NETMOD Working Group
Internet-Draft
Intended status: Informational
Expires: 1 July 2026

R. T. Venkateswaran
S. V. G. Karnati
S. Jain
V. Ramamoorthy
V. H. Nagamangalam
Cisco Systems
28 December 2025

Enhancements to the YANG Language for Capturing Subtree Replacements
draft-rtv-netmod-yang-subtree-replacement-01

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.

This Internet-Draft will expire on 1 July 2026.

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

Currently, when YANG model nodes are deprecated or obsoleted, the information about replacement nodes is typically documented in separate, unstructured documents such as CSV files, release notes, 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.

At IETF 123, the NETMOD working group expressed strong interest in addressing this problem statement with requests for additional clarification on the solution approach.

3. Solution

To address the challenges associated with deprecated or obsolete nodes in YANG models, we propose an extension-based approach that embeds replacement information directly within the YANG models themselves. This approach follows the recommendations in [RFC8407] for extending YANG functionality without modifying the core language.

3.1. Proposed Extension Mechanism

Specifically, we introduce a custom extension, ietf-depr:deprecated-info, that authors attach to nodes marked status deprecated. This extension carries a structured tuple that documents the migration path during the deprecation window—the critical period when users are expected to transition to replacement nodes. This approach represents successor paths as structured tuples instead of opaque strings and enables the expression of one-to-many relationships through comma-separated paths within a single tuple.

Importantly, this extension is designed for the *deprecation phase only*. When a node transitions from status deprecated to status obsolete, the extension MUST be removed. This design decision addresses a critical challenge in long-lived YANG modules: industry best practices (RFC 8407, BBF OD-360) mandate that obsolete nodes remain in modules indefinitely for backward compatibility. In large-scale production environments with hundreds or thousands of deprecated nodes across module portfolios, keeping structured metadata on obsolete nodes would result in permanent, ever-growing clutter with minimal benefit, since automated migration should occur during the deprecation window.

Each extension statement carries a structured tuple that follows the ABNF in Figure 1. The tuple consists of one or more path entries, where each entry specifies a _path-type_ (absolute, relative, or none) and its corresponding _path-target_. When multiple path entries are provided as a comma-separated list, automated migration tools MUST process all paths, treating them as a one-to-many split of functionality where each target is mandatory for complete migration. Each path entry in a comma-separated list may use a different path-type as needed.

```
lifecycle-entry = path-entry *( "," path-entry )
path-entry      = path-type ":" path-target
path-type       = "absolute" / "relative" / "none"
path-target     = 1*(%x21-2B / %x2D-3A / %x3C-7E)
                  ; any printable except comma and semicolon
```

Figure 1: Lifecycle Tuple ABNF

Each path-entry is self-contained with its own path-type prefix. When path-type is absolute, path-target is an absolute-schema-nodeid that resolves from the module root. When path-type is relative, the path is interpreted relative to the schema node that instantiates the grouping containing the deprecated node. When path-type is none, path-target MUST be the literal none to indicate no direct replacement exists.

Human-readable migration guidance, such as operational notes, migration complexity warnings, or removal timelines, SHOULD be documented in the node's description statement rather than in the structured tuple. This maintains a clear separation: the extension provides machine-readable migration paths for automation tools, while the description provides human-readable context for operators and developers.

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 with comma-separated paths allows one-to-many replacements with clear, unambiguous semantics for automated migration tools, including cases that cross module boundaries.

3.2. Allowed Placements and Lifecycle Rules

The ietf-depr:deprecated-info extension MAY be applied to the following YANG statements: container, list, leaf, leaf-list, choice, case, anydata, and anyxml. This extension MUST NOT appear on RPC, action, or notification statements.

The extension MUST be used only with status deprecated. Tooling MUST treat the presence of this extension on a node without status deprecated as an error. This normative pairing ensures the extension serves as structured lifecycle metadata rather than advisory hints.

Each schema node SHOULD have at most one occurrence of the extension. When functionality splits across multiple replacement nodes (one-to-many migration), publishers MUST provide a comma-separated list of paths within a single extension statement. Automated migration tools MUST process all paths in such a list, treating them as mandatory targets for complete migration. This ensures unambiguous semantics: a single tuple defines the complete, deterministic migration path.

Removal Upon Obsolescence: When a node's status transitions from deprecated to obsolete, the ietf-depr:deprecated-info extension MUST be removed. Rationale: Industry best practices mandate that obsolete nodes remain in YANG modules indefinitely for backward compatibility. Retaining structured migration metadata on obsolete nodes leads to unbounded accumulation—in high-volume environments, this can result in hundreds or thousands of unnecessary metadata entries for nodes that users should have migrated from years earlier. Migration guidance for obsolete nodes, if needed, belongs in human-readable description statements.

3.3. Design-Time Evaluation Rationale

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 approach aligns with the guidance in [RFC8407] on providing migration help inside the model itself.

3.4. Extension Versus Instance Data

Replacement paths are documented as extensions rather than instance data because the metadata must travel with the module revisions that introduce deprecations. Instance data is deployment-specific and would not be available to the many tools (IDEs, code generators, documentation sites) that process YANG modules in isolation during development and design-time validation.

3.5. Lifecycle Transitions and Module Maintainability

Several YANG modules deprecate nodes for one or more releases before ultimately marking them status obsolete. This creates a three-stage lifecycle:

1. *Current phase*: Node is in active use with no deprecation or obsoletion marking (the default state in YANG). No migration metadata is needed.

2. *Deprecation phase*: Node carries status deprecated with ietf-depr:deprecated-info extension. This is the *active migration window* where users should transition to replacements and automated tooling can leverage the structured metadata.

3. *Obsolete phase*: Node carries status obsolete and the ietf-depr:deprecated-info extension is *removed*. The node remains in the module indefinitely per industry best practices, with migration guidance provided in the description statement for any laggards.

This approach balances the competing needs of:

* *Automation support*: Structured metadata during deprecation enables proactive, tool-assisted migration during the critical window.

* *Long-term maintainability*: Removing metadata at obsolescence prevents unbounded accumulation. In large-scale production environments with extensive module portfolios, retaining structured metadata on all obsolete nodes would create permanent clutter with minimal value—users encountering an obsolete node years after deprecation have long missed the automation window.

When a deprecated node has no successor (functionality is being removed entirely), the tuple uses path-type = none and path-target = none. This signals to tooling that there is no automated migration target and manual intervention is required. Upon transition to obsolete, this extension is also removed.

3.6. Related Work and Alternatives

Alternative approaches for documenting deprecation paths include 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. Implementation Guidelines and Syntax

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.

4.1. Path Reference Types

To accommodate different structural relationships between deprecated nodes and their replacements, we define three types of path references 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. This approach ensures the reference remains valid regardless of where the grouping is used.

* *none*: Used to signal that no direct replacement exists. Human-readable migration guidance should be provided in the node’s description statement.

Each path reference type has specific syntax requirements and use cases, which are illustrated in the examples that follow.

4.2. Summary of Syntax Notations

The following syntax notations are used for replacement paths:

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

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

* none:none: Indicates that there is no replacement for the deprecated node. Human-readable migration guidance should be provided in the node’s description statement.

* absolute:/path1,absolute:/path2: Multiple paths in a comma-separated list for one-to-many migrations. Each path has its own path-type prefix and all paths are mandatory for complete migration.

* absolute:/path1,relative:/grouping:path2: Mixed path types are allowed in comma-separated lists when replacements span different structural contexts.

4.3. Simple Node Replacement Example

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

YANG Example:

```
ietf-depr:deprecated-info "absolute:/{File_name}:{abs_path}";
```

4.4. No Replacement Example

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

YANG Example:

```
leaf old-feature {
  type string;
  status deprecated;
  description
    "DEPRECATED: This feature is being removed with no direct replacement.
     Manual migration to alternative approaches is required.";
  ietf-depr:deprecated-info "none:none";
}
```

4.5. Grouping Examples

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.

4.5.1. 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-depr:deprecated-info "relative:/{File_name}:{grouping_name}/{rel_path_inside_grou
ping}";
```

4.5.2. 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-depr:deprecated-info "absolute:/{File_name}:{abs_path}";
```

4.5.3. 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-depr:deprecated-info "relative:/{File_name}:{grouping_name}/{rel_path_inside_grou
ping}";
```

4.5.4. 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-depr:deprecated-info "absolute:/{File_name}:{abs_path}";
```

4.6. 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-depr:deprecated-info "absolute:/{File_name}:{replacement_container_path}";

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

5. Example Implementation

The following examples demonstrate how the lifecycle extensions can be implemented. 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.

5.1. ietf-deprecated-info.yang module

YANG Example:

```
   module ietf-deprecated-info {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-deprecated-info";
     prefix ietf-depr;

     organization "IETF NETMOD Working Group";

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

     description
       "This module defines an extension for documenting migration paths
        for deprecated YANG nodes.

        Copyright (c) 2024 IETF Trust and the persons identified as
        authors of the code. All rights reserved.";

     extension deprecated-info {
       argument "tuple";
       description
         "Provides structured migration path for deprecated nodes.

          This extension MUST be used only with 'status deprecated'.

          The tuple provides machine-readable replacement paths for
          automated migration tools. Human-readable guidance (migration
          notes, operational considerations, timelines) SHOULD be
          documented in the node's description statement, maintaining
          clear separation between structured automation metadata and
          human-oriented documentation.

          When a node transitions from 'status deprecated' to
          'status obsolete', this extension MUST be removed to prevent
          accumulation of unnecessary metadata on nodes that will never
          be removed from the module per industry best practices
          (RFC 8407, BBF OD-360).

          Rationale: Industry standards mandate that obsolete nodes
          remain in YANG modules indefinitely for backward compatibility.
          With high-volume deprecations in production environments,
          keeping structured metadata only during the active migration
          window (deprecated phase) balances automation needs with
          long-term module maintainability.";
     }
   }
```

5.2. 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-deprecated-info {
    prefix ietf-depr;
  }

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). "
        "Replacement: case1-replacement. "
        "Note: case1-legacy available for backward compatibility.";
      ietf-depr:deprecated-info
        "absolute:/ietf-deprecation-regression-test-17131:"
        "deprecation-regression-test/configurations/config/"
        "case1-replacement";
    }
    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-depr:deprecated-info
          "absolute:/ietf-deprecation-regression-test-17131:"
          "deprecation-regression-test/configurations/"
          "replacementContainerCase2/case2-replacement";
      }
      container replacementContainerCase2 {
        leaf case2-replacement {
          type uint8;
          description
            "Case 2 - Replacement leaf in different container (NEW)";
        }
      }

      // Case 3: Replacement located in another module.
      leaf case3 {
        type uint8;
        status deprecated;
        description
          "Case 3 - Replacement located in another module, "
          "leaf case3 (DEPRECATED). "
          "Note: The replacement is defined in the ietf-deprecation-regression-test-file
-2 module.";
        ietf-depr:deprecated-info
          "absolute:/ietf-deprecation-regression-test-file-2-17131:"
          "deprecation-regression-test-2/configurations/config/"
          "case3-replacement";
      }

      // 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-depr:deprecated-info
            "absolute:/ietf-deprecation-regression-test-17131:"
            "deprecation-regression-test/configurations/"
            "replacementListCase4/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). "
          "Replacement: case5-replacement (relative within grouping).";
        ietf-depr:deprecated-info
          "relative:/ietf-deprecation-regression-test-17131:"
          "groupCase5/case5-replacement";
      }
      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-depr:deprecated-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-depr:deprecated-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-depr:deprecated-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-depr:deprecated-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-depr:deprecated-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-depr:deprecated-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-depr:deprecated-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-depr:deprecated-info
            "absolute:/ietf-deprecation-regression-test-17131:"
            "deprecation-regression-test/configurations/config/"
            "replacementListCase12/key-case12-replacement";
        }
        leaf case12 {
          type uint8;
          status deprecated;
          ietf-depr:deprecated-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-depr:deprecated-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). "
           "Functionality split across case14a-replacement and case14b-replacement.";
         ietf-depr:deprecated-info
           "absolute:/ietf-deprecation-regression-test-17131:deprecation-regression-test/
configurations/config/case14a-replacement,absolute:/ietf-deprecation-regression-test-1713
1: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).
       Note: This node was previously deprecated with structured
       migration metadata. Upon transition to obsolete status,
       the ietf-depr:deprecated-info extension was removed per
       best practices to prevent metadata accumulation.";
  }

  // Case 16: Container replacement annotated per level.
  container containerCase16 {
    presence "true";
    status deprecated;
    description
      "Case 16 - ContainerCase16 replaced by successor container "
      "(DEPRECATED)";
    ietf-depr:deprecated-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-depr:deprecated-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-depr:deprecated-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)";
          }
        }
      }
    }   } }
```

5.3. 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-deprecated-info {
       prefix ietf-depr;
     }

   // 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-depr:deprecated-info
           "relative:/ietf-depr-reg-test-helper-17131:"
           "groupCase6/case6-replacement";
       }
       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)";
       }
     }
   }
```

5.4. 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-deprecated-info {
    prefix ietf-depr;
  }

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

6. 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.

7. Operational Considerations

Network operators and YANG model consumers can leverage the information provided by the ietf-depr:deprecated-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.

8. 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.

9. 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-deprecated-info 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-deprecated-info, namespace urn:ietf:params:xml:ns:yang:ietf-deprecated-info, prefix ietf-depr, reference this document.

No other IANA actions are required.

10. References

10.1. 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>.

10.2. 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 NETMOD working group participants for their valuable feedback during the IETF 123 discussion. Special thanks to Kent Watsen, Lou Berger, Balazs Lengyel, Reshad Rahman, Deepak Rajaram, and Rob Wilton for their constructive comments that significantly shaped this specification.

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

Change Log

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

Changes from draft-rtv-netmod-yang-subtree-replacement-00 to draft-rtv-netmod-yang-subtree-replacement-01:

* Renamed extension from replacement-info to deprecated-info to reflect source node state rather than target, based on IETF 123 feedback from Kent Watsen

* Extension now mandatory only for status deprecated nodes; MUST be removed when transitioning to status obsolete to prevent unbounded metadata accumulation in long-lived modules

* Renamed module from ietf-replace-path-ext to ietf-deprecated-info to align with extension name and clarify purpose

* Changed path type syntax from REPLACEMENT_ABSOLUTE_PATH/ REPLACEMENT_REL_PATH to lowercase absolute/relative/none for consistency with YANG conventions

* Added formal ABNF grammar definition for lifecycle tuple structure, with each path in a comma-separated list carrying its own path-type prefix to support mixed absolute/relative replacements in one-to-many migrations

* Added explicit rationale for extension-based approach versus instance data (portability and tool coverage requirements)

* Added design-time evaluation rationale section addressing IETF 123 feedback from Rob Wilton and others

* Added lifecycle transitions section explaining three-stage lifecycle (current, deprecated, obsolete) and metadata removal policy

* Clarified allowed statement types and validation rules for extension placement

* Restructured Section 4: moved "Summary of Syntax Notations" before case examples for improved readability

* Documented IETF 123 NETMOD working group chair poll outcome showing broad interest in addressing the problem statement

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