NETMOD Working Group
Internet-Draft
Intended status: Informational
Expires: 19 July 2026

R. T. Venkateswaran
S. V. G. Karnati
S. Jain
V. Ramamoorthy
V. H. Nagamangalam
Cisco Systems
15 January 2026

Enhancements to the YANG Language for Capturing Subtree Replacements
draft-rtv-netmod-yang-subtree-replacement-02

Abstract

As YANG data models evolve over time, model nodes are often deprecated or made obsolete. Current practices for documenting replacement paths for these nodes rely on unstructured external documents, making it difficult to programmatically identify and migrate to replacement nodes. This document proposes a YANG extension mechanism that embeds replacement path information directly within YANG models, enabling automation tools to identify replacement nodes and assist users in migrating from deprecated elements to their replacements.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://datatracker.ietf.org/doc/draft-rtv-netmod-yang-subtree-replacement/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-rtv-netmod-yang-subtree-replacement/.

Discussion of this document takes place on the NETMOD Working Group mailing list (mailto:netmod@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/netmod/. Subscribe at https://www.ietf.org/mailman/listinfo/netmod/.

Source for this draft and an issue tracker can be found at https://github.com/rajesh-rtv/yang-replacement.

Status of This Memo

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

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC7950] is a data modeling language used to define the configuration and operational data of network devices. As network technologies evolve, some nodes within YANG modules become deprecated or obsolete. Operators then need a reliable way to discover which schema nodes supersede the deprecated ones so they can migrate configuration and operational tooling.

The YANG language provides the status statement to mark nodes as deprecated or obsolete, but it offers no standardized mechanism to point to the successor node(s). Current practice relies on unstructured documents, spreadsheets, or ad-hoc emails. The goal of this document is to embed that knowledge directly in the YANG modules so that both humans and automation can infer replacements programmatically.

The term "replacement" in this document refers to guidance that accompanies deprecated or obsolete nodes. The intent is not to redefine YANG status values, but to provide structured metadata that helps operators and tooling identify where to migrate.

2. Problem Statement

The reviewed material highlighted that today the only way to communicate replacement paths for deprecated nodes is through separate, unstructured documents like CSV files or emails. This approach suffers from several significant drawbacks:

* *Inefficiency*: Network operators must manually search through external documentation to find replacement paths, often requiring correlation between multiple documents.

* *Error-prone*: External documentation can become outdated or may not accurately reflect the current node structure and paths defined in the YANG files, leading to incorrect migrations.

* *Lack of automation*: Without a standardized, machine-readable way to express replacement information, tools cannot programmatically identify replacement nodes or assist users in migration.

* *Documentation fragmentation*: Replacement information becomes scattered across multiple documents, making it difficult to maintain a complete view of model evolution.

These challenges are particularly acute for large-scale network operators who must manage configuration across numerous devices with diverse YANG models, and for vendors who need to support customers through model transitions.

It’s worth noting that this functionality has been proposed as a potential enhancement to a future version of the YANG language itself ([YANG-NEXT]). However, developing and standardizing a new version of YANG would likely take a long while. The solution proposed in this document is designed to be standardized quickly and used with existing YANG modules and infrastructure.

During IETF 123, the NETMOD chairs polled the working group on whether the problem statement in this document should be pursued. The response showed strong interest in solving the problem, while also signalling that additional clarification and tooling guidance are required before adoption. This revision keeps the problem statement separate from the proposed solution so the working group can iterate on each aspect independently.

3. Solution

The reviewed copy initially described a new keyword named replacement. During working group discussion we agreed that the same behaviour can be achieved with an extension, avoiding changes to the base language. We therefore define a reusable extension that can be attached to any node marked status deprecated or status obsolete. This keeps the proposal backward compatible and aligns with the guidance in [RFC8407] for extending YANG functionality without modifying the core language.

3.1. Proposed Extension Mechanism

Specifically, we introduce two custom extensions, ietf-ext:deprecation-info and ietf-ext:obsoletion-info. Authors use the former while a node remains status deprecated and the latter once it transitions to status obsolete. Either extension can appear multiple times on the same node so that each occurrence represents a migration tuple. This satisfies the request from the NETMOD working group to represent successor paths as structured tuples instead of opaque strings and enables the expression of one-to-many or many-to-one relationships.

Each extension statement carries a tuple encoded as key/value pairs that follow the ABNF in Figure 1. The tuple minimally identifies the _path-type_ (absolute or relative) and the _path-target_ of the replacement node. Optional qualifiers allow the publisher to flag migration hints (for example priority or free-form notes) without inventing separate extensions.

```
lifecycle-entry = path-type ":" path-target *( ";" kv-pair )
path-type         = "absolute" / "relative" / "none"
path-target       = 1*(%x21-7E)
kv-pair           = key "=" value
key               = ALPHA *( ALPHA / DIGIT / "-" )
value             = 1*(%x20-7E)
```

Figure 1: Lifecycle Tuple ABNF

When path-type is absolute, path-target is an absolute-schema-nodeid that resolves from the module root. When path-type is relative, the tuple is interpreted relative to the schema node that instantiates the grouping containing the deprecated node. When path-type is none, the publisher MUST set path-target to the literal none to indicate that no direct replacement exists.

The optional key/value qualifiers are intentionally open-ended so tooling can evolve. Two keys are RECOMMENDED:

* *state=deprecated|obsolete*: Identifies whether the tuple accompanies a deprecated or obsolete node. If omitted, consumers SHOULD assume the tuple inherits the YANG status of the enclosing node.

* *note=.../priority=.../status=preferred*: Convey migration hints such as preferred replacements, operational notes, or removal timelines.

The benefits of this approach include:

* *Backward compatibility*: The extension approach doesn’t require changes to the YANG language itself and is compatible with existing tools.

* *Self-documenting models*: Replacement information is embedded directly in the models, eliminating the need for external documentation.

* *Automation enablement*: Tools can programmatically identify replacement paths and assist with migration.

* *Improved developer experience*: YANG model developers and users gain clear guidance for transitions between deprecated and replacement nodes.

* *Support for complex replacements*: The tuple format and ability to repeat ietf-ext:deprecation-info or ietf-ext:obsoletion-info statements allow one-to-many and many-to-one replacements, including cases that cross module boundaries.

3.2. Allowed Placements

The ietf-ext:deprecation-info and ietf-ext:obsoletion-info extensions MAY be applied to the following YANG statements: container, list, leaf, leaf-list, choice, case, anydata, and anyxml. These extensions MUST NOT appear on RPC, action, or notification statements.

Whenever either extension is used, the enclosing schema node MUST also carry the matching lifecycle state. Use status deprecated; with ietf-ext:deprecation-info and status obsolete; with ietf-ext:obsoletion-info. Tooling MUST treat the absence of a matching status statement as an error.

Publishers MAY repeat the extensions multiple times on the same schema node. Each occurrence conveys a distinct tuple and tooling SHOULD process them in the order they appear if prioritisation is required.

Feedback from Kent Watsen noted that the earlier naming focused heavily on the replacement node. Using lifecycle-oriented extension names and normatively tying them to the existing status statements makes the tuples an explicit companion to the lifecycle metadata instead of a standalone hint.

3.3. Design-Time Evaluation Rationale

During the IETF 123 NETMOD discussion, implementers emphasized that the ability to reason about replacements at design time is critical. Embedding the tuples directly in the YANG source enables:

* *Model reviews*: Working group and vendor reviewers can confirm that every deprecated node is paired with an up-to-date replacement before publishing modules.

* *Toolchain validation*: Compilers and CI pipelines can flag missing or malformed tuples while the module is being authored instead of deferring to deployment time.

* *Automation planning*: Operators and orchestrators can evaluate the impact of upcoming deprecations without requiring device connectivity.

This design-time emphasis was highlighted by Rob Wilton and others, and aligns with the guidance in [RFC8407] on providing migration help inside the model itself.

3.4. Extension Versus Instance Data

Some participants asked why the information cannot be carried as instance data. Instance documents are unsuitable for two reasons:

* *Portability*: Instance data is tied to a particular deployment, whereas the tuples must travel with the module revisions that introduce the deprecation.

* *Tool coverage*: Many downstream tools (IDEs, code generators, documentation sites) consume YANG modules only. Requiring the retrieval of separate instance documents would exclude those workflows.

Embedding the tuples as a standard extension therefore provides a single source of truth that can be validated and reasoned about during module design.

3.5. Handling Obsolete Nodes

Several YANG modules deprecate nodes for one or more releases before ultimately marking them status obsolete. Authors SHOULD switch from ietf-ext:deprecation-info to ietf-ext:obsoletion-info when the state changes so the metadata continues to reflect the migration path. The state qualifier remains available for tooling that prefers an explicit flag (for example state=obsolete).

When a node is marked obsolete with no successor, the tuple MUST use path-type = none and path-target = none. Tooling can then surface a precise warning while recognising that there is no automated migration target.

3.6. Related Work and Alternatives

The NETMOD working group also noted other approaches under discussion, including model-specific conventions, instance-data catalogues, and proposals for future YANG language revisions (often referred to collectively as “YANG Next”). This document positions the extension as an immediately deployable option that complements those efforts. Publishers can still provide supplemental instance data where useful, but the extension ensures there is a normative pointer available inside the module itself.

4. YANG Module

This section defines the normative YANG module that implements the lifecycle extensions described in this document.

4.1. ietf-lifecycle-ext.yang module

The ietf-lifecycle-ext module defines the deprecation-info and obsoletion-info extensions that enable authors to embed replacement information directly within YANG modules.

```
<CODE BEGINS> file "ietf-lifecycle-ext@2026-01-15.yang"
module ietf-lifecycle-ext {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-lifecycle-ext";
  prefix ietf-ext;

  organization "IETF NETMOD Working Group";

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

```
        Editor:   Rajesh Tarakkad Venkateswaran
                  <mailto:rtv@cisco.com>";

      description
        "This module defines lifecycle extensions for recording migration
         metadata for deprecated and obsolete YANG nodes.

         Copyright (c) 2026 IETF Trust and the persons identified as
         authors of the code.  All rights reserved.

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

         This version of this YANG module is part of RFC XXXX
         (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.";

      revision 2026-01-15 {
        description
          "Second revision. Added CODE BEGINS markers for YANG validation.
           Restructured sections to separate normative module from examples.";
        reference "draft-rtv-netmod-yang-subtree-replacement-02";
      }

      revision 2025-12-28 {
        description "Initial revision.";
        reference "draft-rtv-netmod-yang-subtree-replacement-01";
      }

      extension deprecation-info {
        argument "tuple";
        description
          "Each use of this extension attaches a lifecycle-entry tuple as
           defined in the main body of this document. This extension is intended
           for nodes that remain status deprecated.";
      }
      extension obsoletion-info {
        argument "tuple";
```

```
      description
        "Each use of this extension attaches the same lifecycle-entry tuple
         but is intended for nodes that have become status obsolete.";
    }
  }
  <CODE ENDS>
```

5. Implementation of the Lifecycle Tuples Across Various Scenarios

The following sections demonstrate how the lifecycle extensions can be applied across different replacement scenarios. The examples illustrate common patterns encountered when evolving YANG models and show how the tuples provide clear migration paths for each situation.

5.1. Path Reference Types

To accommodate different structural relationships between deprecated nodes and their replacements, the reviewed copy distinguished between absolute and relative XPath expressions. The tuple grammar in this document encodes the same information in the path-type field:

* *absolute*: Used when the replacement node can be directly referenced with an absolute schema path from the YANG module root. This is the most common case for nodes inside standard containers and lists.

* *relative*: Used when the replacement node appears in multiple places, such as within groupings or augments. The path is resolved relative to the instantiation of the grouping that contains the deprecated node.

* *none*: Used to signal that no direct replacement exists. Tooling can still present the tuple together with human-readable notes supplied through key/value qualifiers.

Each path reference type has specific syntax requirements and use cases, which are illustrated in the scenarios that follow. When a node is deprecated without a direct successor, the tuple uses path-type = none with path-target = none, matching the reviewed copy' s None token.

5.2. Case 1: Simple Node with Replacement

Description: a node is deprecated and has a replacement node in the same container.

YANG Example:

```
   ietf-ext:deprecation-info "absolute:/{File_name}:{abs_path}";
```

5.3. Case 2: Node Deprecated with No Replacement

Description: a node is deprecated and there is no direct replacement node.

YANG Example:

```
ietf-ext:deprecation-info "none:none;note=manual-migration";
```

5.4. Case 3: Grouping Cases

Identifying XPath nodes within YANG models, particularly when dealing with groupings, presents a challenge. Nodes that are imported through grouping may not be easily pinpointed using absolute XPath. The solution is to uniquely identify the grouping using the file name and grouping name, then provide a relative XPath from the top level of the grouping. In all other cases, an absolute XPath can be provided.

Utilize the file name and grouping name to uniquely identify the grouping within the YANG model. Provide a relative XPath from the top level of the grouping to pinpoint the node.

This approach ensures that nodes within groupings can be accurately identified and referenced, while maintaining clarity and precision for nodes outside of groupings.

5.4.1. Sub-Case 1: Node Deprecated Outside Grouping, Replacement Inside Grouping

Description: a node that is deprecated outside a grouping structure but has a replacement node within a specific grouping.

YANG Example:

```
ietf-ext:deprecation-info "relative:/{File_name}:{grouping_name}/{rel_path_inside_grou
ping}";
```

5.4.2. Sub-Case 2: Node Deprecated Inside Grouping, Replacement Outside in a non-group

Description: a node that is deprecated within a grouping structure but has a replacement node outside any grouping.

YANG Example:

```
ietf-ext:deprecation-info "absolute:/{File_name}:{abs_path}";
```

5.4.3. Sub-Case 3: Node Deprecated Inside Grouping, Replacement Inside Existing/New Grouping

Description: a node that is deprecated within a grouping structure and has a replacement node within the same or a new grouping structure.

YANG Example:

```
ietf-ext:deprecation-info "relative:/{File_name}:{grouping_name}/{rel_path_inside_grou
ping}";
```

5.4.4. Sub-Case 4: Node Deprecated Outside Grouping, Replacement Outside Grouping

Description: a node that is deprecated outside a grouping structure and has a replacement node also outside any grouping.

YANG Example:

```
ietf-ext:deprecation-info "absolute:/{File_name}:{abs_path}";
```

5.5. Summary of Syntax Notations

The following syntax notations are used for replacement paths:

* relative:/{File_name}:{grouping_name}/{rel_path_inside_grouping}: Indicates the relative path of the replacement node within a specified grouping in a file.

* absolute:/{File_name}:{path}: Indicates the absolute path of the replacement node in a file.

* none:none: Indicates that there is no replacement for the deprecated node; publishers can append notes (for example none:none;note=legacy).

5.6. Replacement Guidelines for Non-Leaf level Deprecation

When deprecating structures like container or list at their respective levels, it is essential to ensure that the replacement is explicitly mentioned at all sub-levels, including child elements such as leaf, leaf-list, or nested structures. This approach ensures clarity and provides a complete mapping of deprecated elements to their replacements, making it easier for users to transition to the new structure.

YANG Example:

```
   container old-container {
     status deprecated;
     ietf-ext:deprecation-info "absolute:/{File_name}:{replacement_container_path}";

     leaf old-leaf {
       status deprecated;
       ietf-ext:deprecation-info "absolute:/{File_name}:{replacement_leaf_path}";
     }
   }
```

6. Example Implementation

The following examples demonstrate how the lifecycle extensions can be used in practice. These are vendor-neutral examples created specifically for this document to illustrate the functionality and are not intended to be actual YANG modules used in production environments.

6.1. ietf-deprecation-regression-test-17131.yang module

YANG Example:

```
   module ietf-deprecation-regression-test-17131 {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-deprecation-regression-test";
     prefix depr-reg-test;

   import ietf-depr-reg-test-helper-17131 {
       prefix depr-reg-test-helper;
     }

   import ietf-lifecycle-ext {
       prefix ietf-ext;
     }

   container deprecation-regression-test {
       container configurations {
         container config {


       // Case 1: Replacement in same container.
       leaf case1 {
         type uint8;
         status deprecated;
         description
           "Case 1 - Replacement in same container, "
           "leaf case1 (DEPRECATED)";
         ietf-ext:deprecation-info
```

```
          "absolute:/ietf-deprecation-regression-test-17131:"
          "deprecation-regression-test/configurations/config/"
          "case1-replacement;status=preferred";
        ietf-ext:deprecation-info
          "absolute:/ietf-deprecation-regression-test-17131:"
          "deprecation-regression-test/configurations/config/"
          "case1-legacy;note=legacy-support";
      }
      leaf case1-replacement {
        type uint16;
        description
          "Case 1 - Replacement in same container, "
          "leaf case1-replacement (NEW)";
      }

      // Case 2: Replacement in different container.
      leaf case2 {
        type uint8;
        status deprecated;
        description
          "Case 2 - Leaf deprecated with a replacement in a different "
          "container, leaf i (DEPRECATED)";
        ietf-ext:deprecation-info
          "absolute:/ietf-deprecation-regression-test-17131:"
          "deprecation-regression-test/configurations/"
          "replacementContainerCase2/case2-replacement";
      }

      // Case 3: Replacement located in another module.
      leaf case3 {
        type uint8;
        status deprecated;
        description
          "Case 3 - Replacement located in another module, "
          "leaf case3 (DEPRECATED)";
        ietf-ext:deprecation-info
          "absolute:/ietf-deprecation-regression-test-file-2-17131:"
          "deprecation-regression-test-2/configurations/config/"
          "case3-replacement;note=external-module";
      }

      // Case 4: List leaf replaced elsewhere.
      list listCase4 {
        key "key-case4";
        leaf key-case4 {
          type string;
        }
        leaf case4 {
```

```
          type uint8;
          status deprecated;
          description
            "Case 4 - List leaf replaced in another list, "
            "leaf case4 (DEPRECATED)";
          ietf-ext:deprecation-info
            "absolute:/ietf-deprecation-regression-test-17131:"
            "deprecation-regression-test/configurations/"
            "replacementContainerCase4/case4-replacement";
        }
      }
      list replacementListCase4 {
        key "key-case4-replacement";
        leaf key-case4-replacement {
          type string;
        }
        leaf case4-replacement {
          type uint8;
          description
            "Case 4 - List leaf replacement, "
            "case4-replacement (NEW)";
        }
      }

      // Case 5: Grouping leaf replaced inside grouping.
      grouping groupCase5 {
        leaf case5 {
          type uint8;
          status deprecated;
          description
            "Case 5 - Grouping leaf deprecated and reused via the same "
            "grouping, leaf case5 (DEPRECATED)";
          ietf-ext:deprecation-info
            "relative:/ietf-deprecation-regression-test-17131:"
            "groupCase5/case5-replacement;status=preferred";
          ietf-ext:deprecation-info
            "absolute:/ietf-deprecation-regression-test-17131:"
            "deprecation-regression-test/configurations/config/"
            "case5-replacement-alt;note=alternate-path";
        }
        leaf case5-replacement {
          type uint16;
          description
            "Case 5 - Replacement leaf reused via grouping, "
            "case5-replacement (NEW)";
        }
      }
      container containerP1 {
```

```
      uses groupCase5;
      description
        "Case 5 - Grouping usage via containerP1 (NEW)";
    }
    container containerP2 {
      uses groupCase5;
      description
        "Case 5 - Grouping usage via containerP2 (NEW)";
    }

    // Case 6: Grouping leaf replaced via imported helper.
    container containerP3 {
      uses depr-reg-test-helper:groupCase6;
      description
        "Case 6 - Grouping reuse via helper module, containerP3 (NEW)";
    }
    container containerP4 {
      uses depr-reg-test-helper:groupCase6;
      description
        "Case 6 - Grouping reuse via helper module, containerP4 (NEW)";
    }

    // Case 7: Replacement leaf shares name.
    leaf case7 {
      type uint8;
      status deprecated;
      description
        "Case 7 - Leaf replaced by another leaf with the same name "
        "elsewhere, leaf case7 (DEPRECATED)";
      ietf-ext:deprecation-info
        "absolute:/ietf-deprecation-regression-test-17131:"
        "deprecation-regression-test/configurations/"
        "replacementContainerCase7/case7";
    }
    container replacementContainerCase7 {
      presence "true";
      description
        "Case 7 - Replacement container holding leaf case7 (NEW)";
      leaf case7 {
        type uint8;
        description
          "Case 7 - New leaf case7 at alternate location (NEW)";
      }
    }

    // Case 8 : Leaf-list deprecated, replaced by another leaf-list
    leaf-list case8 {
      type uint8;
```

```
        status deprecated;
        description
          "Case 8 - Leaf-list case8 replaced by new list (DEPRECATED)";
        ietf-ext:deprecation-info
          "absolute:/ietf-deprecation-regression-test-17131:"
          "deprecation-regression-test/configurations/config/"
          "case8-replacement";
      }
      leaf-list case8-replacement {
        type uint8;
        description
          "Case 8 - Replacement leaf-list case8-replacement (NEW)";
      }

      // Case 9 : Empty leaf deprecated, replaced by another empty leaf
      leaf case9 {
        type empty;
        status deprecated;
        description
          "Case 9 - Empty leaf case9 replaced by new empty leaf "
          "(DEPRECATED)";
        ietf-ext:deprecation-info
          "absolute:/ietf-deprecation-regression-test-17131:"
          "deprecation-regression-test/configurations/config/"
          "case9-replacement";
      }
      leaf case9-replacement {
        type empty;
        description
          "Case 9 - Replacement empty leaf case9-replacement (NEW)";
      }

      // Case 10 : A container is deprecated with a replacement container.
      container containerCase10 {
        leaf case10 {
          type uint8;
          status deprecated;
          description
            "Case 10 - Leaf case10 in deprecated container (DEPRECATED)";
          ietf-ext:deprecation-info
            "absolute:/ietf-deprecation-regression-test-17131:"
            "deprecation-regression-test/configurations/config/"
            "replacementContainerCase10/case10-replacement";
        }
        status deprecated;
        description
          "Case 10 - ContainerCase10 replaced by successor (DEPRECATED)";
        ietf-ext:deprecation-info
```

```
          "absolute:/ietf-deprecation-regression-test-17131:"
          "deprecation-regression-test/configurations/config/"
          "replacementContainerCase10";
      }
      container replacementContainerCase10 {
        leaf case10-replacement {
          type uint8;
          description
            "Case 10 - Replacement leaf case10-replacement (NEW)";
        }
        description
          "Case 10 - Replacement container replacementContainerCase10 "
          "(NEW)";
      }

      // Case 11: Multiple items mapped to one leaf.
      leaf case11a {
        type uint8;
        status deprecated;
        description
          "Case 11 - Items merged into leaf case11a (DEPRECATED)";
        ietf-ext:deprecation-info
          "absolute:/ietf-deprecation-regression-test-17131:"
          "deprecation-regression-test/configurations/config/"
          "case11-replacement";
      }
      leaf case11b {
        type uint8;
        status deprecated;
        description
          "Case 11 - Items merged into leaf case11b (DEPRECATED)";
        ietf-ext:deprecation-info
          "absolute:/ietf-deprecation-regression-test-17131:"
          "deprecation-regression-test/configurations/config/"
          "case11-replacement";
      }
      leaf case11-replacement {
        type uint16;
        description
          "Case 11 - Replacement leaf case11-replacement (NEW)";
      }

      // Case 12: List item replaced by different list type.
      list listCase12 {
        key "key-case12";
        status deprecated;
        leaf key-case12 {
          type string;
```

```
          status deprecated;
          ietf-ext:deprecation-info
            "absolute:/ietf-deprecation-regression-test-17131:"
            "deprecation-regression-test/configurations/config/"
            "replacementListCase12/key-case12-replacement";
        }
        leaf case12 {
          type uint8;
          status deprecated;
          ietf-ext:deprecation-info
            "absolute:/ietf-deprecation-regression-test-17131:"
            "deprecation-regression-test/configurations/config/"
            "replacementListCase12/case12-replacement";
        }
      }
      list replacementListCase12 {
        key "key-case12-replacement";
        leaf key-case12-replacement {
          type string;
        }
        leaf case12-replacement {
          type uint8;
        }
      }

      // Case 13: Choice case provides alternate leaf.
      choice choice13 {
        case caseCase13 {
          leaf case13 {
            type uint8;
            status deprecated;
            description
              "Case 13 - Choice branch leaf case13 (DEPRECATED)";
            ietf-ext:deprecation-info
              "absolute:/ietf-deprecation-regression-test-17131:"
              "deprecation-regression-test/configurations/config/"
              "choice13/case13-replacement";
          }
        }
        case caseCase13Replacement {
          leaf case13-replacement {
            type uint8;
            description
              "Case 13 - Alternate choice leaf case13-replacement (NEW)";
          }
        }
      }
```

```
       // Case 14: One leaf replaced by multiple new leaves.
       leaf case14 {
         type uint16;
         status deprecated;
         description
           "Case 14 - Single leaf replaced by many, case14 (DEPRECATED)";
         ietf-ext:deprecation-info
           "absolute:/ietf-deprecation-regression-test-17131:"
           "deprecation-regression-test/configurations/config/"
           "case14a-replacement";
         ietf-ext:deprecation-info
           "absolute:/ietf-deprecation-regression-test-17131:"
           "deprecation-regression-test/configurations/config/"
           "case14b-replacement";
       }
         leaf case14a-replacement {
           type uint8;
           description
           "Case 14 - Replacement leaf case14a-replacement (NEW)";
       }
         leaf case14b-replacement {
           type uint8;
           description
           "Case 14 - Replacement leaf case14b-replacement (NEW)";
       }

       // Case 15: Leaf obsoleted with no replacement.
       leaf case15 {
         type uint8;
         status obsolete;
         description
           "Case 15 - Feature retired; no replacement (OBSOLETE)";
         ietf-ext:obsoletion-info
           "none:none;note=manual-migration";
       }

       // Case 16: Container replacement annotated per level.
       container containerCase16 {
         presence "true";
         status deprecated;
         description
           "Case 16 - ContainerCase16 replaced by successor container "
           "(DEPRECATED)";
         ietf-ext:deprecation-info
           "absolute:/ietf-deprecation-regression-test-17131:"
           "deprecation-regression-test/configurations/config/"
           "replacementContainerCase16";
```

```
        leaf case16-leaf1 {
          type string;
          status deprecated;
          description
            "Case 16 - Leaf case16-leaf1 deprecated in containerCase16 "
            "(DEPRECATED)";
          ietf-ext:deprecation-info
            "absolute:/ietf-deprecation-regression-test-17131:"
            "deprecation-regression-test/configurations/config/"
            "replacementContainerCase16/case16a";
        }

        leaf case16-leaf2 {
          type uint8;
          status deprecated;
          description
            "Case 16 - Leaf case16-leaf2 deprecated in containerCase16 "
            "(DEPRECATED)";
          ietf-ext:deprecation-info
            "absolute:/ietf-deprecation-regression-test-17131:"
            "deprecation-regression-test/configurations/config/"
            "replacementContainerCase16/case16b";
        }
      }

      container replacementContainerCase16 {
        presence "true";
        description
          "Case 16 - Replacement container replacementContainerCase16 "
          "(NEW)";

        leaf case16a {
          type string;
          description
            "Case 16 - Replacement for case16-leaf1 (NEW)";
        }

        leaf case16b {
          type uint8;
          description
            "Case 16 - Replacement for case16-leaf2 (NEW)";
        }
      }
    }
  }   } }
```

6.2. ietf-deprecation-regression-test-helper-module-17131.yang module

YANG Example:

```
module ietf-depr-reg-test-helper-17131 {
  yang-version 1.1;

namespace "urn:ietf:params:xml:ns:yang:ietf-depr-reg-test-helper";

prefix depr-reg-test-helper;

organization "IETF NETMOD Working Group";

contact "IETF NETMOD Working Group <netmod@ietf.org>";

import ietf-lifecycle-ext {
    prefix ietf-ext;
  }

// Case 6 : Leaf inside a grouping deprecated, replaced in the same grouping but impor
ted through
   // a different module and used in various modules.
   grouping groupCase6 {
     leaf case6 {
       type uint8;
       default 10;
       status deprecated;
       description
         "Case 6 - Leaf inside a grouping
          deprecated, replaced in the same grouping but imported through
          a different module and used in various modules,
          leaf case-6 (DEPRECATED)";
       ietf-ext:deprecation-info
         "relative:/ietf-depr-reg-test-helper-17131:"
         "groupCase6/case6-replacement;status=preferred";
     }
     leaf case6-replacement {
       type uint8;
       default 11;
       description
         "Case 6 - Leaf inside a grouping
          deprecated, replaced in the same grouping but imported
          through a different module and used in various modules,
          leaf case6-replacement (NEW)";
     }
   }
}
```

6.3. ietf-deprecation-regression-test-file-2-17131.yang module

YANG Example:

```
module ietf-deprecation-regression-test-file-2-17131 {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-deprecation-regression-test-file-2";
  prefix depr-reg-test-2;

import ietf-lifecycle-ext {
    prefix ietf-ext;
  }

container deprecation-regression-test-2 {
    container configurations {
      container config {
        // Case 3: Replacement located in another module.
        leaf case3-replacement {
          type uint8;
          default 15;
        }
      }
    }
  }
}
```

7. IETF 123 NETMOD Feedback

This document incorporates feedback received during the NETMOD working group meeting at IETF 123 (https://notes.ietf.org/notes-ietf-123-netmod?both). Key discussion points included:

* *Kent Watsen*: Requested that replacement metadata be modelled as a tuple rather than an opaque string and questioned the naming. The current revision satisfies this by defining a structured tuple and restricting usage to deprecated/obsolete nodes.

* *Lou Berger*: Asked that the working group agree on the problem statement before merging solution details. This draft keeps a dedicated problem statement section and clearly labels solution content.

* *Balazs Lengyel*: Noted that other approaches may exist. Section Section 3.6 summarises related efforts and how they complement this proposal.

* *Reshad Rahman*: Asked whether the extension applies beyond leaf nodes. Section Section 3.2 now defines the permitted statement types.

* *Deepak Rajaram*: Questioned why an extension is needed instead of instance data. Section Section 3.4 provides the rationale.

* *Rob Wilton*: Highlighted the importance of design-time evaluation, addressed in Section Section 3.3.

The chair poll (“Should the problem on slide 2 be addressed by the WG?”) indicated broad interest, with several participants asking for more detail and none objecting. This revision documents the poll outcome and adds guidance for follow-on tooling work.

8. Conventions and Definitions

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.

9. Operational Considerations

Network operators and YANG model consumers can leverage the information provided by the ietf-ext:deprecation-info extension in several ways:

* *Automated Migration Tools*: Software tools can be developed to scan configurations using deprecated nodes and automatically suggest or implement replacements.

* *Documentation Generation*: Model documentation tools can highlight deprecated nodes and their replacements, making it easier for network operators to understand migration paths.

* *Configuration Validation*: Validation tools can warn about the use of deprecated nodes and suggest alternatives based on the extension data.

* *Training and Knowledge Transfer*: The explicit documentation of replacements can help in training and knowledge transfer as teams adopt newer model versions.

10. Security Considerations

The extension defined in this document does not introduce new protocol behaviour or data plane interactions. The tuples are metadata that accompany YANG modules and are evaluated at design time. Consequently, the security considerations of [RFC7950] apply.

Incorrect or stale tuples could mislead automation systems or operators during migrations. Publishers SHOULD validate tuples during module reviews and CI testing, and consumers SHOULD treat the tuples as advisory rather than authoritative until verified in their environments.

The tuples do not carry sensitive information beyond schema paths. They neither relax access control nor expose configuration data. Implementations SHOULD ensure that tuple parsing failures are handled gracefully (for example by logging warnings) to avoid denial-of-service conditions caused by malformed metadata.

11. IANA Considerations

IANA is requested to add a new entry to the “IETF XML Registry” for the namespace urn:ietf:params:xml:ns:yang:ietf-lifecycle-ext with reference to this document.

IANA is also requested to add a new entry to the “YANG Module Names” registry with the following values: name ietf-lifecycle-ext, namespace urn:ietf:params:xml:ns:yang:ietf-lifecycle-ext, prefix ietf-ext, reference this document.

No other IANA actions are required.

12. References

12.1. Normative References

[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/rfc/rfc2119>.

[RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/rfc/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/rfc/rfc8174>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/rfc/rfc8407>.

12.2. Informative References

[YANG-NEXT] "YANG Next Issue #130: Deprecation Replacement Metadata", GitHub netmod-wg/yang-next#130, 2024, <https://github.com/netmod-wg/yang-next/issues/130>.

Acknowledgments

The authors would like to thank the members of the NETMOD working group for their valuable input and feedback.

The authors acknowledge the ongoing YANG Next discussions captured in [YANG-NEXT] that provided initial inspiration for this proposal.

Authors' Addresses

Rajesh Tarakkad Venkateswaran
Cisco Systems
Email: rtv@cisco.com

Sai Venkata Giri Karnati
Cisco Systems
Email: saikarna@cisco.com

Sarthak Jain
Cisco Systems
Email: sarthakj@cisco.com

Veena Ramamoorthy
Cisco Systems
Email: vemoorth@cisco.com

Venkata Harish Nagamangalam
Cisco Systems
Email: vnagaman@cisco.com