Workgroup: Network Working Group
Internet-Draft:
draft-ietf-netmod-yang-schema-comparison-02
Updates: 7950 (if approved)
Published: 11 March 2023
Intended Status: Standards Track
Expires: 12 September 2023
Authors: P. Andersson, Ed.
Cisco Systems, Inc.
R. Wilton
Cisco Systems, Inc.

YANG Schema Comparison

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. This document defines a YANG extension that provides YANG annotations to help the tool accurately determine the scope of changes between two revisions.

Table of Contents

1. Key Issues
  1.1. On-wire vs Schema analysis
  1.2. error-tags, error messages, and other error statements
  1.3. Comparison on module or full schema (YANG artifact, arbitrary blob. Questions
2. Open Issues
  2.1. Override/per-node tags
  2.2. Separate rules for config vs state
  2.3. Tool/report verbosity
  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?
  2.7. Only for YANG 1.1?
  2.8. renamed-from
3. Tool options
4. Introduction
5. Terminology and Conventions
6. Generic YANG schema tree comparison algorithm
  6.1. YANG module revision scope extension annotations
  6.2. Node compatibility extension statements
7. YANG module comparison algorithm
8. YANG schema comparison algorithms
  8.1. Standard YANG schema comparison algorithm
  8.2. Filtered YANG schema comparison algorithm
9. Comparison tooling
10. Module Versioning Extension YANG Modules
11. Contributors
12. Security Considerations
13. IANA Considerations
  13.1. YANG Module Registrations
14. References
  14.1. Normative References
  14.2. Informative References
Authors' Addresses

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 consesus reached was to define two separate algorithms, one for on-wire format and one for schema.

On the wire: the focus is on what types of changes affect the client requests and server responses for YANG driven protocols, e.g. NETCONF, RESTCONF, gNMI. If the same requests and responses occur, then there is no "on the wire" impact of the change. For example, changing the name of a "choice" has no impact "on the wire". For many clients, this level of compatiblity is enough.

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. error-tags, error messages, and other error statements

Error tags and messages might be relied on verbatim by users.

*error-tag: standardized in [RFC6241]

*error-app-tag: arbitrary text ([RFC6241] but also model)

*error-message: arbitrary

Failed must statement, error-message, assumed NBC

Default behaviour is changes to error tags, messages etc are NBC.

1.3. 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 subrtree, 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 nbc, 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?

2.8. renamed-from

3. Tool options

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

During the work a list of useful tool options are identified for later discussion and publication in an appendix.

*An option for how to interpret description changes (for the on-wire algorithm) by default, e.g. treat them as editorial or nbc.

*Option: --skip-error-tags, etc

4. Introduction

Warning, this is an early (-00) draft with the intention of scoping the outline of the solution, hopefully for the WG to back the direction of the solution. Refinement of the solution details is expected, if this approach is accepted by the WG.

This document defines a solution to Requirement 2.2 in [I-D.ietf-netmod-yang-versioning-reqs]. Complementary documents provide a complete solution to the YANG versioning requirements, with the overall relationship of the solution drafts described in [I-D.ietf-netmod-yang-solutions].

YANG module 'revision-labels'
[I-D.ietf-netmod-yang-module-versioning] and the use of YANG
semantic version numbers [I-D.ietf-netmod-yang-semver] can be used
to help manage and report changes between revisions of individual
YANG modules.

YANG packages [I-D.ietf-netmod-yang-packages] along with YANG
semantic version numbers can be used to help manage and report
changes between revisions of YANG schema.

[I-D.ietf-netmod-yang-module-versioning] and
[I-D.ietf-netmod-yang-packages] define how to classify changes
between two module or package revisions, respectively, as backwards
compatible or non-backwards-compatible.
[I-D.ietf-netmod-yang-semver] refines the definition, to allow
backwards compatible changes to be classified as 'minor changes' or
'editorial changes'.

'Revision-label's and YANG semantic version numbers, whilst being
generally simple and helpful in the mainline revision history case,
are not sufficient in all scenarios. For example, when comparing two
revisions/versions on independent revision branches, without a
direct ancestor relationship between the two revisions/versions. In
this cases, an algorithmic comparison approach is beneficial.

In addition, the module revision history's 'nbc-changes' extension
statement, and YANG semantic version numbers, effectively declare
the worst case scenario. If any non-backwards-compatible changes are
restricted to only parts of the module/schema that are not used by
an operator, then the operator is able to upgrade, and effectively
treat the differences between the two revisions/versions as
backwards compatible because they are not materially impacted by the
non-backwards-compatible changes.

Hence, this document defines algorithms that can be applied to
revisions of YANG modules or versions of YANG schema (e.g., as
represented by YANG packages), to determine the changes, and scope
of changes between the revisions/versions.

For many YANG statements, programmatic tooling can determine whether
the changes between the statements constitutes a backwards-
compatible or non-backwards-compatible change. However, for some
statements, it is not feasible for current tooling to determine
whether the changes are backwards-compatible or not. For example, in
the general case, tooling cannot determine whether the change in a
YANG description statement causes a change in the semantics of a
YANG data node. If the change is to fix a typo or spelling mistake
then the change can be classified as an editorial backwards-
compatible change. Conversely, if the change modifies the behavioral

specification of the data node then the change would need to be classified as either a non editorial backwards-compatible change or a non-backwards-compatible change. Hence, extension statements are defined to annotate a YANG module with additional information to clarify the scope of changes in cases that cannot be determined by algorithmic comparison.

Open issues are tracked at https://github.com/netmod-wg/yang-ver-dt/issues, tagged with 'schema-comparison'.

5. 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 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, backwards-compatible, or editorial.

* 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.

6. Generic YANG schema tree comparison algorithm

The generic schema comparison algorithm works on any YANG schema. This could be a schema associated with an individual YANG module, or a YANG schema represented by a set of modules, e.g., specified by a YANG package.

The algorithm performs a recursive tree wise comparison of two revisions of a YANG schema, with the following behavior:

- The comparison algorithm primarily acts on the parts of the schema defined by unique identifiers.

- Each identifier is qualified with the name of the module that defines the identifier.

- Identifiers in different namespaces (as defined in 6.2.1 or RFC 7950) are compared separately. E.g., 'features' are compared separately from 'identities'.

- Within an identifier namespace, the identifiers are compared between the two schema revisions by qualified identifier name. The 'renamed-from' extension allow for a meaningful comparison where the name of the identifier has changed between revisions. The 'renamed-from' identifier parameter is only used when an identifier in the new schema revision cannot be found in the old schema revision.

- YANG extensions, features, identities, typedefs are checked by comparing the properties defined by their YANG sub-statements between the two revisions.

- YANG groupings, top-level data definition statements, rpcs, and notifications are checked by comparing the top level properties defined by their direct child YANG sub-statements, and also by recursively checking the data definition statements.

- The rules specified in section 3 of [I-D.ietf-netmod-yang-module-versioning] determine whether the changes are backwards-compatible or non-backwards-compatible.

- The rules specified in section 3.2 of [I-D.ietf-netmod-yang-packages] determine whether backwards-compatible changes are 'minor' or 'editorial'.

- For YANG "description", "must", and "when" statements, the "backwards-compatible" and "editorial" extension statements can be used to mark instances when the statements have changed in a backwards-compatible or editorial way. Since by default the comparison algorithm assumes that any changes in these statements are non-backwards-compatible. XXX, more info required here, since the revisions in the module history probably need to be available for this to work in the general branched revisions case.

- Submodules are not relevant for schema comparison purposes, i.e. the comparison is performed after submodule resolution has been completed.

6.1. YANG module revision scope extension annotations

6.2. Node compatibility extension statements

In addition to the revision extension statement in [I-D.ietf-netmod-yang-module-versioning], this document defines YANG extension statements to indicate compatibility information for individual schema nodes and certain YANG statements.

The node compatibility extension statements are applicable to schema nodes (e.g. leaf, rpc, choice) as defined in [RFC7950], as well as a set of YANG statements (e.g. typedef) as listed in the YANG definition of the nbc-change-at extension in the ietf-yang-revisions module in this document.

While the top level non-backwards-compatible-revision statement is mandatory when there is a non-backwards-compatible change, the node compatibility statements are optional.

For many YANG statements, programmatic tooling can determine whether the changes to a statement between two module revisions constitutes a backwards-compatible or non-backwards-compatible change. However, for some statements, it may be impractical for tooling to determine whether the changes are backwards-compatible or not. For example, in the general case, tooling cannot determine whether the change in a YANG description statement causes a change in the semantics of a YANG schema node. If the change is to fix a typo or spelling mistake then the change can be classified as an editorial backwards-compatible change. Conversely, if the change modifies the behavioral specification of the data node then the change would need to be classified as either a non editorial backwards-compatible change or a non-backwards-compatible change. Hence, extension statements are defined to annotate a YANG module with additional information to clarify the scope of changes in cases that cannot be determined by algorithmic comparison.

Three extensions are defined for schema node compatibility information:

nbc-change-at: Indicates a specific YANG statement had a non-backwards-compatible change at a particular module or sub-module revision

bc-change-at: Indicates a specific YANG statement had a backwards-compatible change at a particular module or sub-module revision

editorial-change-at: Indicates a specific YANG statement had an editorial change at a particular module or sub-module revision. The meaning of an editorial change is as per YANG Semver [I-D.ietf-netmod-yang-semver]

When a node compatibility statement is added to a schema node in a sub-module, the revision indicated for the compatibility statement is that of the sub-module.

Adding, modifying or removing any of the node compatibility statements is considered to be a BC change.

The following example illustrates the node compatibility statements:

```
container some-stuff {
  leaf used-to-be-a-string {
    rev:nbc-change-at "3.0.0" {
      description "Changed from a string to a uint32.";
    }
    type uint32;
  }
  leaf fixed-my-description-typo {
    rev:editorial-change-at "2022-06-03";
    type string;
    description "This description used to have a typo."
  }
  list sir-changed-a-lot {
    rev:editorial-change-at "3.0.0";
    rev:bc-change-at "2.3.0";
    rev:bc-change-at "1.2.1_non_compatible";
    description "a list of stuff";
    ordered-by user;
    key "foo";
    leaf foo {
      type string;
    }
    leaf thing {
      type uint8;
    }
  }
```

Note that an individual YANG statement may have a backwards-compatible change in a revision that is non-backwards-compatible (e.g. some other node changed in a non-backwards-compatible fashion in that particular revision).

If changes are ported from one branch of YANG model revisions to another branch, care must be taken with any node compatibilty statements. A simple copy-n-paste should not be used. The node compatibilty statements may incorrectly reference a revision that is not in the history of the new revision. Further, the statements might not apply depending on what the history is like in that new branch (e.g., an NBC change that is ported might not be an NBC change in the new branch). Node compatiblity statements should not

be copied over to the new branch. Instead, the changes should be considered as completely new on the new branch, and any compatibility information should be generated from scratch.

When a node compatibility statement is present, that compatibilty statement is the authoritative classification of the backwards compatibility of the change to the schema node in the specifed revision. This allows a human author to explicitly communicate the compatibilty and potentially override the rules specified in this document. This is useful in a number of situations including:

- When a tool may not be able to accurately determine the compatibilty of a change. For example, a change in a 'pattern' or 'must' statement can be difficult for a user or tool to determine if it is a compatible change.

- When a pattern, range or other statement is changed to more correctly define the server constraint. An example is correcting a pattern that incorrectly included 355.xxx.xxx.xxx as a possible IPv4 address to make it only accept up to 255.xxx.xxx.xxx.

Nothing about the backwards compatibility of a schema node is implied by the absence of a node compatibility statement. Hence, the schema node definition must be compared between the two revisions to determine the backwards compatibility.

If any nbc-change-at extension statements exists in a module or sub-module, then the module or sub-module MUST have non-backwards-compatible-revision substatements in each revision statement of the module or sub-module history where the revision matches the argument of any nbc-change-at statements. If any revision statements are removed, then all node compatibiilty statements that reference that revision MUST also be removed. Conversely, node compatibilty statements MUST NOT be removed unless the associated revision statement in the revision history is removed.

If a node compatiblity statement is added to a grouping, then all instances where the grouping is used in the module or by an importing module are also impacted by the compatibilty information. Similarly for a 'typedef', all leafs and leaf-lists that use that typedef share the specified compatibility classification. A non-backwards-compatible change to a typedef or grouping defined in one module that is used by an importing module, does not cause the importing module to add a non-backwards-compatible-revision statement to the revision history. Non-backwards-compatible marking does not carry through import statements.

A node compatibility statement at a leaf, leaf-list, or typedef context takes precedence over a node compatibility statement in a

typedef used by the leaf, leaf-list, or typedef. If multiple typedefs with compatibility statements are used by a leaf, leaf-list, or typedef (e.g. a union), and there is no compatibility statement at the top leaf, leaf-list, or typedef context, then the order of precedence used to classify the compatibility of the top level leaf, leaf-list, or typedef is as follows: nbc-change-at, bc-change-at, and finally editorial-change-at. That is, the leaf, leaf-list, or typedef takes the most impactful change classification of all the underlying typedefs.

Node compatibility statements are not supported on YANG statements such as 'pattern' or 'range'. The compatibility statement instead goes against the leaf, leaf-list, or typedef context.

Node compatibility statements that refer to pre-release revisions of a module MUST be removed when a full release revision of the module is published.

Node compatibilty statements SHOULD NOT be used when it isn't clear which change the statement is referring to. For example: If a leaf is reordered within a container, a node compatibility statement SHOULD NOT be used against the parent container nor against the reordered leaf. Similarly, if a leaf is renamed or moved to another context without keeping the old leaf present in the model and marked obsolete, a node compatibilty statement SHOULD not be used.

7. YANG module comparison algorithm

The schema comparison algorithm defined in Section 6 can be used to compare the schema for individual modules, but with the following modifications:

- Changes to the module's metadata information (i.e. module level description, contact, organization, reference) should be checked (as potential editorial changes).

- The module's revision history should be ignored from the comparison.

- Changes to augmentations and deviations should be sorted by path and compared.

8. YANG schema comparison algorithms

8.1. Standard YANG schema comparison algorithm

The standard method for comparing two YANG schema versions is to individually compare the module revisions for each module

implemented by the schema using the algorithm defined in Section 7 and then aggregating the results together:

* If all implemented modules in the schema have only changed in an editorial way then the schema is changed in an editorial way

* If all implemented modules in the schema have only been changed in an editorial or backwards-compatible way then the schema is changed in a backwards-compatible way

* Otherwise if any implemented module in the schema has been changed in a non-backwards-compatible way then the schema is changed in a non-backwards-compatible way.

The standard schema comparison method is the RECOMMENDED scheme to calculate the version number change for new versions of YANG packages, because it allows the package version to be calculated based on changes to implemented modules revision history (or YANG semantic version number if used to identify module revisions).

8.2. Filtered YANG schema comparison algorithm

Another method to compare YANG schema, that is less likely to report inconsequential differences, is to construct full schema trees for the two schema versions, directly apply a version of the comparison algorithm defined in Section 6. This may be particular useful when the schema represents a complete datastore schema for a server because it allows various filtered to the comparison algorithm to provide a more specific answer about what changes may impact a particular client.

The full schema tree can easily be constructed from a YANG package definition, or alternative YANG schema definition.

Controlled by input parameters to the comparison algorithm, the following parts of the schema trees can optionally be filtered during the comparison:

All "grouping" statements can be ignored (after all "use" statements have been processed when constructing the schema).

All module and submodule metadata information (i.e. module level description, contact, organization, reference) can be ignored.

The comparison can be restricted to the set of features that are of interest (different sets of features may apply to each schema versions).

The comparison can be restricted to the subset of data nodes, RPCs, notifications and actions, that are of interest (e.g., the

subset actually used by a particular client), providing a more meaningful result.

The comparison could filter out backwards-compatible 'editorial' changes.

In addition to reporting the overall scope of changes at the schema level, the algorithm output can also optionally generate a list of specific changes between the two schema, along with the classification of those individual changes.

9. Comparison tooling

'pyang' has some support for comparison two module revisions, but this is currently limited to a linear module history.

TODO, it would be helpful if there is reference tooling for schema comparison.

10. Module Versioning Extension YANG Modules

YANG module with extension statements for annotating NBC changes, revision label, status description, and importing by version.

```
<CODE BEGINS> file "ietf-yang-rev-annotations@2023-02-14.yang"

module ietf-yang-rev-annotations {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-rev-annotations";
  prefix rev-ext;

  import ietf-yang-revisions {
    prefix rev;
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

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

  description
    "This YANG 1.1 module contains extensions to annotation to YANG
     module with additional metadata information on the nature of
     changes between two YANG module revisions.

     XXX, maybe these annotations could also be included in
     ietf-yang-revisions?

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

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

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

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

  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  // RFC Ed.: replace XXXX (inc above) with actual RFC number and
```

```
// remove this note.

revision 2023-03-11 {
  rev:revision-label 1.0.0-draft-ietf-netmod-yang-schema-comparison-02
  description
    "Draft revision";
  reference
    "XXXX: YANG Schema Comparison";
}

extension nbc-change-at {
  argument revision-date-or-label;
  description
    "A node compatibility statement that identifies a revision
    (by revision-label, or revision date if a revision-label is
    not available) where a non-backwards-compatible change has
    occurred in a particular YANG statement relative to the
    previous revision listed in the revision history.

    The format of the revision-label argument MUST conform to the
    pattern defined for the ietf-yang-revisions
    revision-date-or-label typedef.

    The following YANG statements MAY have zero or more
    nbc-change-at substatements:
      - all schema node statements (leaf, rpc, choice, etc)
      - 'feature' statements
      - 'grouping' statements
      - 'identity' statements
      - 'must' statements
      - 'refine' statements
      - 'typedef' statements
      - YANG extensions

    Each YANG statement MUST only a have a single node
    compatibilty statement (one of nbc-change-at, bc-change-at,
    or editorial-change-at) for a particular revision. When a node
    has more than one of the node compatibilty statements (for
    different revisions), they must be ordered from most recent
    to least recent.

    An nbc-change-at statement can have 0 or 1 'description'
    substatements.

    The nbc-change-at statement in not inherited by descendants
    in the schema tree. It only applies to the specific YANG
    statement with which it is associated.
    ";

  reference
```

```
    "XXXX: YANG Schema Comparison;
     Section XXX, XXX";

}

extension bc-change-at {
  argument revision-date-or-label;
  description
    "A node compatibility statement that identifies a revision
    (by revision-label, or revision date if a revision-label is
    not available) where a backwards-compatible change has
    occurred in a particular YANG statement relative to the
    previous revision listed in the revision history.

    The format of the revision-label argument MUST conform to the
    pattern defined for the ietf-yang-revisions
    revision-date-or-label typedef.

    The following YANG statements MAY have zero or more
    bc-change-at substatements:
      - all schema node statements (leaf, rpc, choice, etc)
      - 'feature' statements
      - 'grouping' statements
      - 'identity' statements
      - 'must' statements
      - 'refine' statements
      - 'typedef' statements
      - YANG extensions

    Each YANG statement MUST only a have a single node
    compatibilty statement (one of nbc-change-at, bc-change-at,
    or editorial-change-at) for a particular revision. When a node
    has more than one of the node compatibilty statements (for
    different revisions), they must be ordered from most recent
    to least recent.

    An bc-change-at statement can have 0 or 1 'description'
    substatements.

    The bc-change-at statement in not inherited by descendants
    in the schema tree. It only applies to the specific YANG
    statement with which it is associated.
    ";

  reference
    "XXXX: YANG Schema Comparison;
     Section XXX, XXX";

}
```

```
extension editorial-change-at {
  argument revision-date-or-label;
  description
    "A node compatibility statement that identifies a revision
    (by revision-label, or revision date if a revision-label is
    not available) where an editorial change has
    occurred in a particular YANG statement relative to the
    previous revision listed in the revision history.

    The format of the revision-label argument MUST conform to the
    pattern defined for the ietf-yang-revisions
    revision-date-or-label typedef.

    The following YANG statements MAY have zero or more
    editorial-change-at substatements:
      - all schema node statements (leaf, rpc, choice, etc)
      - 'feature' statements
      - 'grouping' statements
      - 'identity' statements
      - 'must' statements
      - 'refine' statements
      - 'typedef' statements
      - YANG extensions

    Each YANG statement MUST only a have a single node
    compatibilty statement (one of nbc-change-at, bc-change-at,
    or editorial-change-at) for a particular revision. When a node
    has more than one of the node compatibilty statements (for
    different revisions), they must be ordered from most recent
    to least recent.

    An editorial-change-at statement can have 0 or 1 'description'
    substatements.

    The editorial-change-at statement in not inherited by descendants
    in the schema tree. It only applies to the specific YANG
    statement with which it is associated.
    ";

  reference
    "XXXX: YANG Schema Comparison;
     Section XXX, XXX";

}

extension backwards-compatible {
  argument revision-date-or-label;
  description
    "Identifies a revision (by revision-label, or revision date if
     a revision-label is not available) where a
```

```
      backwards-compatible change has occurred relative to the
      previous revision listed in the revision history.

      The format of the revision-label argument MUST conform to the
      pattern defined for the ietf-yang-revisions
      revision-date-or-label typedef.

      The following YANG statements MAY have zero or more
      'rev-ext:non-backwards-compatible' statements:
          description
          must
          when

      Each YANG statement MUST only a have a single
      non-backwards-compatible, backwards-compatible, or editorial
      extension statement for a particular revision-label, or
      corresponding revision-date.";

  reference
    "XXXX: YANG Schema Comparison;
     Section XXX, XXX";
}

extension editorial {
  argument revision-date-or-label;
  description
    "Identifies a revision (by revision-label, or revision date if
     a revision-label is not available) where an editorial change
     has occurred relative to the previous revision listed in the
     revision history.

     The format of the revision-label argument MUST conform to the
     pattern defined for the ietf-yang-revisions
     revision-date-or-label typedef.

     The following YANG statements MAY have zero or more
     'rev-ext:non-backwards-compatible' statements:
         description

     Each YANG statement MUST only a have a single
     non-backwards-compatible, backwards-compatible, or editorial
     extension statement for a particular revision-label, or
     corresponding revision-date.";

  reference
    "XXXX: YANG Schema Comparison;
     Section XXX, XXX";
}

extension renamed-from {
```

```
    argument yang-identifier;
    description
      "Specifies a previous name for this identifier.

       This can be used when comparing schema to optimize handling
       for data nodes that have been renamed rather than naively
       treated them as data nodes that have been deleted and
       recreated.

       The argument 'yang-identifier' MUST take the form of a YANG
       identifier, as defined in section 6.2 of RFC 7950.

       Any YANG statement that takes a YANG identifier as its
       argument MAY have a single 'rev-ext:renamed-from'
       sub-statement.

       TODO, we should also facilitate identifiers being moved into
       other modules, e.g. by supporting a module-name qualified
       identifier.";

    reference
      "XXXX: YANG Schema Comparison;
       Section XXX, XXX";
  }
}

<CODE ENDS>
```

11. 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.

12. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

13. IANA Considerations

13.1. YANG Module Registrations

The following YANG module is requested to be registered in the "IANA Module Names" registry:

The ietf-yang-rev-annotations module:

Name: ietf-yang-rev-annotations

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-rev-annotations

Prefix: rev-ext

Reference: [RFCXXXX]

14. References

14.1. 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-08, 12 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-versioning-08>.

[I-D.ietf-netmod-yang-packages] Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-03, 4 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-packages-03>.

[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-10, 17 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-semver-10>.

[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-07, 10 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-versioning-reqs-07>.

[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>.

[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>.

[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>.

[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>.

14.2. 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)
Cisco Systems, Inc.

Email: perander@cisco.com

Robert Wilton
Cisco Systems, Inc.

Email: rwilton@cisco.com