

Network Working Group  
Internet-Draft  
Updates: 7950 (if approved)  
Intended status: Standards Track  
Expires: 16 April 2026

P. Andersson, Ed.  
Ionio Systems  
R. Wilton  
Cisco Systems, Inc.  
M. Valisko  
CESNET  
13 October 2025

YANG Schema Comparison  
draft-ietf-netmod-yang-schema-comparison-04

## Abstract

This document specifies an algorithm for comparing two revisions of a YANG schema to determine the scope of changes, and a list of changes, between the revisions. The output of the algorithm can be used to help select an appropriate revision-label or YANG semantic version number for a new revision. Included is also a YANG module describing the output of this algorithm.

## 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 16 April 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. Key Issues . . . . .	2
1.1. On-wire vs Schema analysis . . . . .	3
1.2. Comparison on module or full schema (YANG artifact, arbitrary blob. Questions . . . . .	4
2. Open Issues . . . . .	4
2.1. Override/per-node tags . . . . .	4
2.2. Separate rules for config vs state . . . . .	4
2.3. Tool/report verbosity . . . . .	4
2.4. sub-modules . . . . .	4
2.5. Write algorithm in pseudo code or just describe the rules/ goals in text? . . . . .	4
2.6. Categories in the report: bc, nbc, potentially-nbc, editorial. Allow filtering in the draft without defining it? . . . . .	4
2.7. Only for YANG 1.1? . . . . .	5
3. Introduction . . . . .	5
4. Terminology and Conventions . . . . .	5
5. Compiled YANG schema tree . . . . .	6
6. Compiled YANG schema tree comparison algorithm . . . . .	7
6.1. Module identification . . . . .	8
6.2. Change content . . . . .	8
6.3. Change conformance . . . . .	9
7. Comparison tooling . . . . .	9
8. Schema Comparison YANG Module . . . . .	9
9. Contributors . . . . .	30
10. Security Considerations . . . . .	31
11. IANA Considerations . . . . .	32
11.1. The "IETF XML" Registry . . . . .	32
11.2. The "YANG Module Names" Registry . . . . .	32
12. References . . . . .	32
12.1. Normative References . . . . .	32
12.2. Informative References . . . . .	35
Authors' Addresses . . . . .	35

## 1. Key Issues

{ This section is only to present the current ongoing work, not part of the final draft. }

The contributors have identified several key issues that need attention. This section presents selected key issues which have been discussed together with suggestions for proposed solution or requirements.

### 1.1. On-wire vs Schema analysis

Should one algorithm be used or two? The consensus reached was to define two separate algorithms, one for on-wire format and one for schema.

Schema: any changes that affect the YANG schema in an NBC manner according to the full rules of [I-D.ietf-netmod-yang-module-versioning]. This may be important for clients that, for example, automatically generate code using the YANG and where the change of a typedef name or a choice name could be significant. Also important for other modules that may augment or deviate the schema being compared.

Changes to the module that aren't semantic should raise that there has been editorial changes

Ordering in the schema, RFC 7950 doesn't allow reordering; thus an NBC change.

Open Questions:

Groupings / uses

typedefs, namespaces, choice names, prefixes, module metadata.

- \* typedef renaming (on-wire, same base type etc)
- \* Should all editorial (text) diffs be reported?
- \* What about editorial changes that might change semantics, e.g. a description of a leaf?
- \* Metadata arguments which relies on the formatted input text. E.g description, contact (etc), extension (how does the user want to tune verbosity level for editorial changes: whitespace, spelling, editorial, potentially-nbc?
- \* XPath, must, when: don't normalize XPath expressions
- \* presence statements

## 1.2. Comparison on module or full schema (YANG artifact, arbitrary blob. Questions

- \* features
- \* packages vs directories vs libraries vs artifact
- \* package specific comparison, package metadata or only looking at the modules
- \* import only or implemented module

Filter out comparison for a specific subtree, path etc. Use case for on-wire e.g. yang subscriptions, did the model change fro what is subscribed on?

## 2. Open Issues

{ This section is only to present the current ongoing work, not part of the final draft. }

The following issues have not ben discussed in any wider extent yet.

### 2.1. Override/per-node tags

### 2.2. Separate rules for config vs state

### 2.3. Tool/report verbosity

- \* where to report changes (module, grouping, typedef, uses)
- \* output level (conceptual level or exact strings)
- \* granularity: error/warning/info level per reported change category

### 2.4. sub-modules

### 2.5. Write algorithm in pseudo code or just describe the rules/goals in text?

### 2.6. Categories in the report: bc, nbc, potentially-nbc, editorial. Allow filtering in the draft without defining it?

One option can be to have a tool option that presents the reason behind the decision, e.g. --details could be used to explain to the user why a certain change was marked as nbc.

Another option is to present reasoning and analysis in deeper levels of verbosity; e.g. one extra level of verbosity, `-v`, could present the reason for categorizing a change `nb`, and an additional extra level of verbosity, e.g. `-vv`, could also present the detailed analysis the tool made to categorize the change.

## 2.7. Only for YANG 1.1?

## 3. Introduction

This document defines the algorithm for comparing two revisions of a YANG schema. There are two kinds of YANG schemas: on-the-wire or compiled schema and text or parsed schema.

The compiled schema is what tools use when working with any kinds of YANG data. Generally, in all the YANG modules, all the schema nodes need to be fully resolved and have their final form. For example, every "uses" statement is replaced with the nodes from the corresponding "grouping", every type referencing a "typedef" is replaced with the type of the "typedef" and so on. On the other hand, the parsed schema is comprised of all the YANG modules together as they are. That means no statements are resolved and they remain in exactly the same text form as present in the YANG modules.

For determining whether a new revision of a YANG 1.1 module is or is not backwards-compatible based on all the rules in [RFC7950] and [I-D.ietf-netmod-yang-module-versioning], both of these schemas would be required. Similarly for YANG 1.0, the rules in [RFC6020]. This document defines only the algorithm for comparing the compiled schema. The output of this algorithm can be utilized by tools processing YANG instance data to determine what exact changes should they expect and even how to transform instance data valid for an old revision of a YANG module to be valid for a newer revision of the same YANG module. Such tools are typically clients or servers of YANG-driven protocols, e.g. NETCONF, RESTCONF, or gNMI.

In addition, the comparison algorithm can assist YANG module authors with updating their modules and be used to verify the semantic version of the new revision of the YANG module based on the [I-D.ietf-netmod-yang-semver] document.

## 4. 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 makes use of the following terminology introduced in the YANG 1.1 Data Modeling Language [RFC7950]:

- \* schema node

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

This document makes use of the following terminology introduced in the YANG Packages [I-D.ietf-netmod-yang-packages]:

- \* YANG schema

In addition, this document defines the terminology:

- \* Change scope: Whether a change between two revisions is classified as non-backwards-compatible or backwards-compatible.
- \* Node compatibility statement: An extension statements (e.g. nbc-change-at) that can be used to indicate the backwards compatibility of individual schema nodes and specific YANG statements.

## 5. Compiled YANG schema tree

The compiled schema tree resolves all the statements used for syntactic abstraction and simplification to obtain one schema node tree with only data definition statements. Textual fields are kept as well because they may affect backwards compatibility of the whole YANG module.

The steps taken when compiling a schema:

- \* 'Import' statements are resolved by finding the referenced YANG modules and making their contents available.
- \* All submodules of a single module are merged to create one large YANG module.
- \* 'Uses' statements are replaced by the referenced 'groupings' and the YANG nodes they define.
- \* 'Types' referencing any 'typedef' statements are replaced with the actual type details.
- \* 'Augments' are resolved by placing the included YANG nodes in their targets.

- \* 'Deviation' and 'refine' statements are applied by changing the appropriate schema nodes.
- \* All the 'if-feature' statements are evaluated based on the enabled 'features' of implemented YANG modules and the disabled schema nodes are removed.

After the compilation the result is still a valid YANG module without the following statements:

- \* import
- \* include
- \* revision
- \* typedef
- \* grouping
- \* augment
- \* deviation
- \* feature
- \* if-feature
- \* refine
- \* extension

#### 6. Compiled YANG schema tree comparison algorithm

This algorithm is defined for separate compiled YANG modules. It first compares all the non data-definition statements in the module and then all the data-definition statements recursively. These schema-only statements are skipped:

- \* choice
- \* case

The non data-definition statements are considered modified when their content is changed or any of their substatements are changed. If a statement is present in the old revision of a module and missing in the new revision, it is considered removed. Vice-versa, if missing in the old revision but present in the new revision, it is considered added.

Data-definition statements are compared similarly except that a change in a descendant data-definition statement is not considered a change of the parent data-definition statement but a separate change instead.

Only statements with any changes are reported and included in the algorithm output.

#### 6.1. Module identification

When generating the comparison output of two compiled YANG modules, it includes unique identification of the modules in question. For both the old and the new revision of the module, its name, revision, list of submodules, and a leaf-list of enabled features is included. In addition, the same information is reported for all the imported YANG modules, recursively. This ensures that if any other output is produced with a different set of changes for two YANG modules, their identification information must also differ.

#### 6.2. Change content

Non-data-definition statements are uniquely identified with their YANG identifier and data-definition statements with their absolute schema data node-ids.

For every change, the changed statement and the parent of the changed statement are included. While redundant in many cases, it allows to uniquely report changes in nested statements, e.g. an extension instance change in the type of a leaf statement. Also, all the substatements with their value or content are included for every changed statement, for both older and newer revision of the YANG module, if applicable. This ensures that the output includes all the possibly required information without relying on learning it from the YANG modules themselves.



### 6.3. Change conformance

Every change is classified as either backwards-compatible or non-backwards-compatible based on sections 3.1.1 and 3.1.2 in [I-D.ietf-netmod-yang-module-versioning]. But, there are cases when it is impossible for tooling to determine whether a change is backwards-compatible or not. The affected statements are:

- \* pattern
- \* when
- \* must
- \* description
- \* extension instance

Normally, any change in these statement is reported as non-backwards-compatible. However, if the author of the new revision of a YANG module determines that the change of an aforementioned statement is backwards-compatible based on the compatibility rules for the specific statement, they can decide to use the 'schema-cmp:backwards-compatible' extension to signify this fact. Then, this change must be reported as backwards-compatible.

### 7. Comparison tooling

The 'yanglint' tool from the libyang library is able to generate the 'ietf-schema-comparison' YANG data of two revisions of a YANG module.

### 8. Schema Comparison YANG Module

YANG module with nodes describing the differences between two revisions of compiled YANG modules. The "ietf-schema-comparison" YANG module imports definitions from the "ietf-yang-types" module defined in [RFC6991], "ietf-yang-library" module defined in [RFC8525], and "ietf-yang-structure-ext" module defined in [RFC8791].

```
<CODE BEGINS> file "ietf-yang-schema-comparison@2025-10-13.yang"
module ietf-yang-schema-comparison {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-schema-comparison";
  prefix schema-cmp;

  import ietf-yang-types {
    prefix yang;
    reference
```

```
"RFC 6991: Common YANG Data types";
}
import ietf-yang-library {
  prefix yanglib;
  reference
    "RFC 8525: YANG Library";
}
import ietf-yang-structure-ext {
  prefix sx;
  reference
    "RFC 8791: YANG Data Structure Extensions";
}

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

contact
  "WG List: NETCONF WG list <mailto:netmod@ietf.org>
  WG Web:  https://datatracker.ietf.org/wg/netmod

  Author:  Michal Vasko
           <mailto:mvasko@cesnet.cz>

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

  Author:  Per Andersson
           <mailto:per.ietf@ionio.se>";

description
  "This YANG 1.1 module contains definitions and extensions to
  support comparison between different versions of YANG modules. The
  output of the comparison algorithm is described in this YANG module.

  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
  (https://www.rfc-editor.org/info/rfcXXXX); 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 (inc above) with actual RFC number and
// remove this note.
```

```
revision 2025-10-13 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Schema Comparison";
}
```

```
extension backwards-compatible {
  description
    "Marks statements in a new revision of a module
    that were changed but in a backwards-compatible
    manner according to YANG 1.1 and 1.0 rules for
    updating a module.

    When used as a substatement of a 'pattern'
    statement, it means its allowed value space has
    been expanded.

    When used as a substatement of a 'when' or
    'must' statement, it means their constraint has
    been relaxed.

    When used as a substatement of a 'description'
    statement, it means it has been changed in a
    backwards-compatible way.

    When used as a substatement of an extension
    instance, it means adding, modifying, or removing
    it is a backwards-compatible change.

    Not allowed to be instantiated for any other
    statements.";
  reference
    "RFC 7950: The YANG 1.1 Data Modeling Language, section 11;
    RFC 6020: YANG - A Data Modeling Language for the Network
    Configuration Protocol (NETCONF), section 10";
}
```

```
typedef data-path {
  type string;
  description
    "Used for any data paths and XPath expressions in the compared YANG
    modules. A generic type is used to not enforce their evaluation or
    validation in this context.";
}

typedef stmt-type {
  type enumeration {
    enum "base" {
      description
        "YANG statement 'base'.";
    }
    enum "bit" {
      description
        "YANG statement 'bit'.";
    }
    enum "config" {
      description
        "YANG statement 'config'.";
    }
    enum "contact" {
      description
        "YANG statement 'contact'.";
    }
    enum "default" {
      description
        "YANG statement 'default'.";
    }
    enum "description" {
      description
        "YANG statement 'description'.";
    }
    enum "enum" {
      description
        "YANG statement 'enum'.";
    }
    enum "error-app-tag" {
      description
        "YANG statement 'error-app-tag'.";
    }
    enum "error-message" {
      description
        "YANG statement 'error-message'.";
    }
    enum "extension-instance" {
      description
```

```
    "Any extension instance YANG statement.";
  }
  enum "fraction-digits" {
    description
      "YANG statement 'fraction-digits'.";
  }
  enum "identity" {
    description
      "YANG statement 'identity'.";
  }
  enum "length" {
    description
      "YANG statement 'length'.";
  }
  enum "mandatory" {
    description
      "YANG statement 'mandatory'.";
  }
  enum "max-elements" {
    description
      "YANG statement 'max-elements'.";
  }
  enum "min-elements" {
    description
      "YANG statement 'min-elements'.";
  }
  enum "must" {
    description
      "YANG statement 'must'.";
  }
  enum "node" {
    description
      "YANG statement 'container', 'leaf', 'leaf-list', 'list', 'anydata',
        'anyxml', 'rpc', 'action', or 'notification'.";
  }
  enum "ordered-by" {
    description
      "YANG statement 'ordered-by'.";
  }
  enum "organization" {
    description
      "YANG statement 'organization'.";
  }
  enum "path" {
    description
      "YANG statement 'path'.";
  }
  enum "pattern" {
```

```
        description
            "YANG statement 'pattern'.";
    }
    enum "presence" {
        description
            "YANG statement 'presence'.";
    }
    enum "range" {
        description
            "YANG statement 'range'.";
    }
    enum "reference" {
        description
            "YANG statement 'reference'.";
    }
    enum "require-instance" {
        description
            "YANG statement 'require-instance'.";
    }
    enum "status" {
        description
            "YANG statement 'status'.";
    }
    enum "type" {
        description
            "YANG statement 'type'.";
    }
    enum "units" {
        description
            "YANG statement 'units'.";
    }
    enum "unique" {
        description
            "YANG statement 'unique'.";
    }
    enum "when" {
        description
            "YANG statement 'when'.";
    }
}
description
    "Type of the statement that a change affects.";
}

typedef revision-or-empty {
    type union {
        type empty;
        type yanglib:revision-identifier;
    }
}
```

```
    }
    description
      "Module or submodule revision. If it has none, an empty value is
      used.";
  }

  grouping ext-instance-changes {
    description
      "Describes an extension-instance statement change.";
    leaf module {
      type yang:yang-identifier;
      mandatory true;
      description
        "Module with the extension definition of an extension instance.";
    }
    leaf name {
      type yang:yang-identifier;
      mandatory true;
      description
        "Name of the extension definition of an extension instance.";
    }
    leaf argument {
      type string;
      description
        "Argument used of an extension definition.";
    }
    anydata substatements {
      description
        "Any substatements of the extension instance.";
    }
  }

  grouping status-changes {
    description
      "Describes a status statement change.";
    leaf status {
      type enumeration {
        enum "current" {
          description
            "Current effective status of a statement.";
        }
        enum "deprecated" {
          description
            "Deprecated effective status of a statement.";
        }
        enum "obsolete" {
          description
            "Obsolete effective status of a statement.";
        }
      }
    }
  }
```

```
    }
  }
  mandatory true;
  description
    "Status substatement value.";
}
}

grouping must-changes {
  description
    "Describes a must statement change.";
  leaf condition {
    type data-path;
    description
      "Condition substatement value.";
  }
  leaf description {
    type string;
    description
      "Description substatement value.";
  }
  leaf reference {
    type string;
    description
      "Reference substatement value.";
  }
  leaf error-message {
    type string;
    description
      "Error-message substatement value.";
  }
  leaf error-app-tag {
    type string;
    description
      "Error-app-tag substatement value.";
  }
  list ext-instance {
    description
      "List of extension-instance substatements.";
    uses ext-instance-changes;
  }
}

grouping when-changes {
  description
    "Describes a when statement change.";
  leaf condition {
    type data-path;
```



```
        description
            "Condition substatement value.";
    }
    leaf description {
        type string;
        description
            "Description substatement value.";
    }
    leaf reference {
        type string;
        description
            "Reference substatement value.";
    }
    uses status-changes;
    list ext-instance {
        description
            "List of extension-instance substatements.";
        uses ext-instance-changes;
    }
}

grouping restriction-substmts {
    description
        "Common substatements shared by all restriction statements.";
    leaf description {
        type string;
        description
            "Description substatement value.";
    }
    leaf reference {
        type string;
        description
            "Reference substatement value.";
    }
    leaf error-message {
        type string;
        description
            "Error-message substatement value.";
    }
    leaf error-app-tag {
        type string;
        description
            "Error-app-tag substatement value.";
    }
    list ext-instance {
        description
            "List of extension-instance substatements.";
        uses ext-instance-changes;
    }
}
```

```
    }  
  }  
  
  grouping type-changes {  
    description  
      "Describes a type statement change.";  
    leaf base-type {  
      type enumeration {  
        enum "int8" {  
          description  
            "YANG built-in type 'int8'.";  
        }  
        enum "int16" {  
          description  
            "YANG built-in type 'int16'.";  
        }  
        enum "int32" {  
          description  
            "YANG built-in type 'int32'.";  
        }  
        enum "int64" {  
          description  
            "YANG built-in type 'int64'.";  
        }  
        enum "uint8" {  
          description  
            "YANG built-in type 'uint8'.";  
        }  
        enum "uint16" {  
          description  
            "YANG built-in type 'uint16'.";  
        }  
        enum "uint32" {  
          description  
            "YANG built-in type 'uint32'.";  
        }  
        enum "uint64" {  
          description  
            "YANG built-in type 'uint64'.";  
        }  
        enum "decimal64" {  
          description  
            "YANG built-in type 'decimal64'.";  
        }  
        enum "string" {  
          description  
            "YANG built-in type 'string'.";  
        }  
      }  
    }  
  }  
}
```

```
enum "boolean" {
  description
    "YANG built-in type 'boolean'.";
}
enum "enumeration" {
  description
    "YANG built-in type 'enumeration'.";
}
enum "bits" {
  description
    "YANG built-in type 'bits'.";
}
enum "binary" {
  description
    "YANG built-in type 'binary'.";
}
enum "leafref" {
  description
    "YANG built-in type 'leafref'.";
}
enum "identityref" {
  description
    "YANG built-in type 'identityref'.";
}
enum "empty" {
  description
    "YANG built-in type 'empty'.";
}
enum "union" {
  description
    "YANG built-in type 'union'.";
}
enum "instance-identifier" {
  description
    "YANG built-in type 'instance-identifier'.";
}
}
description
  "Type substatement value.";
}
container range {
  description
    "Range statement substatements.";
  list interval {
    description
      "Intervals of a range statement.";
    leaf min {
      type int64;
    }
  }
}
```

```
        description
            "Lower boundary of an interval of a range statement.";
    }
    leaf max {
        type int64;
        description
            "Upper boundary of an interval of a range statement.";
    }
}
uses restriction-substmts;
}
container length {
    description
        "Length statement substatements.";
    list interval {
        description
            "Intervals of a length statement.";
        leaf min {
            type uint64;
            description
                "Lower boundary of an interval of a length statement.";
        }
        leaf max {
            type uint64;
            description
                "Upper boundary of an interval of a length statement.";
        }
    }
    uses restriction-substmts;
}
leaf fraction-digits {
    type uint8;
    description
        "Fraction-digits substatement value.";
}
list pattern {
    description
        "List of pattern statements.";
    leaf expression {
        type string;
        description
            "Pattern substatement value.";
    }
    leaf inverted {
        type boolean;
        description
            "Inverted substatement value.";
    }
}
```

```
    uses restriction-substmts;
}
list enum {
  key "name";
  ordered-by user;
  description
    "List of enum statements. Keeps the same order of the statements as in
    the module.";
  leaf name {
    type yang:yang-identifier;
    description
      "Enum statement name.";
  }
  leaf description {
    type string;
    description
      "Description substatement value.";
  }
  leaf reference {
    type string;
    description
      "Reference substatement value.";
  }
  leaf value {
    type int32;
    description
      "Value substatement value.";
  }
  uses status-changes;
  list ext-instance {
    description
      "List of extension-instance substatements.";
    uses ext-instance-changes;
  }
}
list bit {
  key "name";
  ordered-by user;
  description
    "List of bit statements. Keeps the same order of the statements as in
    the module.";
  leaf name {
    type yang:yang-identifier;
    description
      "Bit statement name.";
  }
  leaf description {
    type string;
```

```
        description
            "Description substatement value.";
    }
    leaf reference {
        type string;
        description
            "Reference substatement value.";
    }
    leaf position {
        type uint32;
        description
            "Position substatement value.";
    }
    uses status-changes;
    list ext-instance {
        description
            "List of extension-sintance substatements.";
        uses ext-instance-changes;
    }
}
leaf path {
    type data-path;
    description
        "Path substatement value.";
}
leaf require-instance {
    type boolean;
    description
        "Require-instance substatement value.";
}
leaf-list base {
    type yang:yang-identifier;
    description
        "List of base substatement values.";
}
list ext-instance {
    description
        "List of extension-instance substatements.";
    uses ext-instance-changes;
}
}

grouping node-changes {
    description
        "All compiled substatements of a data-definition node.";
    leaf config {
        type boolean;
        description
```

```
        "Config substatement value.";
    }
    leaf description {
        type string;
        description
            "Description substatement value.";
    }
    leaf mandatory {
        type boolean;
        description
            "Mandatory substatement value.";
    }
    list must {
        description
            "List of must substatements.";
        uses must-changes;
    }
    leaf presence {
        type boolean;
        description
            "Presence substatement existence.";
    }
    leaf reference {
        type string;
        description
            "Reference substatement value.";
    }
    uses status-changes;
    list when {
        description
            "List of when substatements.";
        uses when-changes;
    }
    container type {
        description
            "Type substatement.";
        uses type-changes;
        list union-type {
            description
                "List of a union type substatements.";
            uses type-changes;
        }
    }
    leaf units {
        type string;
        description
            "Units substatement value.";
    }
}
```

```
leaf ordered-by {
  type enumeration {
    enum "system" {
      description
        "System-ordered list or leaf-list of items.";
    }
    enum "user" {
      description
        "User-ordered list or leaf-list of items.";
    }
  }
  description
    "Ordered-by substatement value.";
}
leaf-list default {
  type string;
  description
    "List of default substatement values.";
}
leaf min-elements {
  type uint32;
  description
    "Min-elements substatement value.";
}
leaf max-elements {
  type uint32;
  description
    "Max-elements substatement value.";
}
list unique {
  description
    "List of unique substatements.";
  leaf-list node {
    type yang:yang-identifier;
    description
      "List of nodes referenced by a unique substatement.";
  }
}
list ext-instance {
  description
    "List of extension-instance substatements.";
  uses ext-instance-changes;
}
}

grouping module-changes {
  description
    "All compiled non-data-definition substatements of a module.";
}
```



```
leaf organization {
  type string;
  description
    "Organization substatement value.";
}
leaf contact {
  type string;
  description
    "Contact substatement value.";
}
leaf description {
  type string;
  description
    "Description substatement value.";
}
leaf reference {
  type string;
  description
    "Reference substatement value.";
}
list identity {
  key "name";
  description
    "List of identity substatements.";
  leaf name {
    type yang:yang-identifier;
    description
      "Identity statement name.";
  }
  list ext-instance {
    description
      "List of extension-instance substatements.";
    uses ext-instance-changes;
  }
}
list ext-instance {
  description
    "List of extension-instance substatements.";
  uses ext-instance-changes;
}
}

grouping conformance-type {
  description
    "Conformance type of a change.";
  leaf conformance {
    type enumeration {
      enum "backwards-compatible" {
```

```
        description
            "Backwards-compatible or editorial change.";
    }
    enum "non-backwards-compatible" {
        description
            "Non-backwards-compatible change.";
    }
}
mandatory true;
description
    "Conformance information of a change in the compared YANG modules.";
}
}

grouping change-info {
    description
        "Details of a single change.";
    list changed {
        key "stmt";
        min-elements 1;
        description
            "List of changes.";
        leaf stmt {
            type stmt-type;
            description
                "Changed statement.";
        }
        leaf parent-stmt {
            type stmt-type;
            description
                "Parent statement of the changed statement.";
        }
        leaf change {
            type enumeration {
                enum "modified" {
                    description
                        "Statement existed and was modified.";
                }
                enum "added" {
                    description
                        "Statement was added.";
                }
                enum "removed" {
                    description
                        "Statement was removed.";
                }
                enum "moved" {
                    description
```

```
        "Statement was moved and the position affects the meaning.";
    }
}
mandatory true;
description
    "Type of change of the statement.";
}
uses conformance-type;
}
}

grouping module-params {
    description
        "Identification setails of a processed YANG module.";
    leaf module {
        type yang:yang-identifier;
        mandatory true;
        description
            "Compared module name.";
    }
    leaf revision {
        type revision-or-empty;
        mandatory true;
        description
            "Compared module revision.";
    }
    list submodule {
        key "name revision";
        leaf name {
            type yang:yang-identifier;
            mandatory true;
            description
                "Submodule name.";
        }
        leaf revision {
            type revision-or-empty;
            mandatory true;
            description
                "Submodule revision.";
        }
    }
    description
        "Submodule of the main module.";
}
leaf-list enabled-feature {
    type yang:yang-identifier;
    description
        "All the enabled features of the module.";
}
```

```
}

sx:structure schema-comparison {
  description
    "Schema comparison details.";
  list compiled-diff {
    description
      "Instance of a schema comparison of 2 YANG modules in different
      revisions with all their imports.";
    container source {
      description
        "Source module information.";
      uses module-params;
    }
    list source-import {
      key "module revision";
      description
        "List of source module imported modules information.";
      uses module-params;
    }
    container target {
      description
        "Target module information.";
      uses module-params;
    }
    list target-import {
      key "module revision";
      description
        "List of target module imported modules information.";
      uses module-params;
    }
    uses conformance-type;
    container module-diff {
      description
        "Information about direct module statement substatement changes.";
      uses change-info;
      container old {
        description
          "Changed statements in the older revision of the YANG module.";
        uses module-changes;
      }
      container new {
        description
          "Changed statements in the newer revision of the YANG module.";
        uses module-changes;
      }
    }
  }
  list node-diff {
```

```
key "node";
ordered-by user;
description
  "Information about data-definition (node) statement substatement
  changes. Keeps the traversed order of the nodes by the algorithm.
  It SHOULD follow the depth-first search order.";
leaf node {
  type data-path;
  description
    "Path to the changed data-definition statement.";
}
leaf node-type {
  type enumeration {
    enum "container" {
      description
        "YANG statement 'container' node.";
    }
    enum "leaf" {
      description
        "YANG statement 'leaf' node.";
    }
    enum "leaf-list" {
      description
        "YANG statement 'leaf-list' node.";
    }
    enum "list" {
      description
        "YANG statement 'list' node.";
    }
    enum "anydata" {
      description
        "YANG statement 'anydata' node.";
    }
    enum "anyxml" {
      description
        "YANG statement 'anyxml' node.";
    }
    enum "rpc" {
      description
        "YANG statement 'rpc' node.";
    }
    enum "action" {
      description
        "YANG statement 'action' node.";
    }
    enum "notification" {
      description
        "YANG statement 'notification' node.";
    }
  }
}
```

```

    }
  }
  mandatory true;
  description
    "Type of the changed data-definition statement.";
}
leaf in-rpc-action {
  type enumeration {
    enum "input" {
      description
        "Data-definition statement is a descendant if an input
        statement.";
    }
    enum "output" {
      description
        "Data-definition statement is a descendant if an output
        statement.";
    }
  }
}
description
  "Present if the changed data-definition statement is a descendant
  of an action or rpc statement.";
}
uses change-info;
container old {
  description
    "All the node substatements in the older revision of the YANG
    module.";
  uses node-changes;
}
container new {
  description
    "All the node substatements in the newer revision of the YANG
    module.";
  uses node-changes;
}
}
}
}
}
<CODE ENDS>

```

## 9. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following individuals are (or have been) members of the design team and have worked on the YANG versioning project:

- \* Balazs Lengyel
- \* Benoit Claise
- \* Bo Wu
- \* Ebben Aries
- \* Jason Sterne
- \* Joe Clarke
- \* Juergen Schoenwaelder
- \* Mahesh Jethanandani
- \* Michael Wang
- \* Qin Wu
- \* Reshad Rahman
- \* Rob Wilton
- \* Jan Lindblad
- \* Per Andersson

The ideas for a tooling based comparison of YANG module revisions was first described in [I-D.clacla-netmod-yang-model-update]. This document extends upon those initial ideas.

## 10. Security Considerations

This section follows the template defined in Section 3.7.1 of [RFC8407].

The YANG module specified in this document defines a schema for data that is designed to be used by offline tooling to generate output for differences in supplied YANG modules.

The YANG module specified in this document MAY be accessed via 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 [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

When the module is used for offline tooling there are no security considerations, since the user has full access to all YANG modules used.

The structure "schema-comparison" contains all the groupings reflecting the changes between YANG modules. If the data of this structure is published in online tooling, care needs to be taken so that knowledge of YANG modules are not leaked.

Since the module does not define any RPCs, actions, or notifications, the security considerations for such statements are not provided here.

## 11. IANA Considerations

### 11.1. The "IETF XML" Registry

This document register one URI in the "ns" subregistry of the IETF XML Registry [RFC3688] maintained at <https://www.iana.org/assignments/xml-registry/xml-registry.xhtml#ns>. Following the format in [RFC3688], the following registration is requested:

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

### 11.2. The "YANG Module Names" Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020] maintained at <https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml>. Following the format defined in [RFC6020], the below registration is requested:

Name: ietf-yang-schema-comparison  
XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-schema-comparison  
Prefix: schema-cmp  
Reference: RFC XXXX

## 12. References

### 12.1. Normative References



- [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-14, 6 October 2025,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-versioning-14>>.
- [I-D.ietf-netmod-yang-packages]  
Wilton, R., Rahman, R., Clarke, J., and J. Sterne, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-06, 7 July 2025,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-packages-06>>.
- [I-D.ietf-netmod-yang-semver]  
Clarke, J., Wilton, R., Rahman, R., Lengyel, B., Sterne, J., and B. Claise, "YANG Semantic Versioning", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-semver-24, 29 September 2025,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-semver-24>>.
- [I-D.ietf-netmod-yang-solutions]  
Wilton, R., "YANG Versioning Solution Overview", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-solutions-01, 2 November 2020,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-solutions-01>>.
- [I-D.ietf-netmod-yang-versioning-reqs]  
Clarke, J., "YANG Module Versioning Requirements", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-versioning-reqs-12, 21 July 2025,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-versioning-reqs-12>>.
- [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>>.

- [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>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [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>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [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>>.

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

## 12.2. Informative References

[I-D.clacla-netmod-yang-model-update]  
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", Work in Progress, Internet-Draft, draft-clacla-netmod-yang-model-update-06, 2 July 2018, <<https://datatracker.ietf.org/doc/html/draft-clacla-netmod-yang-model-update-06>>.

## Authors' Addresses

Per Andersson (editor)  
Ionio Systems  
Email: [per.ietf@ionio.se](mailto:per.ietf@ionio.se)

Robert Wilton  
Cisco Systems, Inc.  
Email: [rwilton@cisco.com](mailto:rwilton@cisco.com)

Michal Vasko  
CESNET  
Email: [mvasko@cesnet.cz](mailto:mvasko@cesnet.cz)