolved package schema.

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

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

* 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.

* Adding a package or module to the package definition (including at a mount point), if that package or module version is already present from an included package.

* Updating the location information of any module or package from an included package.

* Any change to any meta-data associated with a package definition.

6.1.2. Guidelines for Package Versions During Package Development

During development of a new package, or while updating a previously released package, special care should be taken with the selection of the version associated with the package.

General guidance from chapter 6 of [I-D.ietf-netmod-yang-semver] may be helpful here. [TODO - Will we provide more specific guidance in a separate draft for IETF YANG Packages?]

7. Package Conformance

Better and easier conformance is a major design goal for YANG Packages. YANG package conformance is similiar to how YANG [RFC7950] requires that servers either implement a module faithfully, or otherwise use deviations to indicate areas of non-conformance. Ultimately, each version of a YANG package resolves, as per (Section 4), to a YANG schema that is defined as a set of implemented modules and import-only modules, deviations, features, and mounted schema. For YANG package conformance, it is necessary to determine whether an implementation faithfully conforms to the full YANG schema defined by a resolved YANG package.

The YANG packaging solution is designed to allow for conformance to be checked at a package level, potentially using cached offline package definitions, 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.

In the case where a device does not completely conform to an standard or industry defined YANG package definition, then there are a few suggestions on how this can be handled:

* Automatic module version resolution rules can be used to select the latest module version between packages.

* Device specific implementation packages can be defined that 'wrap' standard/industry packages to accurately report the device's schema.

7.1. Resolving conflicting module versions in included packages

Sometimes a package definition may include multiple packages that implement different versions of a module.

As per the package definition rules in Section 3.1, a package can only implement a single version of a module, and hence in cases of conflicting versions in included packages/modules it is necessary to resolve which version of the module is used. The default behaviour, defined in Section 4.1 is to use automatic resolution, which is generally the best choice. Manual resolution could be used to select a different module version instead, or even remove the module from the package entirely. However, care must be taken if an older module version is chosen, or a new non-backwards-compatible newer version is chosen, because, in both cases, this may affect conformance in one of the included packages.

7.2. Handling multiple included package versions

Unlike modules in a package definition, where there can only be a single version of a module in a resolved package definition, this does not apply to included packages. As per the package resolution rules in Section 4, when multiple included packages define different versions of the same package, then both versions are retained in the resolved package definition.

Instead, package and schema comparison tooling can be used to determine what level of package conformance has been achieved for each of the recursively included packages.

7.3. Exclusions and Deviations

Whenever possible, servers should aim to implement packages accurately since this maximizes interoperability for clients. However, if a server cannot faithfully implement a YANG package then it can define a new package to accurately report what it does implement. The RECOMMENDED mechanism for this is to define and advertise a separate "server implementation" package which includes the package to be conformed to, and then excludes modules or selects different versions, adds deviation modules, and excludes mandatory-features to indicate the actual conformance of the server implementation.

TODO, please see Appendix X for an example.

If an implementation doesn’t support any functionality in a module then it should exclude the module rather than using deviations to exclude all data nodes added by the module to the resolved schema. This gives a clearer indication to users of the package definition as to the intent. However, be aware when combining two included packages, that a module removed by one package could still be readded by another included package. If deviations are used this won’t happen unless the module defining the deviations is explicitly removed.

If an alternative version of a module is used then it is RECOMMENDED to use a newer module version, if possible, rather than an older version. Selecting a backwards-compatible version is also helpful because it maximizes the chance that clients will be able to easily interoperate with the server.

7.4. Use of features in YANG modules and YANG packages

The YANG language, [RFC7950] section 5.6.2, supports feature statements as the mechanism to make parts of a schema optional. Published standard YANG modules 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 include the 'features' list, which allow the package to define a set of 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.

TODO, see issue https://github.com/netmod-wg/yang-ver-dt/issues/273.

7.5. Use of YANG semantic versioning for YANG packages

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 'example-foo' package at version 1.0.0 should interoperate with a server implementing 'example-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.

7.6. Restrictions on the 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.

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.

7.7. The relationship between packages and datastores

As defined by NMDA [RFC8342], each datastore has an associated datastore schema. These datastore schema can be advertised by servers using YANG Library [RFC8525], augmented with the associated YANG package information, as per Section 5.4.3. Sections 5.1 and 5.3 of NMDA defines further constraints on the schema associated with datastores. These constraints can be summarized thus:

* The schema for all conventional datastores is the same.

* 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.

* 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 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):

* 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.

* 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.

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

* The packages for the operational datastore schema SHOULD 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 packages (which may be packages attached to the datastore, or may be packages included by other packages).

* 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 that relies on the automatic resolution of module versions.

* The resolved schema representing a datastore MUST be referentially complete.

7.8. Using package comparison tools

Clients fetch the package information from the server (if required), and then can use tools to generate the resolved package schema. The resolved package schema may list multiple versions of the same package (if included with different versions), and it may list package versions that are not completely implemented by the device. By using package schema comparison, as described below, tooling can report on the level of conformance for each package and included package version advertised by the device.

YANG package schema comparison tools (and also documentation) can be used to determine how closely a device implements particular YANG package definitions advertised by the device. The tooling, by resolving the package definition, then comparing the set of module versions, features, deviations and mounts, can determine if the package schema is implemented exactly, or if the package schema is backwards-compatible, or non-backwards-compatible. Tooling can determine if modules have been removed, mounts have been changed, or deviations have been applied.

8. YANG Modules

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

8.1. ietf-yang-package-types

```
<CODE BEGINS> file "ietf-yang-package-types#0.8.0.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-semver  {
    prefix ys;
    reference "XXXX: YANG Semantic Versioning";
  }

  import ietf-yang-types {
```

```
  prefix yang;
  rev:recommended-min-date 2019-07-21;
  reference "RFC 6991bis: Common YANG Data Types.";
}

import ietf-inet-types {
  prefix inet;
  rev:recommended-min-date 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 provides type and grouping definitions for YANG
   packages.

   Copyright (c) 2026 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 Revised BSD License set
   forth in Section 4.c of the IETF Trust's Legal Provisions
   Relating to IETF Documents
   (https://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 2026-01-30 {
       ys:version 0.8.0;
       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 ys:version;
       description
         "Packages are versioning using YANG Semver version labels.";
     }

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

     typedef version-or-rev-date {
       type union {
         type rev:revision-date;
         type ys:version;
       }
       description
         "Identifies a module by YANG semantic version or revision
          date";
     }

     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 mount-ypath {
        type string;

        description
          "A path that identifies a set of data nodes in the schema tree.

          This leaf is encoded as a JSON style encoded
          instance-identifier (regardless of whether the format
          used to encode the YANG instance data), as specified in
          RFC 7951, section 6.11, except that keys are optional.

          For optional keys, the name and value of the key is
          excluded from the key list.

          TODO - Check if this definition is sufficient.";
      }

      /*
       * 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-exclusions {
       description
         "Parameters for excluding modules and packages from a YANG
          package definition";

       container excludes {
         description
           "Contains parameters for excluding modules and packages
            from a YANG package definition";

         leaf-list module {
           type module-name;
           description
             "Lists implemented modules, of any version, that may have
              have been brought in by included packages, but are
              explicitly excluded from this package definition.

              Excluding a module can affect the compliance and
              correctness of any included packages that expect that
              module to be implemented.

              Excluding a module also implicitly excludes any submodules
              and mandatory-features defined in the excluded module.

              It is an error to list a module in both this list and the
              'includes/module' list.";
         }

         list import-only-module {
           key "name";
           description
             "Lists import-only modules that may have have been brought
              in by included packages, but are explicitly excluded from
              this package definition.

              It is an error to list a module in both this list and the
              'includes/import-only-module' list.";

           leaf name {
             type module-name;
             mandatory true;
             description
               "The name of the import-only module to exclude some
                versions of.";
           }

           leaf-list version {
             type version-or-rev-date;
```

```
            description
              "Lists specific versions of the import-only module being
                excluded.  If no versions are listed, all versions of
                the import-only module are excluded.

                If required, the YANG Semantic Version SHOULD be used to
                identify the module version, otherwise the YANG module
                revision date is used.";
          }
        }

        leaf-list feature {
          type scoped-feature;
          description
            "Lists features from the mandatory-features exported by an
            included package that are reclassified as being OPTIONAL
            to support by any server implementing the package,
            overriding the behavior specified by the included package.

            Features MUST NOT be specified both in this list and also
            the 'includes/feature' list.

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

    grouping yang-pkg-location {
      description
        "Parameters for locating a YANG package instance";

      leaf-list location {
        type inet:uri;
        description
          "Contains a URL that represents where an instance data file
           (RFC 9195) 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.";
      }
    }

    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.";
      }

      container includes {
        description
          "Lists package and modules that are included in the package
           definition.";

        list package {
```

```
      key "name";
      description
        "An entry in this list represents a package that is
         included as part of the package definition, or to change
         the version of a descendent included package.

         An entry in this list overrides any other package version
         'included' by an included package, which can be used for
         resolving conflicting package versions from included
         packages.

         A package definition MUST resolve to including only a
         single version of any YANG package.";

      uses yang-pkg-identification-leafs;
      uses yang-pkg-location;
    }

    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.

        A entry in this list overrides any module version
        'implemented' by an included package.";
      reference
        "RFC 7950: The YANG 1.1 Data Modeling Language.";

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

      leaf version {
        type version-or-rev-date;
        mandatory true;
        description
          "Identifies the module version.  If available, the YANG
           Semantic Version SHOULD be used, otherwise the YANG
           module revision date is used.";
      }

      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.";
          }

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

            leaf name {
              type module-name;
              mandatory true;
              description
                "The YANG submodule name.";
            }

            leaf version {
              type version-or-rev-date;
              mandatory true;
              description
                "The YANG submodule revision date or YANG Semantic
                 version.

                 If the parent module include statement for this
                 submodule includes a revision date then it MUST match
                 the revision date specified here or it MUST match the
                 revision-date associated with the version specified
                 here.";
            }

            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.";
            }
          }
        }
```

```
        list import-only-module {
          key "name version";
          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 module-name;
            mandatory true;
            description
              "The YANG module name.";
          }

          leaf version {
            type version-or-rev-date;
            mandatory true;
            description
              "Identifies the module version.  If available, the YANG
               Semantic Version SHOULD be used, otherwise the YANG
               module revision date is used.";
          }

          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.";
          }

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

            leaf name {
              type module-name;
              mandatory true;
```

```
            description
              "The YANG submodule name.";
          }

          leaf version {
            type version-or-rev-date;
            mandatory true;
            description
              "The YANG submodule revision date or YANG Semantic
               version.

               If the parent module include statement for this
               submodule includes a revision date then it MUST match
               the revision date specified here or it MUST match the
               revision-date associated with the version specified
               here.";
          }

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

           Mandatory-features specified by any directly included
           packages MUST also be supported by server
           implementations, unless excluded by an entry in the
           'excludes/feature' list, 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>.";
        }
      }

      uses yang-pkg-exclusions;

      list mount {
        key "mount-path";
        description
          "An entry in this list represents additions to, or a
           replacement of, the schema found at the specified mount
           point.

           The full schema at the mount point depends on the
           setting of the inherit-packages leaf:

           If set to true (the default) - the schema is defined
           as the union of the resolved package schema at the mount
           point by any packages in the 'includes/package' list
           combined with the resolve package schema of all packages
           listed in the 'packages' list.

           If set to false - the schema is defined as the union of
           only the resolved package schema of all packages
           listed in the 'packages' list.";

        leaf "mount-path" {
          type mount-ypath;
          mandatory true;
          description
            "This path identifies a mount point in the schema.

             This leaf is encoded as a JSON style encoded
             instance-identifier (regardless of whether the format
             used to encode the YANG instance data), as specified in
             RFC 7951, section 6.11, except that keys are optional.

             For optional keys, the name and value of the key is
             excluded from the key list.

             Mount paths MUST only be used for schema mount points
             defined in the package schema.

             For example, if an example module 'ex-module' defines a
             mount point under list entry'/modules/module/' then a
             mount path of

             - '/modules/module[name=foo]' would indicate the mounted
```

```
            package schema for only the 'foo' entry in the module
            list.  Each entry in the list could have a different
            mounted schema specified.

          - '/modules/module[]' would indicate that the same
            mounted package schema is available for all list
            entries in the module list.";
        }

        leaf inherit-packages {
          type boolean;
          default true;
          description
            "Indicates whether the packages and parent-references
             available at the mount point for this package definition
             automatically include all packages and parent-references
             mounted at the same mount path in any
             'includes/packages' entries.

             If set to false, then only packages and
             parent-references listed here in the 'packages' list are
             included.  This allows the definitions in the mounted
             packages to be modified (e.g., remove or change module
             versions).  To help conformance, the packages listed
             here SHOULD include all the packages that would be have
             been automatically included.
            ";
        }

        list package {
          key "name";
          description
            "The packages that will be mounted at the specified mount
             path either in addition to, or instead of, the mounted
             packages from the 'included/package' list.

             Also see the 'inherit-packages' leaf.";
          uses yang-pkg-identification-leafs;
          uses yang-pkg-location;
        }

        leaf-list parent-reference {
          type mount-ypath;
          description
            "Represents paths in the parent schema that are accessible
             from the mounted schema for the evaluation of XPath
             expressions.
```

```
              See Mount Point path and parent-reference in Schema Mount
              (RFC 8528) for a more detailed description.

              Unlike the YANG module defined in RFC 8528, this leaf is
              encoded as a JSON style encoded instance-identifier
              (regardless of whether the format used to encode the YANG
              instance data), as specified in RFC 7951, section 6.11,
              except that keys are optional.

              For optional keys, the name and value of the key is
              excluded from the key list.";
        }
      }
    }
  }
  <CODE ENDS>
```

8.2. ietf-yang-package-instance

```
  <CODE BEGINS> file "ietf-yang-package-instance#0.8.0.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-semver  {
      prefix ys;
      reference "XXXX: YANG Semantic Versioning";
    }

    import ietf-yang-package-types {
      prefix pkg-types;
      ys:recommended-min-version 0.8.0;
      reference "RFC XXX: this RFC.";
    }

    import ietf-yang-structure-ext {
      prefix sx;
      reference "RFC 8791: 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 content schema for an YANG instance data document specifying
        a YANG package.

        Copyright (c) 2026 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 Revised BSD License set
        forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (https://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 2026-01-30 {
       ys:version 0.8.0;
       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>
```

8.3. ietf-yang-package

```
  <CODE BEGINS> file "ietf-yang-packages#0.8.0.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-semver  {
      prefix ys;
      reference "XXXX: YANG Semantic Versioning";
    }

    import ietf-yang-package-types {
      prefix pkg-types;
      ys:recommended-min-version 0.8.0;
      reference "RFC XXX: this RFC.";
    }

    import ietf-inet-types {
      prefix inet;
      rev:recommended-min-date 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) 2026 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 Revised BSD License set
        forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (https://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 2026-01-30 {
       ys:version 0.8.0;
       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.";
      }

    }

    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";
         }

       }
     }
   }
   <CODE ENDS>
```

8.4. ietf-yl-packages

```
   <CODE BEGINS> file "ietf-yl-packages#0.8.0.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-semver  {
       prefix ys;
       reference "XXXX: YANG Semantic Versioning";
     }

     import ietf-yang-package-types {
```

```
      prefix pkg-types;
      ys:recommended-min-version 0.8.0;
      reference "RFC XXX: YANG Packages.";
    }

    import ietf-yang-packages {
      prefix pkgs;
      ys:recommended-min-version 0.8.0;
      reference "RFC XXX: YANG Packages.";
    }

    import ietf-yang-library {
      prefix yanglib;
      rev:recommended-min-date 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) 2026 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 Revised BSD License set
       forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (https://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 2026-01-30 {
      ys:version 0.8.0;
      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.

         Additional features may be supported by the device listed
         in the 'implemented-feature' list.
         ";

      uses pkgs:yang-ds-pkg-ref;

      leaf-list supported-feature {
        type pkg-types:scoped-feature;
        description
          "The name of each YANG feature supported by the
           server MUST be identified.  Features are
           identified using <module-name>:<feature-name>.

           This list MUST be equivalent to the list of
           supported features advertised in YANG library.";
      }
    }
  }
  <CODE ENDS>
```

8.5. ietf-yang-inst-data-pkg

```
   <CODE BEGINS> file "ietf-yang-inst-data-pkg#0.8.0.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-semver  {
       prefix ys;
       reference "XXXX: YANG Semantic Versioning";
     }

     import ietf-yang-package-types {
       prefix pkg-types;
       ys:recommended-min-version 0.8.0;
       reference "RFC XXX: this RFC.";
     }

     import ietf-yang-structure-ext {
       prefix sx;
       reference "RFC 8791: YANG Data Structure Extensions.";
     }

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

     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 content schema in YANG instance data
        documents.

        Copyright (c) 2026 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 Revised BSD License set
```

```
      forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (https://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 2026-01-30 {
    ys:version 0.8.0;
    description
      "Initial revision";
    reference
      "RFC XXXX: YANG Packages";
  }

  /*
   * Augmentations
   */

  sx:augment-structure
    "/yid:instance-data-set/yid:content-schema/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-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>
```

9. Security Considerations

This document defines YANG modules for defining YANG schema, that are often used to configure and monitor network devices.

The document defines three YANG modules that are accessible from devices, and the normal YANG network management security considerations apply, which are further described below, in Section 9.1.

YANG packages also offers an alternative mechanism to YANG Library [RFC8525] for reporting YANG schema, and hence the security considerations from that document also apply to the use of YANG packages. As per the YANG library security considerations, the module 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.

The ’ietf-yang-package-instance.yang’ YANG file allows YANG packages to be defined in YANG instance data files. In addition, "ietf-yang-inst-data-pkg" allows YANG packages to used to define the schema for YANG instance data files. In both cases, the security considerations from [RFC9195] apply. Since YANG package instance data files are outside the security controls of the network management protocols, it is important to consider controlling access to these files to restrict access to potentially sensitive information.

9.1. YANG Module Security Considerations

This section is modeled after the template described in Section 3.7 of [I-D.ietf-netmod-rfc8407bis].

The "ietf-yang-package-types", "ietf-yang-packages" and "ietf-yl-packages" YANG modules define data models that are designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. These YANG-based management protocols (1) have to use a secure transport layer (e.g., SSH [RFC6242], TLS [RFC8446], and QUIC [RFC9000]) and (2) have to use mutual authentication.

The Network Configuration Access Control Model (NACM) [RFC8341] 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.

Some of the readable data nodes in this YANG module 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. Specifically, the following subtrees and data nodes have particular sensitivities/ vulnerabilities:

* There are no particularly sensitive readable data nodes.

Modules that use the groupings that are defined in this document should identify the corresponding security considerations. For example, reusing some of these groupings will expose privacy-related information (e.g., 'yang-pkg-instance').

10. IANA Considerations

10.1. IANA YANG Module and Namespace Registrations

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
```

10.2. IETF YANG Package Registry

This document requests that IANA create a registry for IETF YANG packages, that lists all versions of all YANG package definitions published by the IETF, and provides a reliable location to store the YANG package definitions.

The name of the registry is "YANG Package Names".

The registry shall record for each entry:

* the name of the YANG package

* a brief description of the package purpose

* The latest published package version

* A list of all published package versions, each hyperlinked to the location where that package definition may be retrieved from.

* a reference to the package documentation (e.g., the RFC number)

There are no initial assignments.

For allocation, the registration policy is Specification Required, as defined in [RFC8126]. All registered YANG package names MUST comply with the rules for identifiers stated in Section 6.2, and MUST have a package name prefix.

The package name prefix 'ietf-' is reserved for package managed and published under the IETF stream [RFC4844], while the package name prefix 'irtf-' is reserved for IRTF stream documents. Packages published in other RFC streams MUST have a similar suitable prefix.

All package names in the registry MUST be unique.

10.3. Media Type Registration for application/ypkg

This document requests IANA to register the following media type, following the procedures of [RFC6838]:

```
Type name: application
Subtype name: ypkg
Required parameters: N/A
Optional parameters: N/A
Encoding considerations: 8bit; YANG package instance data files
are encoded in UTF-8.
Security considerations: See the Security Considerations section
of RFC XXXX.
Interoperability considerations: N/A
Published specification: RFC XXXX
Applications that use this media type: YANG tooling and servers
that generate or consume YANG package instance data files.
Additional information:
   Magic number(s): None
   File extension(s): ypkg
   Macintosh file type code(s): 'Text'
   Fragment identifiers: N/A
```

Person & Email address to contact for further information: NETMOD WG (mailto:netmod@ietf.org)
Intended usage: COMMON
Restrictions on usage: N/A
Author: IETF
Change controller: IESG

11. Acknowledgements

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

Bo Wu acted as a temporary editor for earlier versions of this work.

12. References

12.1. Normative References

[I-D.ietf-netmod-rfc8407bis]
Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-28, 5 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28>.

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bj旦rklund, M., and K. Watsen, "YANG Data Structure Extensions", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-data-ext-05, 9 December 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-data-ext-05>.

[I-D.ietf-netmod-yang-module-versioning]
Wilton, R., Rahman, R., Lengyel, B., Clarke, J., and J. Sterne, "Updated YANG Module Revision Handling", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-15, 18 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-versioning-15>.

[I-D.ietf-netmod-yang-semver]
Clarke, J., Wilton, R., Rahman, R., Lengyel, B., Sterne, J., and B. Claise, "YANG Semantic Versioning", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-semver-24, 29 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-semver-24>.

[I-D.ietf-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-solutions-01, 2 November 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-solutions-01>.

[I-D.ietf-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Schema Selection", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-ver-selection-00, 17 March 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-ver-selection-00>.

[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>.

[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>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC8528] Bjorklund, M. and L. Lhotka, "YANG Schema Mount", RFC 8528, DOI 10.17487/RFC8528, March 2019, <https://www.rfc-editor.org/info/rfc8528>.

[RFC8791] Bierman, A., Bj旦rklund, M., and K. Watsen, "YANG Data Structure Extensions", RFC 8791, DOI 10.17487/RFC8791, June 2020, <https://www.rfc-editor.org/info/rfc8791>.

[RFC9195] Lengyel, B. and B. Claise, "A File Format for YANG Instance Data", RFC 9195, DOI 10.17487/RFC9195, February 2022, <https://www.rfc-editor.org/info/rfc9195>.

12.2. Informative References

[I-D.bierman-netmod-yang-package]
Bierman, A., "The YANG Package Statement", Work in Progress, Internet-Draft, draft-bierman-netmod-yang-package-00, 6 July 2015, <https://datatracker.ietf.org/doc/html/draft-bierman-netmod-yang-package-00>.

[I-D.ietf-netmod-artwork-folding]
Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", Work in Progress, Internet-Draft, draft-ietf-netmod-artwork-folding-12, 20 January 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-artwork-folding-12>.

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

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[RFC4844] Daigle, L., Ed. and IAB, "The RFC Series and RFC Editor", RFC 4844, DOI 10.17487/RFC4844, July 2007, <https://www.rfc-editor.org/info/rfc4844>.

[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>.

[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>.

[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>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <https://www.rfc-editor.org/info/rfc8529>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

Appendix A. Package Examples

This section provides various examples of YANG packages, and as such this text is non-normative. The purpose of the examples is to illustrate the file format of YANG packages, how package dependencies, exclusions, and package mounts 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.

A.1. Basic Package Examples

This section provides some simple examples of YANG package defines, illustrated using the instance data file format defined in Section 5.5.

TODO For brevity, most examples excluding the module and package locations. TODO - Probably should point to IANA instead. Some examples use a shortened URL of "tiny.cc/ietf-yang" as a replacement for "https://raw.githubusercontent.com/YangModels/yang/master/standard/ietf/RFC"

TODO, use an IANA reference here instead?

TODO, line wrapping in examples.

A.1.1. Example IETF Basic Types YANG package, version 1.0.0

A very simple package with no dependencies on other packages that illustrates how a basic types package might be defined. In this case, the module dependencies have been declared as import-only but they could also have been declared as implemented modules.

```
<CODE BEGINS> file "example-base-types-pkg%1.0.0.ypkg"
========= NOTE: '\' line wrapping per BCP XX (RFC XXXX) ===========
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-base-types-pkg",
    "description": "YANG Package definition",
    "content-data": {
      "ietf-yang-package-instance:package": {
        "name": "example-base-types-pkg",
        "version": "1.0.0",
        "timestamp": "2025-05-27T12:26:09.956687Z",
        "description": "Example package containing base IETF and IANA type modules",
        "reference": "RFC 6021, RFC 6536",
        "complete": false,
        "includes": {
          "import-only-module": [
            {
              "name": "ietf-yang-types",
              "version": "2010-09-24"
            },
            {
              "name": "ietf-inet-types",
              "version": "2010-09-24"
            },
            {
              "name": "ietf-netconf-acm",
              "version": "2012-02-22"
            }
          ]
        }
      }
    }
  }
}
<CODE ENDS>
```

A.1.2. Example IETF Basic Types YANG package, version 1.1.0

An updated version of the basic types package that includes Updated versions of the types modules and follows the YANG semver versioning rules.

```
<CODE BEGINS> file "example-base-types-pkg%1.1.0.ypkg"
========= NOTE: '\' line wrapping per BCP XX (RFC XXXX) ===========
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-base-types-pkg",
    "description": "YANG Package definition",
    "content-data": {
      "ietf-yang-package-instance:package": {
        "name": "example-base-types-pkg",
        "version": "1.1.0",
        "timestamp": "2025-05-27T12:29:44.621705Z",
        "description": "Example package containing base IETF and IANA type modules",
        "reference": "RFC 6991, RFC 8341",
        "complete": false,
        "includes": {
          "import-only-module": [
            {
              "name": "ietf-yang-types",
              "version": "2013-07-15"
            },
            {
              "name": "ietf-inet-types",
              "version": "2013-07-15"
            },
            {
              "name": "ietf-netconf-acm",
              "version": "2018-02-14"
            }
          ]
        }
      }
    }
  }
}
<CODE ENDS>
```

A.1.3. 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, the modules are referenced by revision date rather than revision number.

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

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ietf-network-device-pkg",
    "content-schema": {
      "pkg-schema": {
         "name": "ietf-yang-package-defn-pkg",
         "version": "0.1.0"
      }
    },
    "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.ypkg" ],
        "module": [
          {
            "name": "iana-crypt-hash",
            "revision": "2014-08-06",
            "location": [ "https://tiny.cc/ietf-yang/\
                                 iana-crypt-hash%402014-08-06.yang" ],
          },
          {
            "name": "ietf-system",
            "revision": "2014-08-06",
            "location": [ "https://tiny.cc/ietf-yang/\
                                    ietf-system%402014-08-06.yang" ],
          },
```

```
            {
              "name": "ietf-interfaces",
              "revision": "2018-02-20",
              "location": [ "https://tiny.cc/ietf-yang/\
                                  ietf-interfaces%402018-02-20.yang" ],
            },
            {
              "name": "ietf-netconf-acm",
              "revision": "2018-02-14",
              "location": [ "https://tiny.cc/ietf-yang/\
                                 ietf-netconf-acm%402018-02-14.yang" ],
            },
            {
              "name": "ietf-key-chain",
              "revision": "2017-06-15",
              "location": [ "https://tiny.cc/ietf-yang/\
                                    ietf-key-chain@2017-06-15.yang" ],
            },
            {
              "name": "ietf-ip",
              "revision": "2018-02-22",
              "location": [ "https://tiny.cc/ietf-yang/\
                                         ietf-ip%402018-02-22.yang" ],
          }
          ],
          "import-only-module": [
            {
              "name": "ietf-yang-types",
              "revision": "2013-07-15",
              "location": [ "https://tiny.cc/ietf-yang/\
                                  ietf-yang-types%402013-07-15.yang" ],
            },
            {
              "name": "ietf-inet-types",
              "revision": "2013-07-15",
              "location": [ "https://tiny.cc/ietf-yang/\
                                  ietf-inet-types%402013-07-15.yang" ],
            }
          ]
        }
      }
    }
  }
  <CODE ENDS>
```

A.1.4. 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 revision 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.ypkg"
========== NOTE: '\' line wrapping per BCP XX (RFC XXXX) ==========

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ietf-routing-pkg",
    "content-schema": {
      "pkg-schema": {
         "name": "ietf-yang-package-defn-pkg",
         "version": "0.1.0"
      }
    },
    "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.ypkg" ],
          }
```

```
          ],
          "module": [
            {
              "name": "ietf-routing",
              "revision": "2018-03-13",
              "location": [ "https://tiny.cc/ietf-yang/\
                                    ietf-routing@2018-03-13.yang" ],
            },
            {
              "name": "ietf-ipv4-unicast-routing",
              "revision": "2018-03-13",
              "location": [ "https://tiny.cc/ietf-yang/\
                          ietf-ipv4-unicast-routing@2018-03-13.yang" ],
            },
            {
              "name": "ietf-ipv6-unicast-routing",
              "revision": "2018-03-13",
              "location": [ "https://tiny.cc/ietf-yang/\
                          ietf-ipv6-unicast-routing@2018-03-13.yang" ],
            },
            {
              "name": "ietf-isis",
              "revision": "2018-12-11",
              "location": [ "https://tiny.cc/ietf-yang/\
                          " ],
            },
            {
              "name": "ietf-interfaces-common",
              "revision": "2018-07-02",
              "location": [ "https://tiny.cc/ietf-yang/\
                          " ],
            },
            {
              "name": "ietf-if-l3-vlan",
              "revision": "2017-10-30",
              "location": [ "https://tiny.cc/ietf-yang/\
                          " ],
            },
            {
              "name": "ietf-routing-policy",
              "revision": "2018-10-19",
              "location": [ "https://tiny.cc/ietf-yang/\
                          " ],
            },
            {
              "name": "ietf-bgp",
              "revision": "2018-05-09",
              "location": [ "https://tiny.cc/ietf-yang/\
```

```
                            " ],
              },
              {
                "name": "ietf-access-control-list",
                "revision": "2018-11-06",
                "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              }
            ],
            "import-only-module": [
              {
                "name": "ietf-routing-types",
                "revision": "2017-12-04",
                "location": [ "https://tiny.cc/ietf-yang/\
                            ietf-routing-types@2017-12-04.yang" ],
              },
              {
                "name": "iana-routing-types",
                "revision": "2017-12-04",
                "location": [ "https://tiny.cc/ietf-yang/\
                            iana-routing-types@2017-12-04.yang" ],
              },
              {
                "name": "ietf-bgp-types",
                "revision": "2018-05-09",
                "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              },
              {
                "name": "ietf-packet-fields",
                "revision": "2018-11-06",
                "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              },
              {
                "name": "ietf-ethertypes",
                "revision": "2018-11-06",
                "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              }
            ]
          }
        }
      }
    }
    <CODE ENDS>
```

A.2. Package Versioning Examples

This section provides examples of versioning packges.

The examples are non-normative, and for brevity, some expected information (e.g., locations) are omitted.

TODO, line wrapping in examples.

TODO, should we also remove the instance-data wrapper around the examples in this section, that wwould potentially help minimize wrapping?

A.3. Package Resolution Examples

This section provides examples of package resolution.

The examples are non-normative, and for brevity, some expected information (e.g., locations) are omitted.

TODO, line wrapping in examples.

TODO, should we also remove the instance-data wrapper around the examples in this section, that wwould potentially help minimize wrapping?

A.3.1. Example of package import conflict resolution

This section provides an example of how a package can resolve conflicting module revisions 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 revisions 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 revisions 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",
    "content-schema": {
      "pkg-schema": {
         "name": "ietf-yang-package-defn-pkg",
         "version": "0.1.0"
```

```
        }
      },
      "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",
              "revision": "1.0.0"
            },
            {
              "name": "example-module-B",
              "revision": "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",
      "content-schema": {
        "pkg-schema": {
           "name": "ietf-yang-package-defn-pkg",
           "version": "0.1.0"
        }
      },
      "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",
              "revision": "1.2.3"
            },
            {
              "name": "example-module-E",
              "revision": "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",
      "content-schema": {
        "pkg-schema": {
           "name": "ietf-yang-package-defn-pkg",
           "version": "0.1.0"
        }
      },
      "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",
              "revision": "1.2.3"
            }
          ],
          "import-only-module": [
            {
              "name": "example-types-module-C",
              "revision": "2018-11-26",
              "replaces-revision": [ "2018-01-01 "]
            }
          ]
        }
      }
    }
  }
```

A.4. Package Exclusion Examples

This section provides examples of how to exclude modules, and packages in a package definition, and to remove mandatory-features. The examples are non-normative, and for brevity, some expected information (e.g., locations) are omitted.

TODO, line wrapping in examples.

TODO, should we also remove the instance-data wrapper around the examples in this section, that wwould potentially help minimize wrapping?

A.4.1. Example of excluding modules and a mandatory feature

The following example defines two YANG packages.

The first package, "example-ab-pkg", implements two example modules, "example-module-a" and "example-module-b", two related types modules, and declares two mandatory-features.

The second package, "example-c-pkg", imports the first package, but excludes the implemented "example-module-a" module, the import-only "example-module-b-types" module, and the mandatory feature "bar" from the "example-module-b" module.

TODO - Should that feature have been implicitly removed?

The third figure shows the resulting schema in YANG Libary format, but with namespaces and locations elided for brevity.

<CODE BEGINS> file "example-ab-pkg%0.1.0.ypkg"

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ab-pkg",
    "description": "YANG Package definition",
    "content-data": {
      "ietf-yang-package-instance:package": {
        "name": "example-ab-pkg",
        "version": "0.1.0",
        "timestamp": "2025-05-29T05:55:55.774729Z",
        "description": "Example package defining modules A, B and associated types",
        "complete": false,
        "includes": {
          "module": [
            {
              "name": "example-module-a",
              "version": "1.0.0"
            },
            {
              "name": "example-module-b",
              "version": "1.1.0"
            }
          ],
          "import-only-module": [
            {
              "name": "example-module-a-types",
              "version": "1.0.0"
            },
            {
              "name": "example-module-b-types",
              "version": "1.1.0"
            }
          ],
          "features": [
            "example-module-a:foo",
            "example-module-b:bar"
          ]
        }
      }
    }
  }
}
```

<CODE ENDS>

The "example-c-pkg" Yang Package example illustrates exclusions of modules, import-only-modules and features.

<CODE BEGINS> file "example-c-pkg%0.1.0.ypkg"

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-c-pkg",
    "description": "YANG Package definition",
    "content-data": {
      "ietf-yang-package-instance:package": {
        "name": "example-c-pkg",
        "version": "0.1.0",
        "timestamp": "2025-05-29T05:55:55.817530Z",
        "description": "Example package importing A, removing B and adding C",
        "complete": false,
        "includes": {
          "package": [
            {
              "name": "example-ab-pkg",
              "version": "0.1.0",
              "location": [
                "file:///Users/rwilton/git/netmod-wg/yang-ver-dt/yang-packages/out/Exa
mpleAbPackage/makePackage.dest/yang-pkg.ypkg"
              ]
            }
          ],
          "module": [
            {
              "name": "example-module-c",
              "version": "2.0.0"
            }
          ]
        },
        "excludes": {
          "module": [
            "example-module-b"
          ],
          "import-only-module": [
            "example-module-b-types"
          ],
          "features": [
            "example-module-b:bar"
          ]
        }
      }
    }
  }
}
```

<CODE ENDS>

The following JSON illustrates what a resulting YANG library file would look like once all dependencies in the "example-c-pkg" YANG package have been resolved.

<CODE BEGINS> file "example-c-library%0.1.0.ypkg"

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "Package example-c-pkg@0.1.0 schema",
    "description": "YANG Package definition",
    "content-data": {
      "ietf-yang-library:library": {
        "module-set": [
          {
            "name": "Package example-c-pkg@0.1.0",
            "module": [
              {
                "name": "example-module-c",
                "revision": "2.0.0",
                "ietf-yang-library-semver:version": "2.0.0"
              },
              {
                "name": "example-module-a",
                "revision": "1.0.0",
                "ietf-yang-library-semver:version": "1.0.0",
                "feature": [
                  "foo"
                ]
              }
            ],
            "import-only-module": [
              {
                "name": "example-module-a-types",
                "revision": "1.0.0",
                "ietf-yang-library-semver:version": "1.0.0"
              }
            ]
          }
        ],
        "schema": [
          {
            "name": "Package example-c-pkg@0.1.0 schema",
            "module-set": [
              "Package example-c-pkg@0.1.0"
            ]
          }
        ],
        "content-id": "c826ea09"
      }
```

```
      }
    }
  }
  <CODE ENDS>
```

A.5. Hotfix Package Example

TODO, provide an example of how a bugfix package on a device that works. The example should illustrate how an incomplete package contains an updated version of some particular module due to a bugfix, but it does not the dependencies, and hence it is marked as an incomplete package. Both the baseline server package and the bugfix package are advertised by the server for the datastore schema and we rely on automatic module resolution to pickup the bugfix module version. Also note that it is not possible to downrev a module using this hotfix mechanism.

A.6. Mounted Package Examples

This section provides examples of XXX when using mounted packages. The examples are non-normative, and for brevity, some expected information (e.g., locations) are omitted.

TODO, line wrapping in examples.

TODO, should we also remove the instance-data wrapper around the examples in this section, that wwould potentially help minimize wrapping?

A.6.1. Example of package with a mounted package

This example illustrates a YANG package representing a network device that mounts a routing package at the network-instance mount point.

TODO - Can we use example that is the same, or very similar, to the network-instances example?

TODO - Indicate that this is a minimal example to illustrate a concept, and leaves out some optional elements that would be expected in a full implementation.

A.6.2. Example of an package replacing the mounted schema

This example illustrates an implementation of the mounted package defined in XXX, but that modifies the the mounted package.

Although this example illustrates applying deviations to the schema of a mounted package, the same mechanism can be used to change the implemented package version, remove a mounted package in its entirety, remove modules from a mounted package, change mandatory-features, change the behaviour of recursive mounts, etc.

TODO - Probably need more than one example.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems, Inc.
Email: rwilton@cisco.com

Reshad Rahman
Equinix
Email: reshad@yahoo.com

Joe Clarke
Cisco Systems, Inc.
Email: jclarke@cisco.com

Jason Sterne
Nokia
Email: jason.sterne@nokia.com