NETMOD Working Group
Internet-Draft
Intended status: Informational
Expires: 3 December 2025

R. T. Venkateswaran
S. V. G. Karnati
S. Jain
V. Ramamoorthy
V. H. Nagamangalam
Cisco Systems, Inc.
1 July 2025

Enhancements to the YANG Language for Capturing Subtree Replacements
draft-rtv-netmod-yang-subtree-replacement-00

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 devices and services evolve, YANG data models must also evolve to support new features and functionality. This evolution often requires deprecating or obsoleting existing nodes in favor of newer, more appropriate structures.

The YANG language provides "status" statements to indicate that a data node has been deprecated or made obsolete, but it does not provide a standardized mechanism to indicate what new node(s) should be used instead. This lack of standardized replacement information creates challenges for users of YANG models who need to update their systems to use newer model elements.

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 even informal 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.

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-ext:replacement-info, which provides structured metadata indicating the appropriate replacement for a deprecated or obsolete node. This extension carries precise XPath information to the replacement node(s), enabling both human operators and automated tools to locate the new node.

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 extension can represent various replacement scenarios, including one-to-one, one-to-many, many-to-one replacements and replacements across different YANG modules.

4. Implementation of the Replacement Keyword Across Various Scenarios

The following sections demonstrate how the ietf-ext:replacement-info extension can be applied across different replacement scenarios. The examples illustrate common patterns encountered when evolving YANG models and show how the extension provides clear migration paths for each situation.

4.1. Path Reference Types

To accommodate different structural relationships between deprecated nodes and their replacements, we define two types of path references:

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

* *REPLACEMENT_REL_PATH:* Used when the replacement node appears in multiple places, such as within structures like groupings, augments, etc. This approach ensures the reference remains valid regardless of where the node is used.

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

4.2. Case 1: Simple Node with Replacement

Node that is deprecated and has a replacement node.

YANG Example:

```
'''yang

ietf-ext:replacement-info
"REPLACEMENT_ABSOLUTE_PATH:/{File_name}:{abs_path}"; '''
```

4.3. Case 2: Node Deprecated with No Replacement

Node that is deprecated and has no replacement node.

YANG Example:

```
'''yang

ietf-ext:replacement-info "REPLACEMENT_ABSOLUTE_PATH:None"; '''
```

4.4. 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 proposed solution involves uniquely identifying the grouping first using the file name and grouping name, then providing 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.

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

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

YANG Example:

```
'''yang

ietf-ext:replacement-info "REPLACEMENT_REL_PATH:/{File_name}:{groupin
g_name}/{rel_path_inside_grouping}"; '''
```

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

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

YANG Example:

‘‘‘yang

ietf-ext:replacement-info
"REPLACEMENT_ABSOLUTE_PATH:/{File_name}:{abs_path}"; ‘‘‘

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

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

YANG Example:

‘‘‘yang

ietf-ext:replacement-info "REPLACEMENT_REL_PATH:/{File_name}:{groupin
g_name}/{rel_path_inside_grouping}"; ‘‘‘

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

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

YANG Example:

‘‘‘yang

ietf-ext:replacement-info
"REPLACEMENT_ABSOLUTE_PATH:/{File_name}:{abs_path}"; ‘‘‘

4.5. Summary of Syntax Notations

The following syntax notations are used for replacement paths:

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

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

* None: Indicates that there is no replacement for the deprecated node.

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:

```yang
container old-container { status deprecated; ietf-ext:replacement-info "REPLACEMENT_ABSOLUTE_PATH:/{File_name}:{replacement_container_path}";

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

5. Example Implementation

The following examples demonstrate how the replacement path 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-replace-path-ext.yang module

YANG Example:

```yang
module ietf-replace-path-ext { namespace "urn:ietf:params:xml:ns:yang:ietf-replace-path-ext"; prefix ietf-ext;

organization "IETF NETMOD Working Group";
```

```
   contact "IETF NETMOD Working Group netmod@ietf.org
   (mailto:netmod@ietf.org)";

   description "This module defines extensions for additional metadata.
   Copyright (c) 2024 IETF Trust and the persons identified as authors
   of the code.  All rights reserved.";

   extension replacement-info { argument "value"; description "Provides
   replacement model information for deprecated/obsolete model."; } }
   '''
```

5.2. example-deprecation-regression-test-17131.yang module

YANG Example:

```
   '''yang

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

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

   import ietf-replace-path-ext { prefix ietf-ext; }

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

    // Case 1 : Leaf deprecated with a replacement in the same container.
    leaf case1 {
      type uint8;
      status deprecated;
      description
        "This leaf gives information about Case 1 - Leaf deprecated with a
         replacement in the same container, leaf case1 (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/case1-replacement";
    }
    leaf case1-replacement {
      type uint16;
      description
        "This leaf gives information about Case 1 - Leaf deprecated with a
         replacement in the same container, leaf case1-replacement (NEW)";
    }

    // Case 2 : Leaf deprecated with a replacement in a different container.
    leaf case2 {
```

```
      type uint8;
      status deprecated;
      description
        "This leaf gives information about Case 2 - Leaf deprecated with a
         replacement in a different container, leaf i (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/replacementContainerCase2/case2-replacement";
    }

    // Case 3 : Leaf deprecated with a replacement located in another YANG file(example-d
eprecation-regression-test-file-2.yang)
    leaf case3 {
      type uint8;
      status deprecated;
      description
        "This leaf gives information about Case 3 - Leaf deprecated with a
         replacement located in another YANG file, leaf case3 (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-file-2-17131:depr
ecation-regression-test-2/configurations/config/case3-replacement";
    }

    // Case 4 : Leaf inside a list deprecated with a replacement in a different list.
    list listCase4 {
      key "key-case4";
      leaf key-case4 {
        type string;
      }
      leaf case4 {
        type uint8;
        status deprecated;
        description
          "This leaf gives information about Case 4 - Leaf inside a list
           deprecated with a replacement in a different list, leaf case4
           (DEPRECATED)";
        ietf-ext:replacement-info
          "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecati
on-regression-test/configurations/config/replacementListCase4/case4-replacement";
      }
    }
    list replacementListCase4 {
      key "key-case4-replacement";
      leaf key-case4-replacement {
        type string;
      }
      leaf case4-replacement {
        type uint8;
        description
          "This leaf gives information about Case 4 - Leaf inside a list
           deprecated with a replacement in a different list, leaf
           case4-replacement (NEW)";
```

```
      }
    }

    // Case 5 : Leaf inside a grouping deprecated, replaced in the same grouping, and use
d across multiple containers.
    grouping groupCase5 {
      leaf case5 {
        type uint8;
        status deprecated;
        description
          "This leaf gives information about Case 5 - Leaf inside a grouping
           deprecated, replaced in the same grouping, and used across multiple
           containers, leaf case5 (DEPRECATED)";
        ietf-ext:replacement-info
          "REPLACEMENT_REL_PATH:/example-deprecation-regression-test-17131:groupCase5/cas
e5-replacement";
      }
      leaf case5-replacement {
        type uint16;
        description
          "This leaf gives information about Case 5 - Leaf inside a grouping
           deprecated, replaced in the same grouping, and used across multiple
           containers, leaf case5-replacement (NEW)";
      }
    }
    container containerP1 {
      uses groupCase5;
      description
        "This container gives information about Case 5 - Leaf inside a
         grouping deprecated, replaced in the same grouping, and used across
         multiple containers, using containerP1 (NEW)";
    }
    container containerP2 {
      uses groupCase5;
      description
        "This container gives information about Case 5 - Leaf inside a
         grouping deprecated, replaced in the same grouping, and used across
         multiple containers, using containerP2 (NEW)";
    }

    // Case 6 : Leaf inside a grouping deprecated, replaced in the same grouping but impo
rted through a different module and used in various modules.
    container containerP3 {
      uses depr-reg-test-helper:groupCase6;
      description
        "This container gives information about Case 6 - Leaf inside a
         grouping deprecated, replaced in the same grouping but imported
         through a different module and used in various modules, leaf inside
         containerP3 (NEW)";
    }
    container containerP4 {
```

```
      uses depr-reg-test-helper:groupCase6;
      description
        "This container gives information about Case 6 - Leaf inside a
         grouping deprecated, replaced in the same grouping but imported
         through a different module and used in various modules, leaf inside
         containerP4 (NEW)";
    }

    // Case 7 : Leaf deprecated, replaced by a different leaf with the same name in a dif
ferent location.
    leaf case7 {
      type uint8;
      status deprecated;
      description
        "This leaf gives information about Case 7 - Leaf deprecated, replaced
         by a different leaf with the same name in a different location, leaf
         case7 (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/replacementContainerCase7/case7";
    }
    container replacementContainerCase7 {
      presence "true";
      description
        "This container gives information about Case 7 - Leaf deprecated,
         replaced by a different leaf with the same name in a different
         location, container replacementContainerCase7 (NEW)";
      leaf case7 {
        type uint8;
        description
          "This leaf gives information about Case 7 - Leaf deprecated,
           replaced by a different leaf with the same name in a different
           location, leaf case7 (NEW)";
      }
    }

    // Case 8 : Leaf-list deprecated, replaced by another leaf-list
    leaf-list case8 {
      type uint8;
      status deprecated;
      description
        "This leaf-list gives information about Case 8 - Leaf-list deprecated,
         replaced by another leaf-list, leaf-list case8 (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/case8-replacement";
    }
    leaf-list case8-replacement {
      type uint8;
      description
        "This leaf-list gives information about Case 8 - Leaf-list
```

```
         deprecated, replaced by another leaf-list, leaf-list
         case8-replacement (NEW)";
    }

    // Case 9 : Empty leaf deprecated, replaced by another empty leaf
    leaf case9 {
      type empty;
      status deprecated;
      description
        "This leaf gives information about Case 9 - Empty leaf deprecated,
         replaced by another empty leaf, leaf case9 (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/case9-replacement";
    }
    leaf case9-replacement {
      type empty;
      description
        "This leaf gives information about Case 9 - Empty leaf deprecated,
         replaced by another empty leaf, leaf case9-replacement (NEW)";
    }

    // Case 10 : A container is deprecated with a replacement container.
    container containerCase10 {
      leaf case10 {
        type uint8;
        status deprecated;
        description
          "This leaf gives information about Case 10 - A container is
           deprecated with a replacement container, leaf case10
           (DEPRECATED)";
        ietf-ext:replacement-info
          "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecati
on-regression-test/configurations/config/replacementContainerCase10/case10-replacement";
      }
      status deprecated;
      description
        "This container gives information about Case 10 - A container is
         deprecated with a replacement container, containerCase10
         (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/replacementContainerCase10";
    }
    container replacementContainerCase10 {
      leaf case10-replacement {
        type uint8;
        description
          "This leaf gives information about Case 10 - A container is
           deprecated with a replacement container, leaf case10-replacement
           (NEW)";
```

```
      }
      description
        "This container gives information about Case 10 - A container is
         deprecated with a replacement container, replacementContainerCase10
         (NEW)";
    }

    // Case 11 : A multiple deprecated items are replaced by a single item.
    leaf case11a {
      type uint8;
      status deprecated;
      description
        "This leaf gives information about Case 11 - A multiple deprecated
         items are replaced by a single item, leaf case11a (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/case11-replacement";
    }
    leaf case11b {
      type uint8;
      status deprecated;
      description
        "This leaf gives information about Case 11 - A multiple deprecated
         items are replaced by a single item, leaf case11b (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/case11-replacement";
    }
    leaf case11-replacement {
      type uint16;
      description
        "This leaf gives information about Case 11 - A multiple deprecated
         items are replaced by a single item, leaf case11-replacement (NEW)";
    }

    // Case 12 : Leaf inside a list deprecated, but the replacement is in a different lis
t type (not one-to-one mapping).
    list listCase12 {
      key "key-case12";
      leaf key-case12 {
        type string;
      }
      leaf case12 {
        type uint8;
        ietf-ext:replacement-info
          "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecati
on-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 : In a choice-case, the leaf is deprecated in one case with an alternative
 provided in a new case.
    choice choice13 {
      case caseCase13 {
        leaf case13 {
          type uint8;
          status deprecated;
          description
            "This leaf gives information about Case 13 - In a choice-case, the
             leaf is deprecated in one case with an alternative provided in a
             new case, case13 (DEPRECATED)";
          ietf-ext:replacement-info
            "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:depreca
tion-regression-test/configurations/config/choice13/case13-replacement";
        }
      }
      case caseCase13Replacement {
        leaf case13-replacement {
          type uint8;
          description
            "This leaf gives information about Case 13 - In a choice-case, the
             leaf is deprecated in one case with an alternative provided in a
             new case, case13-replacement (NEW)";
        }
      }
    }

    // Case 14 : A single leaf is deprecated, and its functionality is distributed among
multiple new leaves.
    leaf case14 {
      type uint16;
      status deprecated;
      description
        "This leaf gives information about Case 14 - A single leaf is
         deprecated, and its functionality is distributed among multiple new
         leaves, case14 (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/case14a-replacement,
         REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/case14b-replacement";
    }
    leaf case14a-replacement {
      type uint8;
      description
        "This leaf gives information about Case 14 - A single leaf is
```

```
         deprecated, and its functionality is distributed among multiple new
         leaves, case14a-replacement (NEW)";
    }
    leaf case14b-replacement {
      type uint8;
      description
        "This leaf gives information about Case 14 - A single leaf is
         deprecated, and its functionality is distributed among multiple new
         leaves, case14b-replacement (NEW)";
    }

    // Case 15 : A leaf is deprecated without alternate as the feature is no longer suppo
rted in higer versions.
    leaf case15 {
      type uint8;
      status deprecated;
      description
        "This leaf gives information about Case 15 - A leaf is deprecated
         without alternate as the feature is no longer supported in higer
         versions, case15 (DEPRECATED)";
      ietf-ext:replacement-info "REPLACEMENT_ABSOLUTE_PATH:None";
    }

    // Case 16: An container is deprecated with a replacement, and the replacement value
is specified at each level of the container and its child leaf elements.
    container containerCase16 {
      presence "true";
      status deprecated;
      description
        "This container gives information about Case 16 - An empty container is
         deprecated with a replacement, containerCase16 (DEPRECATED)";
      ietf-ext:replacement-info
        "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecation
-regression-test/configurations/config/replacementContainerCase16";

      leaf case16-leaf1 {
        type string;
        status deprecated;
        description
          "This leaf gives information about Case 16 - Leaf case16-leaf1 is
           deprecated within the deprecated container, case16-leaf1
           (DEPRECATED)";
        ietf-ext:replacement-info
          "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecati
on-regression-test/configurations/config/replacementContainerCase16/case16a";
      }

      leaf case16-leaf2 {
        type uint8;
        status deprecated;
        description
          "This leaf gives information about Case 16 - Leaf case16-leaf2 is
```

```
           deprecated within the deprecated container, case16-leaf2
           (DEPRECATED)";
         ietf-ext:replacement-info
           "REPLACEMENT_ABSOLUTE_PATH:/example-deprecation-regression-test-17131:deprecati
on-regression-test/configurations/config/replacementContainerCase16/case16b";
       }
     }

     container replacementContainerCase16 {
       presence "true";
       description
         "This container gives information about Case 16 - Replacement for the
          deprecated containerCase16, replacementContainerCase16 (NEW)";

       leaf case16a {
         type string;
         description
           "This leaf gives information about Case 16 - Replacement for
            case16-leaf1, case16a (NEW)";
       }

       leaf case16b {
         type uint8;
         description
           "This leaf gives information about Case 16 - Replacement for
            case16-leaf2, case16b (NEW)";
       }
     }
   }
}   } } ‘‘‘
```

5.3. example-deprecation-regression-test-helper-module-17131.yang module

YANG Example:

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

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

  prefix depr-reg-test-helper;

  organization "IETF NETMOD Working Group";

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

  import ietf-replace-path-ext {
    prefix ietf-ext;
  }

  // Case 6 : Leaf inside a grouping deprecated, replaced in the same grouping
  // but imported through a different module and used in various modules.
  grouping groupCase6 {
    leaf case6 {
      type uint8;
      default 10;
      status deprecated;
      description
        "This leaf gives information about 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:replacement-info
        "REPLACEMENT_REL_PATH:/example-depr-reg-test-helper-module-17131/groupCase6/case6
-replacement";
    }
    leaf case6-replacement {
      type uint8;
      default 11;
      description
        "This leaf gives information about 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. example-deprecation-regression-test-file-2-17131.yang module

YANG Example:

```
'''yang

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

import ietf-replace-path-ext {
  prefix ietf-ext;
}

container deprecation-regression-test-2 {
  container configurations {
    container config {
      // Case 3 : Leaf deprecated with a replacement located in another YANG file
      // (example-deprecation-regression-test-file-2.yang)
      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-ext:replacement-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

This document specifies an extension to the YANG language for documenting replacement paths for deprecated or obsolete nodes. As such, it does not introduce any new security risks beyond what exists in the current YANG language specification [RFC7950].

9. IANA Considerations

This document has no IANA actions.

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., Ed., "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", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/rfc/rfc8407>.

10.2. Informative References

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/rfc/rfc8340>.

Acknowledgments

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

The authors acknowledge the original work on YANG Next Steps [YANG-NEXT] that provided initial inspiration for this proposal.

[YANG-NEXT] https://github.com/netmod-wg/yang-next/issues/130

Authors' Addresses

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

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

Sarthak Jain
Cisco Systems, Inc.
India
Email: sarthakj@cisco.com

Veena Ramamoorthy
Cisco Systems, Inc.
India
Email: vemoorth@cisco.com

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