

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman
Equinix
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia
7 July 2025

YANG Packages
draft-ietf-netmod-yang-packages-06

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 8 January 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Terminology and Conventions	3
2. Introduction	5
2.1. Main Objectives	5
2.2. Potential Alternative Mechanisms	6
2.3. Open Questions/Issues	7
3. The YANG Package Definition	7
3.1. Package definition rules	10
3.2. Schema referential completeness	11
3.3. Submodules packages considerations	11
3.4. Package Mount Points	11
3.4.1. Exclusions	12
4. Package Resolution	12
4.1. Automatic Module Version Resolution	14
5. YANG Package Usage	14
5.1. Achieving package name uniqueness	15
5.2. Referential completeness and YANG packages	15
5.3. Providing Package Definitions on a Server	15
5.3.1. Package List	16
5.3.2. Tree diagram	16
5.3.3. YANG Library Package Bindings	16
5.4. Package Instance Data Files	17
5.5. YANG packages as schema for YANG instance data document	18
6. Package Evolution and Versioning	18
6.1. Package versioning	18
6.1.1. Updating a package with a new version	19
6.1.1.1. Non-Backwards-compatible changes	19
6.1.1.2. Backwards-compatible changes	20
6.1.1.3. Editorial changes	21
6.1.2. Guidelines for Package Versions During Package Development	21
7. Package Conformance	21
7.1. Resolving Conflicting Module Versions in Included Packages	22
7.2. Exclusions, Deviations	22
7.3. Use of features in YANG modules and YANG packages	23
7.4. Use of YANG semantic versioning	23

7.5. Implementing package versions	24
7.6. Use of deviations in YANG packages	24
7.7. The relationship between packages and datastores	24
8. YANG Modules	26
8.1. ietf-yang-package-types	26
8.2. ietf-yang-package-instance	38
8.3. ietf-yang-package	40
8.4. ietf-yl-packages	43
8.5. ietf-yang-inst-data-pkg	45
9. Security Considerations	47
10. IANA Considerations	48
11. Acknowledgements	50
12. References	50
12.1. Normative References	50
12.2. Informative References	52
Appendix A. Package Examples	53
A.1. Basic Package Examples	53
A.1.1. Example IETF Basic Types YANG package, version 1.0.0	54
A.1.2. Example IETF Basic Types YANG package, version 1.1.0	54
A.1.3. Example IETF Network Device YANG package	55
A.1.4. Example IETF Basic Routing YANG package	58
A.2. Package Versioning Examples	61
A.3. Package Resolution Examples	61
A.3.1. Example of package import conflict resolution	61
A.4. Package Exclusion Examples	64
A.4.1. Example of excluding modules and a mandatory feature	64
A.5. Mounted Package Examples	69
A.5.1. Example of package with a mounted package	69
A.5.2. Example of an package replacing the mounted schema	69
Authors' Addresses	69

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 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.
- * package schema: The combined set of schema nodes defined by a YANG package. Package schema can be used to define datastore schema.
- * 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 schema: The resolved schema is the YANG schema defined by a package after package resolution has been performed, as defined in Section 4. The resolved schema identifies the implemented modules (with any deviations applied), import-only modules, enabled features, and schema mount points.

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 implement some modules or packages, or it implements different versions/revisions, and/or some modules have deviations applied.
- * Tooling should be able to easily work with YANG package definitions to compare package versions and to compare server conformance against expected package definitions.

Protocol mechanisms of how clients can negotiate which packages or package versions are to be used for NETCONF/RESTCONF communications are outside the scope of this document. 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 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.
- * An "excludes" container comprising packages and modules that are implicitly included by entries in the "includes/packages" list, but are excluded from this package definition.
- * A "mandatory-features" container with two lists of YANG module features. The "include" list contains the features required in addition to those defined in included packages. The "exclude" list contains the features that have been defined as being mandatory in the directly included packages, but are excluded from this package definition.
- * 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                         yang:yang-identifier
    | | +-- version                       version-or-rev-date
    | | +-- location*                     inet:uri
    | | +-- submodule* [name]
    | | | +-- name?                       yang:yang-identifier
    | | | +-- version                       version-or-rev-date
    | | | +-- location*                     inet:uri
    | +-- import-only-module* [name version]
    | | +-- name                         yang:yang-identifier
    | | +-- version                       version-or-rev-date
    | | +-- replaces-version*             version-or-rev-date
    | | +-- location*                     inet:uri
    | | +-- submodule* [name]
    | | | +-- name?                       yang:yang-identifier
    | | | +-- version                       version-or-rev-date
    | | | +-- location*                     inet:uri
  +-- excludes
    | +-- module*                         pkg-name
    | +-- import-only-module*             pkg-name
  +-- mandatory-features
    | +-- include*                         scoped-feature
    | +-- exclude*                         scoped-feature
  +-- mounts* [mount-path]
    +-- mount-path?                       mount-ypath
    +-- package* [name]
    | +-- name                         pkg-name
    | +-- version                       pkg-version
    | +-- location*                     inet:uri
    | +-- replaces-package*             pkg-name
  +-- parent-reference*                   mount-ypath
```

3.1. Package definition rules

For a YANG package to be valid, it MUST conform to all of 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.
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/modules") or using automatic 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 'mandatory-features/include' and 'mandatory-features/exclude' lists.
11. Finally, standard rules for YANG instance data apply. In particular 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 5.2 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.
- * Each package definition may include a list of packages found at each mount point. Only a single version of any package found at a given mount point is allowed, and hence the version within the 'mounts/package' list replaces any mounted version exported from included packages, as defined in Section 4. In addition, the package mount definition allows for a package to replace one or more other packages at that mount point. Examples of this are given in Appendix A.5.2.

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. [TODO - Check if this is the best reference.]

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 5.3.3.

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 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 by any entries in the "includes/modules" list, and finally filtered by any entries in "excludes/modules" list.
- * The set of import-only modules is the union of all import-only modules in the resolved included packages, updated by any entries in the "includes/import-only-modules" list whilst taking the 'replaces-version' leaf-list into account, and finally filtered by any entries in "excludes/import-only-modules" list.
- * The set of mandatory-features is the union of the mandatory features in the resolved included packages, with any "mandatory-features/include" entries added, and any "mandatory-features/exclude" entries removed. If a module is excluded by "excludes/modules" 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 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 "mounts" list, with any entries listed in "mounts/package/replaces-package" list removed (or replaced) 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 through 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.

5. YANG Package Usage

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

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 5.4, 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 5.3.

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. 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.

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.

5.3. Providing Package Definitions on a Server

5.3.1. 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 5.3.3 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.

5.3.2. 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
```

5.3.3. 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 rules in Section 4.

If populated, the set of packages listed for a datastore schema MUST resolve to a schema that exactly matches the schema defined the YANG library module-sets, and the resolved schema MUST be referentially complete.

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
```

5.4. 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.

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

1. The file SHOULD be encoded in JSON.
2. The name of the file SHOULD follow the format "<package-name>@<version>.json".
3. The package name MUST be specified in both the instance-data-set 'name' and package 'name' leafs.
4. The 'description' field of the instance-data-set SHOULD be "YANG package definition".
5. The 'timestamp', 'organization', 'contact' fields are defined in both the instance-data-set 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 ...
```

5.5. 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.

6.1.1.1. 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.

6.1.1.1.1. 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.

- * Removing a feature from the 'mandatory-features/include' list unless the feature was not mandatory in any included packages.
- * Adding a feature to the 'mandatory-features/exclude' 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

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

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

For a set of packages representing a datastore schema, servers **SHOULD** implement the package definition faithfully, including all mandatory features.

Package definitions **MAY** modify the schema for directly or hierarchically included packages through the use of different module versions, use of module deviations, and changes to the mandatory features or mount points.

7.1. Resolving Conflicting Module Versions in Included Packages

Sometimes a package definition may include multiple packages that implement different versions of a module. A package must only implement a single version of a module, and hence in these cases 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. However, care should be taken if an older module version is chosen because this may cause a non-backwards-compatible change in one of the included packaged.

7.2. Exclusions, Deviations

If a server cannot faithfully implement a package then it can define a new package to accurately report what it does implement. The new package can include the original package as an included package, and the new package can define modified module versions, exclusions, or additional modules containing deviations to the modules in the original package, allowing the new package to accurately describe the server's behavior.

YANG Package definitions can exclude implemented modules and import-only modules. They can also exclude mandatory features, allowing a server implementation to not implement the feature. Both of these mechanism, whilst useful, should be used with great care, particularly if they break or change the resolved schema of any include packages.

However, it may be necessary for an implementation to use deviations to indicate where the server does not conform to the package, or even to exclude modules that are not implemented in their entirety. The **RECOMMENDED** mechanism for this is to advertise a separate "server implementation" package that includes the package to be conformed to,

and then excludes modules, adds deviation modules, and excluded mandatory-features to indicate the actual conformance of the server. 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.

7.3. Use of features in YANG modules and YANG packages

The YANG language supports feature statements as the mechanism to make parts of a schema optional. Published standard YANG modules 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 'mandatory-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.

7.4. Use of YANG semantic versioning

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

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

For example, a client coded to support '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.

7.5. Implementing package versions

TODO - Document differences in implementing package versions.

7.6. Use of deviations in YANG packages

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

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

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

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

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

7.7. The relationship between packages and datastores

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

- * 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 SHOULD be included in the package for configuration datastores if that helps unify the package definitions.
- * The packages for the operational datastore schema MUST include all packages for all configuration datastores, along with any required modules defining deviations to mark unsupported data nodes. The deviations MAY be defined directly in the packages defining the operational datastore schema, or in separate 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.
- * The resolved schema representing a datastore MUST be referentially complete.

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#0.6.0.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) 2025 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 2025-07-07 {
  ys:version 0.6.0;
  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 used YANG Semver version labels.";
}

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 defined 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 identifies 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 pkg-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.

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

  leaf-list import-only-module {
    type pkg-name;
    description
      "Lists import-only modules, of any version, 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.";
  }
}

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

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 yang:yang-identifier;
```

```
    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.";
  }
}
}
}

uses yang-pkg-exclusions;

container mandatory-features {
  description
    "Contains parameters for specifying the features that MUST
    be supported by any server implementing the package.";

  leaf-list include {
    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
      'mandatory-features/excludes' 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>.";

}

```
leaf-list exclude {  
  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 'mandatory-features/includes' list.

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

}

}

```
list mounts {
```

```
  key "mount-path";  
  description
```

"An entry in this list represents a package that will be found mounted in the schema at the specified mount path.

For a given mount path, the set of mounted package versions is the union of all packages mounted at the given mount point. Any conflicting package versions MUST be explicitly resolved via an entry in the mounted/packages of the package definition.

A mount path with specific keys MUST also includes any mounted packages without specific keys.";

```
leaf "mount-path" {  
  type mount-ypath;  
  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.";

}

```
list package {
  key "name";
  description
    "The packages that will be mounted at the specified mount
    path.

    The set of mounted packages is the union of all packages
    mounted at the given mount by any packages in the
    'includes/package' list, except that each entry in this list
    replaces any other versions of the same package at the
    same mount point. In addition, other package versions may
    be omitted from the mount point via the 'replaces-package'
    leaf-list.";
  uses yang-pkg-identification-leafs;
  uses yang-pkg-location;

  leaf-list replaces-package {
    type pkg-name;
    description
      "Lists other packages that have been explicitly mounted
      at the same mount point by included package, that are
      replaced by this mounted package.

      The replacing mounted package MUST explicitly include
      or exclude the mounted package being replaced.

      Any packages or modules included by the replaced
      package are also removed by this mounted package
```

unless they have also been explicitly mounted at the same mount point, in which case the replacing package MUST also explicitly include/exclude them.

replaces-package is expected to be used if an implementation does not fully implement a mounted package and needs to apply deviations or remove included packages or modules.";

```

    }
  }
  leaf-list parent-reference {
    type mount-ypath;
    description
      "See Mount Point path and parent-reference in Schema Mount";
  }
}
}
}
<CODE ENDS>

```

8.2. ietf-yang-package-instance

```

<CODE BEGINS> file "ietf-yang-package-instance#0.5.0.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 0.5.0;
    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) 2025 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 2025-03-03 {  
  ys:version 0.5.0;  
  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#0.5.0.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 0.5.0;
    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) 2025 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 2025-03-03 {  
  ys:version 0.5.0;  
  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#0.5.0.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-packages {
  prefix pkgs;
  ys:recommended-min-version 0.5.0;
  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) 2025 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 2025-03-03 {
  ys:version 0.5.0;
  description
    "Initial revision";
  reference
    "RFC XXXX: YANG Packages";
}

/*
 * Augmentations
 */

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

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

8.5. ietf-yang-inst-data-pkg

```
<CODE BEGINS> file "ietf-yang-inst-data-pkg#0.5.0.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 0.5.0;
    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) 2025 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 2025-03-03 {
    ys:version 0.5.0;
```

```
    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

TODO - Update this to more closely follow the latest template

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

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

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

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

As per the YANG library security considerations, the module, revision 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.

10. IANA Considerations

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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-netconf-rfc7895bis]

Bierman, A., Björklund, M., Schönwälder, J., Watsen, K., and R. Wilton, "YANG Library", Work in Progress, Internet-Draft, draft-ietf-netconf-rfc7895bis-07, 17 October 2018, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-rfc7895bis-07>>.

[I-D.ietf-netmod-yang-data-ext]

Bierman, A., Björklund, 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-13, 6 February 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-versioning-13>>.

[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-20, 7 February 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-semver-20>>.

- [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>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
<<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011,
<<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012,
<<https://www.rfc-editor.org/info/rfc6536>>.

- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <<https://www.rfc-editor.org/info/rfc8525>>.
- [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., Bjorklund, 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/>>.
- [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>>.
- [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>>.

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 5.4.

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.json"
===== 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.json"
===== 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.json"

===== 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.json" ],
        "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.json"
===== 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.json" ],
          }
        ]
      }
    }
  }
}
```

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

```

        " ],
    },
    {
        "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 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 would 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 would 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 would 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 Library format, but with namespaces and locations elided for brevity.

```
<CODE BEGINS> file "example-ab-pkg%0.1.0.json"
{
  "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"
            }
          ]
        },
        "mandatory-features": {
          "include": [
            "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.json"
{
  "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/ExampleAbPackage/makePackage.dest/yang-pkg.json"
              ]
            }
          ],
          "module": [
            {
              "name": "example-module-c",
              "version": "2.0.0"
            }
          ]
        },
        "excludes": {
          "module": [
            "example-module-b"
          ],
          "import-only-module": [
            "example-module-b-types"
          ]
        },
        "mandatory-features": {
          "exclude": [
            "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.json"
{
  "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. 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 would potentially help minimize wrapping?

A.5.1. 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.5.2. 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