TEAS Working Group
Internet-Draft
Updates: 8776 (if approved)
Intended status: Standards Track
Expires: 8 September 2022

I. Busi
Huawei
A. Guo
Futurewei Technologies
X. Liu
IBM Corporation
T. Saad
Juniper Networks
R. Gandhi
Cisco Systems, Inc.
V. P. Beeram
Juniper Networks
I. Bryskin
Individual
7 March 2022

Updated Common YANG Data Types for Traffic Engineering
draft-busi-teas-te-types-update-01

Abstract

This document defines few additional common data types and groupings in YANG data modeling language to be imported by modules that model Traffic Engineering (TE) configuration and state capabilities.

This document updates RFC 8776 with a new revision of the module ietf-te-types.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

Table of Contents

1. Introduction

After the pubblication of [RFC8776], the need to add a new typedef and a new grouping to ietf-te-types YANG module has arisen.

These definitions have been developed in [I-D.ietf-teas-yang-te] and [I-D.ietf-teas-yang-l3-te-topo] and are quite mature: [I-D.ietf-teas-yang-te] in particular is ready from WG Last Call.

However, these defintions have broader applicability than the I-D where they have originated, so it makes sense to move them within the ietf-te-types YANG module.

1.1. Options considered

The concern is how to be able to update the ietf-te-types YANG module published in [RFC8776] without delaying too much the progress of the mature WG documents.

Three possible options have been identified to address this concern.

One option is to keep these definitions in the YANG modules where they have initially been defined: other YANG modules can still import them. The drawback of this approach is that it defeating the value of common YANG modules like ietf-te-types since common definitions will be spread around multiple specific YANG modules.

A second option is to define them in a new common YANG module (e.g., ietf-te-types-ext). The drawback of this approach is that it will increase the number of YANG modules providing thiny updates to the ietf-te-types YANG module.

A third option is to develop a revision of the ietf-te-types YANG module within an RFC8776-bis. The drawback of this approach is that the process for developing a big RFC8776-bis just for a thiny update is too high. Moreover, it is not clear what could be done with the ietf-te-packet-types YANG module which is also defined in [RFC8776] but it does not need to be revised.

This document explores an alternative option to just update [RFC8776] with a new revision of the module ietf-te-types.

In order to focus the review process of this document only to the changes proposed by this document: - Section 2 describes only the updates to the ietf-te-types YANG module proposed by this document; - Section 3 defines only the diff between the revision of the ietf-te-types YANG module proposed in this document and the revision of the ietf-te-types YANG module published in [RFC8776].

In order to allow all the YANG toolchain to keep working by extracting the revision of the ietf-te-types YANG module proposed in this document, this revision is provided by Appendix A. This text is intended not to be subject to the review of this document.

1.2. Requirements Notation

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.

1.3. Terminology

The terminology for describing YANG data models is found in [RFC7950].

1.4. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects, added to the ietf-te-types YANG module do not need to be prefixed.

The revision of the ietf-te-types YANG module uses the prefixes defined in section 1.2 of [RFC8776].

2. Overview

The module ietf-te-types has been updated to add the following YANG identies, types and groupings which can be reused by TE YANG models:

bandwidth-scientific-notation This types represents the bandwidth in bit-per-second, using the scientific notation (e.g., 10e3).

encoding-and-switching-type This is a common grouping to define the LSP encoding and switching types.

3. TE Types YANG Update

This section provides the diff between the YANG module in section 3.1 of [RFC8776] and the YANG module revision in Appendix A.

Note - This diff has been generated using the following UNIX commands to compare the YANG module revisions in section 3.1 of [RFC8776] and in Appendix A:

```
diff ietf-te-types@2020-06-10.yang ietf-te-types.yang > model-diff.txt
sed 's/^/    /' model-diff.txt > model-diff-spaces.txt
sed 's/^    >   /    >   /' model-diff-spaces.txt > model-updates.txt
```

The output (model-updates.txt) is reported here:

```
55c55
<       Copyright (c) 2020 IETF Trust and the persons identified as
---
>       Copyright (c) 2022 IETF Trust and the persons identified as
65c65
<       This version of this YANG module is part of RFC 8776; see the
---
>       This version of this YANG module is part of RFC XXXX; see the
67a68,74
```

```
  >   revision 2022-03-07 {
  >     description
  >       "Latest revision of TE types.";
  >     reference
  >       "RFC XXXX: Updated Common YANG Data Types for Traffic
  >       Engineering";
  >   }
  545a553,583
  >   typedef bandwidth-scientific-notation {
  >     type string {
  >       pattern
  >         '0(\.0?)?([eE](\+)?0?)?|'
  >       + '[1-9](\.[0-9]{0,6})?[eE](\+)?(9[0-6]|[1-8][0-9]|0?[0-9])?';
  >     }
  >     units "bps";
  >     description
  >       "Bandwidth values, expressed using the scientific notation
  >       in bits per second.
  >       The encoding format is the external decimal-significant
  >       character sequences specified in IEEE 754 and ISO/IEC C99
  >       for 32-bit decimal floating-point numbers:
  >       (-1)**(S) * 10**(Exponent) * (Significant),
  >       where Significant uses 7 digits.
  >       An implementation for this representation may use decimal32
  >       or binary32. The range of the Exponent is from -95 to +96
  >       for decimal32, and from -38 to +38 for binary32.
  >       As a bandwidth value, the format is restricted to be
  >       normalized, non-negative, and non-fraction:
  >       n.dddddde{+}dd, N.DDDDDDE{+}DD, 0e0 or 0E0,
  >       where 'd' and 'D' are decimal digits; 'n' and 'N' are
  >       non-zeror decimal digits; 'e' and 'E' indicate a power of ten.
  >       Some examples are 0e0, 1e10, and 9.953e9.";
  >     reference
  >       "IEEE Std 754-2008: IEEE Standard for Floating-Point
  >       Arithmetic.
  >       ISO/IEC C99: Information technology - Programming
  >       Languages - C.";
  >   }
  >
  3376a3415,3438
  >     }
  >   }
  >
  >   grouping encoding-and-switching-type {
  >     description
  >       "Common grouping to define the LSP encoding and
  >       switching types";
  >     leaf encoding {
```

```
>       type identityref {
>         base te-types:lsp-encoding-types;
>       }
>       description
>         "LSP encoding type.";
>       reference
>         "RFC3945";
>     }
>     leaf switching-type {
>       type identityref {
>         base te-types:switching-capabilities;
>       }
>       description
>         "LSP switching type.";
>       reference
>         "RFC3945";
```

4. IANA Considerations

This document updates the ietf-te-types YANG module registered by [RFC8776].

Therefore this document does not require any IANA actions.

5. Security Considerations

The security considerations defined in section 7 of [RFC8776] applies to the revision of the ietf-te-types YANG module.

This document just adds new typedefs and groupings to the YANG modules defined in [RFC8776] and therefore it does not introduce additional considerations.

6. References

6.1. 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/info/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/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8776] Saad, T., Gandhi, R., Liu, X., Beeram, V., and I. Bryskin, "Common YANG Data Types for Traffic Engineering", RFC 8776, DOI 10.17487/RFC8776, June 2020, <https://www.rfc-editor.org/info/rfc8776>.

6.2. Informative References

[I-D.ietf-teas-yang-l3-te-topo]
Liu, X., Bryskin, I., Beeram, V. P., Saad, T., Shah, H., and O. G. D. Dios, "YANG Data Model for Layer 3 TE Topologies", Work in Progress, Internet-Draft, draft-ietf-teas-yang-l3-te-topo-12, 24 October 2021, <https://www.ietf.org/archive/id/draft-ietf-teas-yang-l3-te-topo-12.txt>.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V. P., Bryskin, I., and O. G. D. Dios, "A YANG Data Model for Traffic Engineering Tunnels, Label Switched Paths and Interfaces", Work in Progress, Internet-Draft, draft-ietf-teas-yang-te-29, 7 February 2022, <https://www.ietf.org/archive/id/draft-ietf-teas-yang-te-29.txt>.

Appendix A. TE Types YANG Module

```
<CODE BEGINS> file "ietf-te-types@2022-03-07.yang"
module ietf-te-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-types";
  prefix te-types;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
```

```
         "RFC 8294: Common YANG Data Types for the Routing Area";
     }

     organization
       "IETF Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";
     contact
       "WG Web:   <https://datatracker.ietf.org/wg/teas/>
        WG List:  <mailto:teas@ietf.org>

        Editor:   Tarek Saad
                  <mailto:tsaad@juniper.net>

        Editor:   Rakesh Gandhi
                  <mailto:rgandhi@cisco.com>

        Editor:   Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Xufeng Liu
                  <mailto:xufeng.liu.ietf@gmail.com>

        Editor:   Igor Bryskin
                  <mailto:i_bryskin@yahoo.com>";
     description
       "This YANG module contains a collection of generally useful
        YANG data type definitions specific to TE.  The model fully
        conforms to the Network Management Datastore Architecture
        (NMDA).

        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.

        Copyright (c) 2022 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

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

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

```
     revision 2022-03-07 {
       description
         "Latest revision of TE types.";
       reference
         "RFC XXXX: Updated Common YANG Data Types for Traffic
         Engineering";
     }
     revision 2020-06-10 {
       description
         "Latest revision of TE types.";
       reference
         "RFC 8776: Common YANG Data Types for Traffic Engineering";
     }

     /**
      * Typedefs
      */

     typedef admin-group {
       type yang:hex-string {
         /* 01:02:03:04 */
         length "1..11";
       }
       description
         "Administrative group / resource class / color representation
          in 'hex-string' type.
          The most significant byte in the hex-string is the farthest
          to the left in the byte sequence.  Leading zero bytes in the
          configured value may be omitted for brevity.";
       reference
         "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
          Version 2
          RFC 5305: IS-IS Extensions for Traffic Engineering
          RFC 7308: Extended Administrative Groups in MPLS Traffic
          Engineering (MPLS-TE)";
     }

     typedef admin-groups {
       type union {
         type admin-group;
         type extended-admin-group;
       }
       description
         "Derived types for TE administrative groups.";
     }

     typedef extended-admin-group {
       type yang:hex-string;
```

```
      description
        "Extended administrative group / resource class / color
         representation in 'hex-string' type.
         The most significant byte in the hex-string is the farthest
         to the left in the byte sequence.  Leading zero bytes in the
         configured value may be omitted for brevity.";
      reference
        "RFC 7308: Extended Administrative Groups in MPLS Traffic
         Engineering (MPLS-TE)";
    }

    typedef path-attribute-flags {
      type union {
        type identityref {
          base session-attributes-flags;
        }
        type identityref {
          base lsp-attributes-flags;
        }
      }
      description
        "Path attributes flags type.";
    }

    typedef performance-metrics-normality {
      type enumeration {
        enum unknown {
          value 0;
          description
            "Unknown.";
        }
        enum normal {
          value 1;
          description
            "Normal.  Indicates that the anomalous bit is not set.";
        }
        enum abnormal {
          value 2;
          description
            "Abnormal.  Indicates that the anomalous bit is set.";
        }
      }
      description
        "Indicates whether a performance metric is normal (anomalous
         bit not set), abnormal (anomalous bit set), or unknown.";
      reference
        "RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions
         RFC 7823: Performance-Based Path Selection for Explicitly
```

```
          Routed Label Switched Paths (LSPs) Using TE Metric
          Extensions
          RFC 8570: IS-IS Traffic Engineering (TE) Metric Extensions";
      }

      typedef srlg {
        type uint32;
        description
          "SRLG type.";
        reference
          "RFC 4203: OSPF Extensions in Support of Generalized
           Multi-Protocol Label Switching (GMPLS)
           RFC 5307: IS-IS Extensions in Support of Generalized
           Multi-Protocol Label Switching (GMPLS)";
      }

      typedef te-common-status {
        type enumeration {
          enum up {
            description
              "Enabled.";
          }
          enum down {
            description
              "Disabled.";
          }
          enum testing {
            description
              "In some test mode.";
          }
          enum preparing-maintenance {
            description
              "The resource is disabled in the control plane to prepare
               for a graceful shutdown for maintenance purposes.";
            reference
              "RFC 5817: Graceful Shutdown in MPLS and Generalized MPLS
               Traffic Engineering Networks";
          }
          enum maintenance {
            description
              "The resource is disabled in the data plane for maintenance
               purposes.";
          }
          enum unknown {
            description
              "Status is unknown.";
          }
        }
```

```
      description
        "Defines a type representing the common states of a TE
         resource.";
    }

    typedef te-bandwidth {
      type string {
        pattern '0[xX](0(((\.0?)?[pP](\+)?0?|(\.0?))|'
              + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?)?)?'
              + '[pP](\+)?(12[0-7]|'
              + '1[01]\d|0?\d?\d)?)|0[xX][\da-fA-F]{1,8}|\d+'
              + '(,(0[xX](0(((\.0?)?[pP](\+)?0?|(\.0?))|'
              + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?)?)?'
              + '[pP](\+)?(12[0-7]|'
              + '1[01]\d|0?\d?\d)?)|0[xX][\da-fA-F]{1,8}|\d+))*';
      }
      description
        "This is the generic bandwidth type.  It is a string containing
         a list of numbers separated by commas, where each of these
         numbers can be non-negative decimal, hex integer, or
         hex float:

         (dec | hex | float)[*(',',(dec | hex | float))]

         For the packet-switching type, the string encoding follows
         the type 'bandwidth-ieee-float32' as defined in RFC 8294
         (e.g., 0x1p10), where the units are in bytes per second.

         For the Optical Transport Network (OTN) switching type,
         a list of integers can be used, such as '0,2,3,1', indicating
         two ODU0s and one ODU3.  ('ODU' stands for 'Optical Data
         Unit'.)  For Dense Wavelength Division Multiplexing (DWDM),
         a list of pairs of slot numbers and widths can be used,
         such as '0,2,3,3', indicating a frequency slot 0 with
         slot width 2 and a frequency slot 3 with slot width 3.
         Canonically, the string is represented as all lowercase and in
         hex, where the prefix '0x' precedes the hex number.";
      reference
        "RFC 8294: Common YANG Data Types for the Routing Area
         ITU-T Recommendation G.709: Interfaces for the
         optical transport network";
    }

    typedef te-ds-class {
      type uint8 {
        range "0..7";
      }
      description
```

```
        "The Differentiated Services Class-Type of traffic.";
      reference
        "RFC 4124: Protocol Extensions for Support of Diffserv-aware
         MPLS Traffic Engineering, Section 4.3.1";
    }

    typedef te-global-id {
      type uint32;
      description
        "An identifier to uniquely identify an operator, which can be
         either a provider or a client.
         The definition of this type is taken from RFCs 6370 and 5003.
         This attribute type is used solely to provide a globally
         unique context for TE topologies.";
      reference
        "RFC 5003: Attachment Individual Identifier (AII) Types for
         Aggregation
         RFC 6370: MPLS Transport Profile (MPLS-TP) Identifiers";
    }

    typedef te-hop-type {
      type enumeration {
        enum loose {
          description
            "A loose hop in an explicit path.";
        }
        enum strict {
          description
            "A strict hop in an explicit path.";
        }
      }
      description
        "Enumerated type for specifying loose or strict paths.";
      reference
        "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
         Section 4.3.3";
    }

    typedef te-link-access-type {
      type enumeration {
        enum point-to-point {
          description
            "The link is point-to-point.";
        }
        enum multi-access {
          description
            "The link is multi-access, including broadcast and NBMA.";
        }
```

```
      }
      description
        "Defines a type representing the access type of a TE link.";
      reference
        "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
         Version 2";
    }

    typedef te-label-direction {
      type enumeration {
        enum forward {
          description
            "Label allocated for the forward LSP direction.";
        }
        enum reverse {
          description
            "Label allocated for the reverse LSP direction.";
        }
      }
      description
        "Enumerated type for specifying the forward or reverse
         label.";
    }

    typedef te-link-direction {
      type enumeration {
        enum incoming {
          description
            "The explicit route represents an incoming link on
             a node.";
        }
        enum outgoing {
          description
            "The explicit route represents an outgoing link on
             a node.";
        }
      }
      description
        "Enumerated type for specifying the direction of a link on
         a node.";
    }

    typedef te-metric {
      type uint32;
      description
        "TE metric.";
      reference
        "RFC 3785: Use of Interior Gateway Protocol (IGP) Metric as a
```

```
          second MPLS Traffic Engineering (TE) Metric";
      }

      typedef te-node-id {
        type yang:dotted-quad;
        description
          "A type representing the identifier for a node in a TE
           topology.
           The identifier is represented as 4 octets in dotted-quad
           notation.
           This attribute MAY be mapped to the Router Address TLV
           described in Section 2.4.1 of RFC 3630, the TE Router ID
           described in Section 3 of RFC 6827, the Traffic Engineering
           Router ID TLV described in Section 4.3 of RFC 5305, or the
           TE Router ID TLV described in Section 3.2.1 of RFC 6119.
           The reachability of such a TE node MAY be achieved by a
           mechanism such as that described in Section 6.2 of RFC 6827.";
        reference
          "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
           Version 2, Section 2.4.1
           RFC 5305: IS-IS Extensions for Traffic Engineering,
           Section 4.3
           RFC 6119: IPv6 Traffic Engineering in IS-IS, Section 3.2.1
           RFC 6827: Automatically Switched Optical Network (ASON)
           Routing for OSPFv2 Protocols, Section 3";
      }

      typedef te-oper-status {
        type te-common-status;
        description
          "Defines a type representing the operational status of
           a TE resource.";
      }

      typedef te-admin-status {
        type te-common-status;
        description
          "Defines a type representing the administrative status of
           a TE resource.";
      }

      typedef te-path-disjointness {
        type bits {
          bit node {
            position 0;
            description
              "Node disjoint.";
          }
```

```
        bit link {
          position 1;
          description
            "Link disjoint.";
        }
        bit srlg {
          position 2;
          description
            "SRLG (Shared Risk Link Group) disjoint.";
        }
      }
      description
        "Type of the resource disjointness for a TE tunnel path.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    typedef te-recovery-status {
      type enumeration {
        enum normal {
          description
            "Both the recovery span and the working span are fully
             allocated and active, data traffic is being
             transported over (or selected from) the working
             span, and no trigger events are reported.";
        }
        enum recovery-started {
          description
            "The recovery action has been started but not completed.";
        }
        enum recovery-succeeded {
          description
            "The recovery action has succeeded.  The working span has
             reported a failure/degrade condition, and the user traffic
             is being transported (or selected) on the recovery span.";
        }
        enum recovery-failed {
          description
            "The recovery action has failed.";
        }
        enum reversion-started {
          description
            "The reversion has started.";
        }
        enum reversion-succeeded {
          description
            "The reversion action has succeeded.";
```

```
        }
        enum reversion-failed {
          description
            "The reversion has failed.";
        }
        enum recovery-unavailable {
          description
            "The recovery is unavailable, as a result of either an
             operator's lockout command or a failure condition
             detected on the recovery span.";
        }
        enum recovery-admin {
          description
            "The operator has issued a command to switch the user
             traffic to the recovery span.";
        }
        enum wait-to-restore {
          description
            "The recovery domain is recovering from a failure/degrade
             condition on the working span that is being controlled by
             the Wait-to-Restore (WTR) timer.";
        }
      }
      description
        "Defines the status of a recovery action.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)
         RFC 6378: MPLS Transport Profile (MPLS-TP) Linear Protection";
    }

    typedef te-template-name {
      type string {
        pattern '/?([a-zA-Z0-9\-_.]+)(/[a-zA-Z0-9\-_.]+)*';
      }
      description
        "A type for the name of a TE node template or TE link
         template.";
    }

    typedef te-topology-event-type {
      type enumeration {
        enum add {
          value 0;
          description
            "A TE node or TE link has been added.";
        }
        enum remove {
```

```
          value 1;
          description
            "A TE node or TE link has been removed.";
        }
        enum update {
          value 2;
          description
            "A TE node or TE link has been updated.";
        }
      }
      description
        "TE event type for notifications.";
    }

    typedef te-topology-id {
      type union {
        type string {
          length "0";
          // empty string
        }
        type string {
          pattern '([a-zA-Z0-9\-_.]+:)*'
                + '/?([a-zA-Z0-9\-_.]+)(/[a-zA-Z0-9\-_.]+)*';
        }
      }
      description
        "An identifier for a topology.
         It is optional to have one or more prefixes at the beginning,
         separated by colons.  The prefixes can be 'network-types' as
         defined in the 'ietf-network' module in RFC 8345, to help the
         user better understand the topology before further inquiry
         is made.";
      reference
        "RFC 8345: A YANG Data Model for Network Topologies";
    }

    typedef te-tp-id {
      type union {
        type uint32;
        // Unnumbered
        type inet:ip-address;
        // IPv4 or IPv6 address
      }
      description
        "An identifier for a TE link endpoint on a node.
         This attribute is mapped to a local or remote link identifier
         as defined in RFCs 3630 and 5305.";
      reference
```

```
        "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
         Version 2
         RFC 5305: IS-IS Extensions for Traffic Engineering";
    }

    typedef bandwidth-scientific-notation {
      type string {
        pattern
          '0(\.0?)?([eE](\+)?0?)?|'
        + '[1-9](\.[0-9]{0,6})?[eE](\+)?(9[0-6]|[1-8][0-9]|0?[0-9])?';
      }
      units "bps";
      description
        "Bandwidth values, expressed using the scientific notation
        in bits per second.
        The encoding format is the external decimal-significant
        character sequences specified in IEEE 754 and ISO/IEC C99
        for 32-bit decimal floating-point numbers:
        (-1)**(S) * 10**(Exponent) * (Significant),
        where Significant uses 7 digits.
        An implementation for this representation may use decimal32
        or binary32. The range of the Exponent is from -95 to +96
        for decimal32, and from -38 to +38 for binary32.
        As a bandwidth value, the format is restricted to be
        normalized, non-negative, and non-fraction:
        n.dddddde{+}dd, N.DDDDDDE{+}DD, 0e0 or 0E0,
        where 'd' and 'D' are decimal digits; 'n' and 'N' are
        non-zeror decimal digits; 'e' and 'E' indicate a power of ten.
        Some examples are 0e0, 1e10, and 9.953e9.";
      reference
        "IEEE Std 754-2008: IEEE Standard for Floating-Point
        Arithmetic.
        ISO/IEC C99: Information technology - Programming
        Languages - C.";
    }

    /* TE features */

    feature p2mp-te {
      description
        "Indicates support for Point-to-Multipoint TE (P2MP-TE).";
      reference
        "RFC 4875: Extensions to Resource Reservation Protocol -
         Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE
         Label Switched Paths (LSPs)";
    }

    feature frr-te {
```

```
      description
        "Indicates support for TE Fast Reroute (FRR).";
      reference
        "RFC 4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels";
    }

    feature extended-admin-groups {
      description
        "Indicates support for TE link extended administrative
         groups.";
      reference
        "RFC 7308: Extended Administrative Groups in MPLS Traffic
         Engineering (MPLS-TE)";
    }

    feature named-path-affinities {
      description
        "Indicates support for named path affinities.";
    }

    feature named-extended-admin-groups {
      description
        "Indicates support for named extended administrative groups.";
    }

    feature named-srlg-groups {
      description
        "Indicates support for named SRLG groups.";
    }

    feature named-path-constraints {
      description
        "Indicates support for named path constraints.";
    }

    feature path-optimization-metric {
      description
        "Indicates support for path optimization metrics.";
    }

    feature path-optimization-objective-function {
      description
        "Indicates support for path optimization objective functions.";
    }

    /*
     * Identities
     */
```

```
identity session-attributes-flags {
  description
    "Base identity for the RSVP-TE session attributes flags.";
}

identity local-protection-desired {
  base session-attributes-flags;
  description
    "Local protection is desired.";
  reference
    "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
     Section 4.7.1";
}

identity se-style-desired {
  base session-attributes-flags;
  description
    "Shared explicit style, to allow the LSP to be established
     and share resources with the old LSP.";
  reference
    "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels";
}

identity local-recording-desired {
  base session-attributes-flags;
  description
    "Label recording is desired.";
  reference
    "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
     Section 4.7.1";
}

identity bandwidth-protection-desired {
  base session-attributes-flags;
  description
    "Requests FRR bandwidth protection on LSRs, if present.";
  reference
    "RFC 4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels";
}

identity node-protection-desired {
  base session-attributes-flags;
  description
    "Requests FRR node protection on LSRs, if present.";
  reference
    "RFC 4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels";
}
```

```
  identity path-reevaluation-request {
    base session-attributes-flags;
    description
      "This flag indicates that a path re-evaluation (of the
       current path in use) is requested.  Note that this does
       not trigger any LSP reroutes but instead just signals a
       request to evaluate whether a preferable path exists.";
    reference
      "RFC 4736: Reoptimization of Multiprotocol Label Switching
       (MPLS) Traffic Engineering (TE) Loosely Routed Label Switched
       Path (LSP)";
  }

  identity soft-preemption-desired {
    base session-attributes-flags;
    description
      "Soft preemption of LSP resources is desired.";
    reference
      "RFC 5712: MPLS Traffic Engineering Soft Preemption";
  }

  identity lsp-attributes-flags {
    description
      "Base identity for LSP attributes flags.";
  }

  identity end-to-end-rerouting-desired {
    base lsp-attributes-flags;
    description
      "Indicates end-to-end rerouting behavior for an LSP
       undergoing establishment.  This MAY also be used to
       specify the behavior of end-to-end LSP recovery for
       established LSPs.";
    reference
      "RFC 4920: Crankback Signaling Extensions for MPLS and GMPLS
       RSVP-TE
       RFC 5420: Encoding of Attributes for MPLS LSP Establishment
       Using Resource Reservation Protocol Traffic Engineering
       (RSVP-TE)
       RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
       Route Object (ERO)";
  }

  identity boundary-rerouting-desired {
    base lsp-attributes-flags;
    description
      "Indicates boundary rerouting behavior for an LSP undergoing
       establishment.  This MAY also be used to specify
```

```
          segment-based LSP recovery through nested crankback for
          established LSPs.  The boundary Area Border Router (ABR) /
          Autonomous System Border Router (ASBR) can decide to forward
          the PathErr message upstream to either an upstream boundary
          ABR/ASBR or the ingress LSR.  Alternatively, it can try to
          select another egress boundary LSR.";
        reference
          "RFC 4920: Crankback Signaling Extensions for MPLS and GMPLS
          RSVP-TE
          RFC 5420: Encoding of Attributes for MPLS LSP Establishment
          Using Resource Reservation Protocol Traffic Engineering
          (RSVP-TE)
          RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
          Route Object (ERO)";
      }

      identity segment-based-rerouting-desired {
        base lsp-attributes-flags;
        description
          "Indicates segment-based rerouting behavior for an LSP
          undergoing establishment.  This MAY also be used to specify
          segment-based LSP recovery for established LSPs.";
        reference
          "RFC 4920: Crankback Signaling Extensions for MPLS and GMPLS
          RSVP-TE
          RFC 5420: Encoding of Attributes for MPLS LSP Establishment
          Using Resource Reservation Protocol Traffic Engineering
          (RSVP-TE)
          RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
          Route Object (ERO)";
      }

      identity lsp-integrity-required {
        base lsp-attributes-flags;
        description
          "Indicates that LSP integrity is required.";
        reference
          "RFC 4875: Extensions to Resource Reservation Protocol -
          Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE
          Label Switched Paths (LSPs)
          RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
          Route Object (ERO)";
      }

      identity contiguous-lsp-desired {
        base lsp-attributes-flags;
        description
          "Indicates that a contiguous LSP is desired.";
```

```
      reference
        "RFC 5151: Inter-Domain MPLS and GMPLS Traffic Engineering --
         Resource Reservation Protocol-Traffic Engineering (RSVP-TE)
         Extensions
         RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
         Route Object (ERO)";
    }

    identity lsp-stitching-desired {
      base lsp-attributes-flags;
      description
        "Indicates that LSP stitching is desired.";
      reference
        "RFC 5150: Label Switched Path Stitching with Generalized
         Multiprotocol Label Switching Traffic Engineering (GMPLS TE)
         RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
         Route Object (ERO)";
    }

    identity pre-planned-lsp-flag {
      base lsp-attributes-flags;
      description
        "Indicates that the LSP MUST be provisioned in the
         control plane only.";
      reference
        "RFC 6001: Generalized MPLS (GMPLS) Protocol Extensions for
         Multi-Layer and Multi-Region Networks (MLN/MRN)
         RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
         Route Object (ERO)";
    }

    identity non-php-behavior-flag {
      base lsp-attributes-flags;
      description
        "Indicates that non-PHP (non-Penultimate Hop Popping) behavior
         for the LSP is desired.";
      reference
        "RFC 6511: Non-Penultimate Hop Popping Behavior and Out-of-Band
         Mapping for RSVP-TE Label Switched Paths
         RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
         Route Object (ERO)";
    }

    identity oob-mapping-flag {
      base lsp-attributes-flags;
      description
        "Indicates that signaling of the egress binding information is
         out of band (e.g., via the Border Gateway Protocol (BGP)).";
```

```
      reference
        "RFC 6511: Non-Penultimate Hop Popping Behavior and Out-of-Band
         Mapping for RSVP-TE Label Switched Paths
         RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
         Route Object (ERO)";
    }

    identity entropy-label-capability {
      base lsp-attributes-flags;
      description
        "Indicates entropy label capability.";
      reference
        "RFC 6790: The Use of Entropy Labels in MPLS Forwarding
         RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
         Route Object (ERO)";
    }

    identity oam-mep-entity-desired {
      base lsp-attributes-flags;
      description
        "OAM Maintenance Entity Group End Point (MEP) entities
         desired.";
      reference
        "RFC 7260: GMPLS RSVP-TE Extensions for Operations,
         Administration, and Maintenance (OAM) Configuration";
    }

    identity oam-mip-entity-desired {
      base lsp-attributes-flags;
      description
        "OAM Maintenance Entity Group Intermediate Points (MIP)
         entities desired.";
      reference
        "RFC 7260: GMPLS RSVP-TE Extensions for Operations,
         Administration, and Maintenance (OAM) Configuration";
    }

    identity srlg-collection-desired {
      base lsp-attributes-flags;
      description
        "SRLG collection desired.";
      reference
        "RFC 7570: Label Switched Path (LSP) Attribute in the Explicit
         Route Object (ERO)
         RFC 8001: RSVP-TE Extensions for Collecting Shared Risk
         Link Group (SRLG) Information";
    }
```

```
  identity loopback-desired {
    base lsp-attributes-flags;
    description
      "This flag indicates that a particular node on the LSP is
       required to enter loopback mode.  This can also be
       used to specify the loopback state of the node.";
    reference
      "RFC 7571: GMPLS RSVP-TE Extensions for Lock Instruct and
       Loopback";
  }

  identity p2mp-te-tree-eval-request {
    base lsp-attributes-flags;
    description
      "P2MP-TE tree re-evaluation request.";
    reference
      "RFC 8149: RSVP Extensions for Reoptimization of Loosely Routed
       Point-to-Multipoint Traffic Engineering Label Switched Paths
       (LSPs)";
  }

  identity rtm-set-desired {
    base lsp-attributes-flags;
    description
      "Residence Time Measurement (RTM) attribute flag requested.";
    reference
      "RFC 8169: Residence Time Measurement in MPLS Networks";
  }

  identity link-protection-type {
    description
      "Base identity for the link protection type.";
  }

  identity link-protection-unprotected {
    base link-protection-type;
    description
      "Unprotected link type.";
    reference
      "RFC 4872: RSVP-TE Extensions in Support of End-to-End
       Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
  }

  identity link-protection-extra-traffic {
    base link-protection-type;
    description
      "Extra-Traffic protected link type.";
    reference
```

```
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity link-protection-shared {
      base link-protection-type;
      description
        "Shared protected link type.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity link-protection-1-for-1 {
      base link-protection-type;
      description
        "One-for-one (1:1) protected link type.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity link-protection-1-plus-1 {
      base link-protection-type;
      description
        "One-plus-one (1+1) protected link type.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity link-protection-enhanced {
      base link-protection-type;
      description
        "A compound link protection type derived from the underlay
         TE tunnel protection configuration supporting the TE link.";
    }

    identity association-type {
      description
        "Base identity for the tunnel association.";
    }

    identity association-type-recovery {
      base association-type;
      description
        "Association type for recovery, used to associate LSPs of the
         same tunnel for recovery.";
```

```
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery
         RFC 6780: RSVP ASSOCIATION Object Extensions";
    }

    identity association-type-resource-sharing {
      base association-type;
      description
        "Association type for resource sharing, used to enable
         resource sharing during make-before-break.";
      reference
        "RFC 4873: GMPLS Segment Recovery
         RFC 6780: RSVP ASSOCIATION Object Extensions";
    }

    identity association-type-double-sided-bidir {
      base association-type;
      description
        "Association type for double-sided bidirectional LSPs,
         used to associate two LSPs of two tunnels that are
         independently configured on either endpoint.";
      reference
        "RFC 7551: RSVP-TE Extensions for Associated Bidirectional
         Label Switched Paths (LSPs)";
    }

    identity association-type-single-sided-bidir {
      base association-type;
      description
        "Association type for single-sided bidirectional LSPs,
         used to associate two LSPs of two tunnels, where one
         tunnel is configured on one side/endpoint and the other
         tunnel is dynamically created on the other endpoint.";
      reference
        "RFC 6780: RSVP ASSOCIATION Object Extensions
         RFC 7551: RSVP-TE Extensions for Associated Bidirectional
         Label Switched Paths (LSPs)";
    }

    identity objective-function-type {
      description
        "Base objective function type.";
    }

    identity of-minimize-cost-path {
      base objective-function-type;
      description
```

```
        "Objective function for minimizing path cost.";
      reference
        "RFC 5541: Encoding of Objective Functions in the Path
         Computation Element Communication Protocol (PCEP)";
    }

    identity of-minimize-load-path {
      base objective-function-type;
      description
        "Objective function for minimizing the load on one or more
         paths.";
      reference
        "RFC 5541: Encoding of Objective Functions in the Path
         Computation Element Communication Protocol (PCEP)";
    }

    identity of-maximize-residual-bandwidth {
      base objective-function-type;
      description
        "Objective function for maximizing residual bandwidth.";
      reference
        "RFC 5541: Encoding of Objective Functions in the Path
         Computation Element Communication Protocol (PCEP)";
    }

    identity of-minimize-agg-bandwidth-consumption {
      base objective-function-type;
      description
        "Objective function for minimizing aggregate bandwidth
         consumption.";
      reference
        "RFC 5541: Encoding of Objective Functions in the Path
         Computation Element Communication Protocol (PCEP)";
    }

    identity of-minimize-load-most-loaded-link {
      base objective-function-type;
      description
        "Objective function for minimizing the load on the link that
         is carrying the highest load.";
      reference
        "RFC 5541: Encoding of Objective Functions in the Path
         Computation Element Communication Protocol (PCEP)";
    }

    identity of-minimize-cost-path-set {
      base objective-function-type;
      description
```

```
        "Objective function for minimizing the cost on a path set.";
      reference
        "RFC 5541: Encoding of Objective Functions in the Path
         Computation Element Communication Protocol (PCEP)";
    }

    identity path-computation-method {
      description
        "Base identity for supported path computation mechanisms.";
    }

    identity path-locally-computed {
      base path-computation-method;
      description
        "Indicates a constrained-path LSP in which the
         path is computed by the local LER.";
      reference
        "RFC 3272: Overview and Principles of Internet Traffic
         Engineering, Section 5.4";
    }

    identity path-externally-queried {
      base path-computation-method;
      description
        "Constrained-path LSP in which the path is obtained by
         querying an external source, such as a PCE server.
         In the case that an LSP is defined to be externally queried,
         it may also have associated explicit definitions (provided
         to the external source to aid computation).  The path that is
         returned by the external source may require further local
         computation on the device.";
      reference
        "RFC 3272: Overview and Principles of Internet Traffic
         Engineering
         RFC 4657: Path Computation Element (PCE) Communication
         Protocol Generic Requirements";
    }

    identity path-explicitly-defined {
      base path-computation-method;
      description
        "Constrained-path LSP in which the path is
         explicitly specified as a collection of strict and/or loose
         hops.";
      reference
        "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels
         RFC 3272: Overview and Principles of Internet Traffic
         Engineering";
```

```
  }

  identity lsp-metric-type {
    description
      "Base identity for the LSP metric specification types.";
  }

  identity lsp-metric-relative {
    base lsp-metric-type;
    description
      "The metric specified for the LSPs to which this identity
       refers is specified as a value relative to the IGP metric
       cost to the LSP's tail end.";
    reference
      "RFC 4657: Path Computation Element (PCE) Communication
       Protocol Generic Requirements";
  }

  identity lsp-metric-absolute {
    base lsp-metric-type;
    description
      "The metric specified for the LSPs to which this identity
       refers is specified as an absolute value.";
    reference
      "RFC 4657: Path Computation Element (PCE) Communication
       Protocol Generic Requirements";
  }

  identity lsp-metric-inherited {
    base lsp-metric-type;
    description
      "The metric for the LSPs to which this identity refers is
       not specified explicitly; rather, it is directly inherited
       from the IGP cost.";
    reference
      "RFC 4657: Path Computation Element (PCE) Communication
       Protocol Generic Requirements";
  }

  identity te-tunnel-type {
    description
      "Base identity from which specific tunnel types are derived.";
  }

  identity te-tunnel-p2p {
    base te-tunnel-type;
    description
      "TE Point-to-Point (P2P) tunnel type.";
```

```
      reference
        "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels";
    }

    identity te-tunnel-p2mp {
      base te-tunnel-type;
      description
        "TE P2MP tunnel type.";
      reference
        "RFC 4875: Extensions to Resource Reservation Protocol -
         Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE
         Label Switched Paths (LSPs)";
    }

    identity tunnel-action-type {
      description
        "Base identity from which specific tunnel action types
         are derived.";
    }

    identity tunnel-action-resetup {
      base tunnel-action-type;
      description
        "TE tunnel action that tears down the tunnel's current LSP
         (if any) and attempts to re-establish a new LSP.";
    }

    identity tunnel-action-reoptimize {
      base tunnel-action-type;
      description
        "TE tunnel action that reoptimizes the placement of the
         tunnel LSP(s).";
    }

    identity tunnel-action-switchpath {
      base tunnel-action-type;
      description
        "TE tunnel action that switches the tunnel's LSP to use the
         specified path.";
    }

    identity te-action-result {
      description
        "Base identity from which specific TE action results
         are derived.";
    }

    identity te-action-success {
```

```
      base te-action-result;
      description
        "TE action was successful.";
    }

    identity te-action-fail {
      base te-action-result;
      description
        "TE action failed.";
    }

    identity tunnel-action-inprogress {
      base te-action-result;
      description
        "TE action is in progress.";
    }

    identity tunnel-admin-state-type {
      description
        "Base identity for TE tunnel administrative states.";
    }

    identity tunnel-admin-state-up {
      base tunnel-admin-state-type;
      description
        "Tunnel's administrative state is up.";
    }

    identity tunnel-admin-state-down {
      base tunnel-admin-state-type;
      description
        "Tunnel's administrative state is down.";
    }

    identity tunnel-state-type {
      description
        "Base identity for TE tunnel states.";
    }

    identity tunnel-state-up {
      base tunnel-state-type;
      description
        "Tunnel's state is up.";
    }

    identity tunnel-state-down {
      base tunnel-state-type;
      description
```

```
        "Tunnel's state is down.";
    }

    identity lsp-state-type {
      description
        "Base identity for TE LSP states.";
    }

    identity lsp-path-computing {
      base lsp-state-type;
      description
        "State path computation is in progress.";
    }

    identity lsp-path-computation-ok {
      base lsp-state-type;
      description
        "State path computation was successful.";
    }

    identity lsp-path-computation-failed {
      base lsp-state-type;
      description
        "State path computation failed.";
    }

    identity lsp-state-setting-up {
      base lsp-state-type;
      description
        "State is being set up.";
    }

    identity lsp-state-setup-ok {
      base lsp-state-type;
      description
        "State setup was successful.";
    }

    identity lsp-state-setup-failed {
      base lsp-state-type;
      description
        "State setup failed.";
    }

    identity lsp-state-up {
      base lsp-state-type;
      description
        "State is up.";
```

```
  }

  identity lsp-state-tearing-down {
    base lsp-state-type;
    description
      "State is being torn down.";
  }

  identity lsp-state-down {
    base lsp-state-type;
    description
      "State is down.";
  }

  identity path-invalidation-action-type {
    description
      "Base identity for TE path invalidation action types.";
  }

  identity path-invalidation-action-drop {
    base path-invalidation-action-type;
    description
      "Upon invalidation of the TE tunnel path, the tunnel remains
       valid, but any packet mapped over the tunnel is dropped.";
    reference
      "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
       Section 2.5";
  }

  identity path-invalidation-action-teardown {
    base path-invalidation-action-type;
    description
      "TE path invalidation action teardown.";
    reference
      "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
       Section 2.5";
  }

  identity lsp-restoration-type {
    description
      "Base identity from which LSP restoration types are derived.";
  }

  identity lsp-restoration-restore-any {
    base lsp-restoration-type;
    description
      "Any LSP affected by a failure is restored.";
  }
```

```
  identity lsp-restoration-restore-all {
    base lsp-restoration-type;
    description
      "Affected LSPs are restored after all LSPs of the tunnel are
       broken.";
  }

  identity restoration-scheme-type {
    description
      "Base identity for LSP restoration schemes.";
  }

  identity restoration-scheme-preconfigured {
    base restoration-scheme-type;
    description
      "Restoration LSP is preconfigured prior to the failure.";
    reference
      "RFC 4427: Recovery (Protection and Restoration) Terminology
       for Generalized Multi-Protocol Label Switching (GMPLS)";
  }

  identity restoration-scheme-precomputed {
    base restoration-scheme-type;
    description
      "Restoration LSP is precomputed prior to the failure.";
    reference
      "RFC 4427: Recovery (Protection and Restoration) Terminology
       for Generalized Multi-Protocol Label Switching (GMPLS)";
  }

  identity restoration-scheme-presignaled {
    base restoration-scheme-type;
    description
      "Restoration LSP is presignaled prior to the failure.";
    reference
      "RFC 4427: Recovery (Protection and Restoration) Terminology
       for Generalized Multi-Protocol Label Switching (GMPLS)";
  }

  identity lsp-protection-type {
    description
      "Base identity from which LSP protection types are derived.";
    reference
      "RFC 4872: RSVP-TE Extensions in Support of End-to-End
       Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
  }

  identity lsp-protection-unprotected {
```

```
      base lsp-protection-type;
      description
        "'Unprotected' LSP protection type.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity lsp-protection-reroute-extra {
      base lsp-protection-type;
      description
        "'(Full) Rerouting' LSP protection type.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity lsp-protection-reroute {
      base lsp-protection-type;
      description
        "'Rerouting without Extra-Traffic' LSP protection type.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity lsp-protection-1-for-n {
      base lsp-protection-type;
      description
        "'1:N Protection with Extra-Traffic' LSP protection type.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity lsp-protection-1-for-1 {
      base lsp-protection-type;
      description
        "LSP protection '1:1 Protection Type'.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity lsp-protection-unidir-1-plus-1 {
      base lsp-protection-type;
      description
        "'1+1 Unidirectional Protection' LSP protection type.";
```

```
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity lsp-protection-bidir-1-plus-1 {
      base lsp-protection-type;
      description
        "'1+1 Bidirectional Protection' LSP protection type.";
      reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS) Recovery";
    }

    identity lsp-protection-extra-traffic {
      base lsp-protection-type;
      description
        "Extra-Traffic LSP protection type.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity lsp-protection-state {
      description
        "Base identity of protection states for reporting purposes.";
    }

    identity normal {
      base lsp-protection-state;
      description
        "Normal state.";
    }

    identity signal-fail-of-protection {
      base lsp-protection-state;
      description
        "The protection transport entity has a signal fail condition
         that is of higher priority than the forced switchover
         command.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity lockout-of-protection {
      base lsp-protection-state;
      description
```

```
        "A Loss of Protection (LoP) command is active.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity forced-switch {
      base lsp-protection-state;
      description
        "A forced switchover command is active.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity signal-fail {
      base lsp-protection-state;
      description
        "There is a signal fail condition on either the working path
         or the protection path.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity signal-degrade {
      base lsp-protection-state;
      description
        "There is a signal degrade condition on either the working
         path or the protection path.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity manual-switch {
      base lsp-protection-state;
      description
        "A manual switchover command is active.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity wait-to-restore {
      base lsp-protection-state;
      description
        "A WTR timer is running.";
```

```
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity do-not-revert {
      base lsp-protection-state;
      description
        "A Do Not Revert (DNR) condition is active because of
         non-revertive behavior.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity failure-of-protocol {
      base lsp-protection-state;
      description
        "LSP protection is not working because of a protocol failure
         condition.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity protection-external-commands {
      description
        "Base identity from which protection-related external commands
         used for troubleshooting purposes are derived.";
    }

    identity action-freeze {
      base protection-external-commands;
      description
        "A temporary configuration action initiated by an operator
         command that prevents any switchover action from being taken
         and, as such, freezes the current state.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity clear-freeze {
      base protection-external-commands;
      description
        "An action that clears the active freeze state.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
```

```
        for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity action-lockout-of-normal {
      base protection-external-commands;
      description
        "A temporary configuration action initiated by an operator
         command to ensure that the normal traffic is not allowed
         to use the protection transport entity.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity clear-lockout-of-normal {
      base protection-external-commands;
      description
        "An action that clears the active lockout of the
         normal state.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity action-lockout-of-protection {
      base protection-external-commands;
      description
        "A temporary configuration action initiated by an operator
         command to ensure that the protection transport entity is
         temporarily not available to transport a traffic signal
         (either normal or Extra-Traffic).";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }

    identity action-forced-switch {
      base protection-external-commands;
      description
        "A switchover action initiated by an operator command to switch
         the Extra-Traffic signal, the normal traffic signal, or the
         null signal to the protection transport entity, unless a
         switchover command of equal or higher priority is in effect.";
      reference
        "RFC 4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS)";
    }
```

```
     identity action-manual-switch {
       base protection-external-commands;
       description
         "A switchover action initiated by an operator command to switch
          the Extra-Traffic signal, the normal traffic signal, or
          the null signal to the protection transport entity, unless
          a fault condition exists on other transport entities or a
          switchover command of equal or higher priority is in effect.";
       reference
         "RFC 4427: Recovery (Protection and Restoration) Terminology
          for Generalized Multi-Protocol Label Switching (GMPLS)";
     }

     identity action-exercise {
       base protection-external-commands;
       description
         "An action that starts testing whether or not APS communication
          is operating correctly.  It is of lower priority than any
          other state or command.";
       reference
         "RFC 4427: Recovery (Protection and Restoration) Terminology
          for Generalized Multi-Protocol Label Switching (GMPLS)";
     }

     identity clear {
       base protection-external-commands;
       description
         "An action that clears the active near-end lockout of a
          protection, forced switchover, manual switchover, WTR state,
          or exercise command.";
       reference
         "RFC 4427: Recovery (Protection and Restoration) Terminology
          for Generalized Multi-Protocol Label Switching (GMPLS)";
     }

     identity switching-capabilities {
       description
         "Base identity for interface switching capabilities.";
       reference
         "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
          Signaling Functional Description";
     }

     identity switching-psc1 {
       base switching-capabilities;
       description
         "Packet-Switch Capable-1 (PSC-1).";
       reference
```

```
        "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
         Signaling Functional Description";
    }

    identity switching-evpl {
      base switching-capabilities;
      description
        "Ethernet Virtual Private Line (EVPL).";
      reference
        "RFC 6004: Generalized MPLS (GMPLS) Support for Metro Ethernet
         Forum and G.8011 Ethernet Service Switching";
    }

    identity switching-l2sc {
      base switching-capabilities;
      description
        "Layer-2 Switch Capable (L2SC).";
      reference
        "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
         Signaling Functional Description";
    }

    identity switching-tdm {
      base switching-capabilities;
      description
        "Time-Division-Multiplex Capable (TDM).";
      reference
        "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
         Signaling Functional Description";
    }

    identity switching-otn {
      base switching-capabilities;
      description
        "OTN-TDM capable.";
      reference
        "RFC 7138: Traffic Engineering Extensions to OSPF for GMPLS
         Control of Evolving G.709 Optical Transport Networks";
    }

    identity switching-dcsc {
      base switching-capabilities;
      description
        "Data Channel Switching Capable (DCSC).";
      reference
        "RFC 6002: Generalized MPLS (GMPLS) Data Channel
         Switching Capable (DCSC) and Channel Set Label Extensions";
    }
```

```
     identity switching-lsc {
       base switching-capabilities;
       description
         "Lambda-Switch Capable (LSC).";
       reference
         "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
          Signaling Functional Description";
     }

     identity switching-fsc {
       base switching-capabilities;
       description
         "Fiber-Switch Capable (FSC).";
       reference
         "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
          Signaling Functional Description";
     }

     identity lsp-encoding-types {
       description
         "Base identity for encoding types.";
       reference
         "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
          Signaling Functional Description";
     }

     identity lsp-encoding-packet {
       base lsp-encoding-types;
       description
         "Packet LSP encoding.";
       reference
         "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
          Signaling Functional Description";
     }

     identity lsp-encoding-ethernet {
       base lsp-encoding-types;
       description
         "Ethernet LSP encoding.";
       reference
         "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
          Signaling Functional Description";
     }

     identity lsp-encoding-pdh {
       base lsp-encoding-types;
       description
         "ANSI/ETSI PDH LSP encoding.";
```

```
      reference
        "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
         Signaling Functional Description";
    }

    identity lsp-encoding-sdh {
      base lsp-encoding-types;
      description
        "SDH ITU-T G.707 / SONET ANSI T1.105 LSP encoding.";
      reference
        "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
         Signaling Functional Description";
    }

    identity lsp-encoding-digital-wrapper {
      base lsp-encoding-types;
      description
        "Digital Wrapper LSP encoding.";
      reference
        "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
         Signaling Functional Description";
    }

    identity lsp-encoding-lambda {
      base lsp-encoding-types;
      description
        "Lambda (photonic) LSP encoding.";
      reference
        "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
         Signaling Functional Description";
    }

    identity lsp-encoding-fiber {
      base lsp-encoding-types;
      description
        "Fiber LSP encoding.";
      reference
        "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
         Signaling Functional Description";
    }

    identity lsp-encoding-fiber-channel {
      base lsp-encoding-types;
      description
        "FiberChannel LSP encoding.";
      reference
        "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
         Signaling Functional Description";
```

```
  }

  identity lsp-encoding-oduk {
    base lsp-encoding-types;
    description
      "G.709 ODUk (Digital Path) LSP encoding.";
    reference
      "RFC 4328: Generalized Multi-Protocol Label Switching (GMPLS)
       Signaling Extensions for G.709 Optical Transport Networks
       Control";
  }

  identity lsp-encoding-optical-channel {
    base lsp-encoding-types;
    description
      "G.709 Optical Channel LSP encoding.";
    reference
      "RFC 4328: Generalized Multi-Protocol Label Switching (GMPLS)
       Signaling Extensions for G.709 Optical Transport Networks
       Control";
  }

  identity lsp-encoding-line {
    base lsp-encoding-types;
    description
      "Line (e.g., 8B/10B) LSP encoding.";
    reference
      "RFC 6004: Generalized MPLS (GMPLS) Support for Metro
       Ethernet Forum and G.8011 Ethernet Service Switching";
  }

  identity path-signaling-type {
    description
      "Base identity from which specific LSP path setup types
       are derived.";
  }

  identity path-setup-static {
    base path-signaling-type;
    description
      "Static LSP provisioning path setup.";
  }

  identity path-setup-rsvp {
    base path-signaling-type;
    description
      "RSVP-TE signaling path setup.";
    reference
```

```
        "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels";
    }

    identity path-setup-sr {
      base path-signaling-type;
      description
        "Segment-routing path setup.";
    }

    identity path-scope-type {
      description
        "Base identity from which specific path scope types are
         derived.";
    }

    identity path-scope-segment {
      base path-scope-type;
      description
        "Path scope segment.";
      reference
        "RFC 4873: GMPLS Segment Recovery";
    }

    identity path-scope-end-to-end {
      base path-scope-type;
      description
        "Path scope end to end.";
      reference
        "RFC 4873: GMPLS Segment Recovery";
    }

    identity route-usage-type {
      description
        "Base identity for route usage.";
    }

    identity route-include-object {
      base route-usage-type;
      description
        "'Include route' object.";
    }

    identity route-exclude-object {
      base route-usage-type;
      description
        "'Exclude route' object.";
      reference
        "RFC 4874: Exclude Routes - Extension to Resource ReserVation
```

```
          Protocol-Traffic Engineering (RSVP-TE)";
      }

      identity route-exclude-srlg {
        base route-usage-type;
        description
          "Excludes SRLGs.";
        reference
          "RFC 4874: Exclude Routes - Extension to Resource ReserVation
           Protocol-Traffic Engineering (RSVP-TE)";
      }

      identity path-metric-type {
        description
          "Base identity for the path metric type.";
      }

      identity path-metric-te {
        base path-metric-type;
        description
          "TE path metric.";
        reference
          "RFC 3785: Use of Interior Gateway Protocol (IGP) Metric as a
           second MPLS Traffic Engineering (TE) Metric";
      }

      identity path-metric-igp {
        base path-metric-type;
        description
          "IGP path metric.";
        reference
          "RFC 3785: Use of Interior Gateway Protocol (IGP) Metric as a
           second MPLS Traffic Engineering (TE) Metric";
      }

      identity path-metric-hop {
        base path-metric-type;
        description
          "Hop path metric.";
      }

      identity path-metric-delay-average {
        base path-metric-type;
        description
          "Average unidirectional link delay.";
        reference
          "RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions";
      }
```

```
  identity path-metric-delay-minimum {
    base path-metric-type;
    description
      "Minimum unidirectional link delay.";
    reference
      "RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions";
  }

  identity path-metric-residual-bandwidth {
    base path-metric-type;
    description
      "Unidirectional Residual Bandwidth, which is defined to be
       Maximum Bandwidth (RFC 3630) minus the bandwidth currently
       allocated to LSPs.";
    reference
      "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
       Version 2
       RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions";
  }

  identity path-metric-optimize-includes {
    base path-metric-type;
    description
      "A metric that optimizes the number of included resources
       specified in a set.";
  }

  identity path-metric-optimize-excludes {
    base path-metric-type;
    description
      "A metric that optimizes to a maximum the number of excluded
       resources specified in a set.";
  }

  identity path-tiebreaker-type {
    description
      "Base identity for the path tiebreaker type.";
  }

  identity path-tiebreaker-minfill {
    base path-tiebreaker-type;
    description
      "Min-Fill LSP path placement.";
  }

  identity path-tiebreaker-maxfill {
    base path-tiebreaker-type;
    description
```

```
        "Max-Fill LSP path placement.";
    }

    identity path-tiebreaker-random {
      base path-tiebreaker-type;
      description
        "Random LSP path placement.";
    }

    identity resource-affinities-type {
      description
        "Base identity for resource class affinities.";
      reference
        "RFC 2702: Requirements for Traffic Engineering Over MPLS";
    }

    identity resource-aff-include-all {
      base resource-affinities-type;
      description
        "The set of attribute filters associated with a
         tunnel, all of which must be present for a link
         to be acceptable.";
      reference
        "RFC 2702: Requirements for Traffic Engineering Over MPLS
         RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels";
    }

    identity resource-aff-include-any {
      base resource-affinities-type;
      description
        "The set of attribute filters associated with a
         tunnel, any of which must be present for a link
         to be acceptable.";
      reference
        "RFC 2702: Requirements for Traffic Engineering Over MPLS
         RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels";
    }

    identity resource-aff-exclude-any {
      base resource-affinities-type;
      description
        "The set of attribute filters associated with a
         tunnel, any of which renders a link unacceptable.";
      reference
        "RFC 2702: Requirements for Traffic Engineering Over MPLS
         RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels";
    }
```

```
identity te-optimization-criterion {
  description
    "Base identity for the TE optimization criteria.";
  reference
    "RFC 3272: Overview and Principles of Internet Traffic
     Engineering";
}

identity not-optimized {
  base te-optimization-criterion;
  description
    "Optimization is not applied.";
}

identity cost {
  base te-optimization-criterion;
  description
    "Optimized on cost.";
  reference
    "RFC 5541: Encoding of Objective Functions in the Path
     Computation Element Communication Protocol (PCEP)";
}

identity delay {
  base te-optimization-criterion;
  description
    "Optimized on delay.";
  reference
    "RFC 5541: Encoding of Objective Functions in the Path
     Computation Element Communication Protocol (PCEP)";
}

identity path-computation-srlg-type {
  description
    "Base identity for SRLG path computation.";
}

identity srlg-ignore {
  base path-computation-srlg-type;
  description
    "Ignores SRLGs in the path computation.";
}

identity srlg-strict {
  base path-computation-srlg-type;
  description
    "Includes a strict SRLG check in the path computation.";
}
```

```
identity srlg-preferred {
  base path-computation-srlg-type;
  description
    "Includes a preferred SRLG check in the path computation.";
}

identity srlg-weighted {
  base path-computation-srlg-type;
  description
    "Includes a weighted SRLG check in the path computation.";
}

/**
 * TE bandwidth groupings
 **/

grouping te-bandwidth {
  description
    "This grouping defines the generic TE bandwidth.
     For some known data-plane technologies, specific modeling
     structures are specified.  The string-encoded 'te-bandwidth'
     type is used for unspecified technologies.
     The modeling structure can be augmented later for other
     technologies.";
  container te-bandwidth {
    description
      "Container that specifies TE bandwidth.  The choices
       can be augmented for specific data-plane technologies.";
    choice technology {
      default "generic";
      description
        "Data-plane technology type.";
      case generic {
        leaf generic {
          type te-bandwidth;
          description
            "Bandwidth specified in a generic format.";
        }
      }
    }
  }
}

/**
 * TE label groupings
 **/

grouping te-label {
```

```
      description
        "This grouping defines the generic TE label.
         The modeling structure can be augmented for each technology.
         For unspecified technologies, 'rt-types:generalized-label'
         is used.";
      container te-label {
        description
          "Container that specifies the TE label.  The choices can
           be augmented for specific data-plane technologies.";
        choice technology {
          default "generic";
          description
            "Data-plane technology type.";
          case generic {
            leaf generic {
              type rt-types:generalized-label;
              description
                "TE label specified in a generic format.";
            }
          }
        }
        leaf direction {
          type te-label-direction;
          default "forward";
          description
            "Label direction.";
        }
      }
    }

    grouping te-topology-identifier {
      description
        "Augmentation for a TE topology.";
      container te-topology-identifier {
        description
          "TE topology identifier container.";
        leaf provider-id {
          type te-global-id;
          default "0";
          description
            "An identifier to uniquely identify a provider.
             If omitted, it assumes that the topology provider ID
             value = 0 (the default).";
        }
        leaf client-id {
          type te-global-id;
          default "0";
          description
```

```
            "An identifier to uniquely identify a client.
             If omitted, it assumes that the topology client ID
             value = 0 (the default).";
        }
        leaf topology-id {
          type te-topology-id;
          default "";
          description
            "When the datastore contains several topologies,
             'topology-id' distinguishes between them.  If omitted,
             the default (empty) string for this leaf is assumed.";
        }
      }
    }

    /**
     * TE performance metrics groupings
     **/

    grouping performance-metrics-one-way-delay-loss {
      description
        "Performance Metrics (PM) information in real time that can
         be applicable to links or connections.  PM defined in this
         grouping are applicable to generic TE PM as well as packet TE
         PM.";
      reference
        "RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions
         RFC 7823: Performance-Based Path Selection for Explicitly
         Routed Label Switched Paths (LSPs) Using TE Metric
         Extensions
         RFC 8570: IS-IS Traffic Engineering (TE) Metric Extensions";
      leaf one-way-delay {
        type uint32 {
          range "0..16777215";
        }
        description
          "One-way delay or latency in microseconds.";
      }
      leaf one-way-delay-normality {
        type te-types:performance-metrics-normality;
        description
          "One-way delay normality.";
      }
    }

    grouping performance-metrics-two-way-delay-loss {
      description
        "PM information in real time that can be applicable to links or
```

```
          connections.  PM defined in this grouping are applicable to
          generic TE PM as well as packet TE PM.";
        reference
          "RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions
           RFC 7823: Performance-Based Path Selection for Explicitly
           Routed Label Switched Paths (LSPs) Using TE Metric
           Extensions
           RFC 8570: IS-IS Traffic Engineering (TE) Metric Extensions";
        leaf two-way-delay {
          type uint32 {
            range "0..16777215";
          }
          description
            "Two-way delay or latency in microseconds.";
        }
        leaf two-way-delay-normality {
          type te-types:performance-metrics-normality;
          description
            "Two-way delay normality.";
        }
      }

      grouping performance-metrics-one-way-bandwidth {
        description
          "PM information in real time that can be applicable to links.
           PM defined in this grouping are applicable to generic TE PM
           as well as packet TE PM.";
        reference
          "RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions
           RFC 7823: Performance-Based Path Selection for Explicitly
           Routed Label Switched Paths (LSPs) Using TE Metric
           Extensions
           RFC 8570: IS-IS Traffic Engineering (TE) Metric Extensions";
        leaf one-way-residual-bandwidth {
          type rt-types:bandwidth-ieee-float32;
          units "bytes per second";
          default "0x0p0";
          description
            "Residual bandwidth that subtracts tunnel reservations from
             Maximum Bandwidth (or link capacity) (RFC 3630) and
             provides an aggregated remainder across QoS classes.";
          reference
            "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
             Version 2";
        }
        leaf one-way-residual-bandwidth-normality {
          type te-types:performance-metrics-normality;
          default "normal";
```

```
        description
          "Residual bandwidth normality.";
      }
      leaf one-way-available-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        units "bytes per second";
        default "0x0p0";
        description
          "Available bandwidth that is defined to be residual
           bandwidth minus the measured bandwidth used for the
           actual forwarding of non-RSVP-TE LSP packets.  For a
           bundled link, available bandwidth is defined to be the
           sum of the component link available bandwidths.";
      }
      leaf one-way-available-bandwidth-normality {
        type te-types:performance-metrics-normality;
        default "normal";
        description
          "Available bandwidth normality.";
      }
      leaf one-way-utilized-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        units "bytes per second";
        default "0x0p0";
        description
          "Bandwidth utilization that represents the actual
           utilization of the link (i.e., as measured in the router).
           For a bundled link, bandwidth utilization is defined to
           be the sum of the component link bandwidth utilizations.";
      }
      leaf one-way-utilized-bandwidth-normality {
        type te-types:performance-metrics-normality;
        default "normal";
        description
          "Bandwidth utilization normality.";
      }
    }

    grouping one-way-performance-metrics {
      description
        "One-way PM throttle grouping.";
      leaf one-way-delay {
        type uint32 {
          range "0..16777215";
        }
        default "0";
        description
          "One-way delay or latency in microseconds.";
```

```
      }
      leaf one-way-residual-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        units "bytes per second";
        default "0x0p0";
        description
          "Residual bandwidth that subtracts tunnel reservations from
           Maximum Bandwidth (or link capacity) (RFC 3630) and
           provides an aggregated remainder across QoS classes.";
        reference
          "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
           Version 2";
      }
      leaf one-way-available-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        units "bytes per second";
        default "0x0p0";
        description
          "Available bandwidth that is defined to be residual
           bandwidth minus the measured bandwidth used for the
           actual forwarding of non-RSVP-TE LSP packets.  For a
           bundled link, available bandwidth is defined to be the
           sum of the component link available bandwidths.";
      }
      leaf one-way-utilized-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        units "bytes per second";
        default "0x0p0";
        description
          "Bandwidth utilization that represents the actual
           utilization of the link (i.e., as measured in the router).
           For a bundled link, bandwidth utilization is defined to
           be the sum of the component link bandwidth utilizations.";
      }
    }

    grouping two-way-performance-metrics {
      description
        "Two-way PM throttle grouping.";
      leaf two-way-delay {
        type uint32 {
          range "0..16777215";
        }
        default "0";
        description
          "Two-way delay or latency in microseconds.";
      }
    }
```

```
    grouping performance-metrics-thresholds {
      description
        "Grouping for configurable thresholds for measured
         attributes.";
      uses one-way-performance-metrics;
      uses two-way-performance-metrics;
    }

    grouping performance-metrics-attributes {
      description
        "Contains PM attributes.";
      container performance-metrics-one-way {
        description
          "One-way link performance information in real time.";
        reference
          "RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions
           RFC 7823: Performance-Based Path Selection for Explicitly
           Routed Label Switched Paths (LSPs) Using TE Metric
           Extensions
           RFC 8570: IS-IS Traffic Engineering (TE) Metric Extensions";
        uses performance-metrics-one-way-delay-loss;
        uses performance-metrics-one-way-bandwidth;
      }
      container performance-metrics-two-way {
        description
          "Two-way link performance information in real time.";
        reference
          "RFC 6374: Packet Loss and Delay Measurement for MPLS
           Networks";
        uses performance-metrics-two-way-delay-loss;
      }
    }

    grouping performance-metrics-throttle-container {
      description
        "Controls PM throttling.";
      container throttle {
        must 'suppression-interval >= measure-interval' {
          error-message "'suppression-interval' cannot be less than "
                      + "'measure-interval'.";
          description
            "Constraint on 'suppression-interval' and
             'measure-interval'.";
        }
        description
          "Link performance information in real time.";
        reference
          "RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions
```

```
           RFC 7823: Performance-Based Path Selection for Explicitly
           Routed Label Switched Paths (LSPs) Using TE Metric
           Extensions
           RFC 8570: IS-IS Traffic Engineering (TE) Metric Extensions";
        leaf one-way-delay-offset {
          type uint32 {
            range "0..16777215";
          }
          default "0";
          description
            "Offset value to be added to the measured delay value.";
        }
        leaf measure-interval {
          type uint32;
          default "30";
          description
            "Interval, in seconds, to measure the extended metric
             values.";
        }
        leaf advertisement-interval {
          type uint32;
          default "0";
          description
            "Interval, in seconds, to advertise the extended metric
             values.";
        }
        leaf suppression-interval {
          type uint32 {
            range "1..max";
          }
          default "120";
          description
            "Interval, in seconds, to suppress advertisement of the
             extended metric values.";
          reference
            "RFC 8570: IS-IS Traffic Engineering (TE) Metric
             Extensions, Section 6";
        }
        container threshold-out {
          uses performance-metrics-thresholds;
          description
            "If the measured parameter falls outside an upper bound
             for all but the minimum-delay metric (or a lower bound
             for the minimum-delay metric only) and the advertised
             value is not already outside that bound, an 'anomalous'
             announcement (anomalous bit set) will be triggered.";
        }
        container threshold-in {
```

```
          uses performance-metrics-thresholds;
          description
            "If the measured parameter falls inside an upper bound
             for all but the minimum-delay metric (or a lower bound
             for the minimum-delay metric only) and the advertised
             value is not already inside that bound, a 'normal'
             announcement (anomalous bit cleared) will be triggered.";
        }
        container threshold-accelerated-advertisement {
          description
            "When the difference between the last advertised value and
             the current measured value exceeds this threshold, an
             'anomalous' announcement (anomalous bit set) will be
             triggered.";
          uses performance-metrics-thresholds;
        }
      }
    }

    /**
     * TE tunnel generic groupings
     **/

    grouping explicit-route-hop {
      description
        "The explicit route entry grouping.";
      choice type {
        description
          "The explicit route entry type.";
        case numbered-node-hop {
          container numbered-node-hop {
            leaf node-id {
              type te-node-id;
              mandatory true;
              description
                "The identifier of a node in the TE topology.";
            }
            leaf hop-type {
              type te-hop-type;
              default "strict";
              description
                "Strict or loose hop.";
            }
            description
              "Numbered node route hop.";
            reference
              "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
               Section 4.3, EXPLICIT_ROUTE in RSVP-TE
```

```
              RFC 3477: Signalling Unnumbered Links in Resource
              ReSerVation Protocol - Traffic Engineering (RSVP-TE)";
          }
        }
        case numbered-link-hop {
          container numbered-link-hop {
            leaf link-tp-id {
              type te-tp-id;
              mandatory true;
              description
                "TE Link Termination Point (LTP) identifier.";
            }
            leaf hop-type {
              type te-hop-type;
              default "strict";
              description
                "Strict or loose hop.";
            }
            leaf direction {
              type te-link-direction;
              default "outgoing";
              description
                "Link route object direction.";
            }
            description
              "Numbered link explicit route hop.";
            reference
              "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
               Section 4.3, EXPLICIT_ROUTE in RSVP-TE
               RFC 3477: Signalling Unnumbered Links in Resource
               ReSerVation Protocol - Traffic Engineering (RSVP-TE)";
          }
        }
        case unnumbered-link-hop {
          container unnumbered-link-hop {
            leaf link-tp-id {
              type te-tp-id;
              mandatory true;
              description
                "TE LTP identifier.  The combination of the TE link ID
                 and the TE node ID is used to identify an unnumbered
                 TE link.";
            }
            leaf node-id {
              type te-node-id;
              mandatory true;
              description
                "The identifier of a node in the TE topology.";
```

```
            }
            leaf hop-type {
              type te-hop-type;
              default "strict";
              description
                "Strict or loose hop.";
            }
            leaf direction {
              type te-link-direction;
              default "outgoing";
              description
                "Link route object direction.";
            }
            description
              "Unnumbered link explicit route hop.";
            reference
              "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
               Section 4.3, EXPLICIT_ROUTE in RSVP-TE
               RFC 3477: Signalling Unnumbered Links in Resource
               ReSerVation Protocol - Traffic Engineering (RSVP-TE)";
          }
        }
        case as-number {
          container as-number-hop {
            leaf as-number {
              type inet:as-number;
              mandatory true;
              description
                "The Autonomous System (AS) number.";
            }
            leaf hop-type {
              type te-hop-type;
              default "strict";
              description
                "Strict or loose hop.";
            }
            description
              "AS explicit route hop.";
          }
        }
        case label {
          container label-hop {
            description
              "Label hop type.";
            uses te-label;
          }
          description
            "The label explicit route hop type.";
```

```
        }
      }
    }

    grouping record-route-state {
      description
        "The Record Route grouping.";
      leaf index {
        type uint32;
        description
          "Record Route hop index.  The index is used to
           identify an entry in the list.  The order of entries
           is defined by the user without relying on key values.";
      }
      choice type {
        description
          "The Record Route entry type.";
        case numbered-node-hop {
          container numbered-node-hop {
            description
              "Numbered node route hop container.";
            leaf node-id {
              type te-node-id;
              mandatory true;
              description
                "The identifier of a node in the TE topology.";
            }
            leaf-list flags {
              type path-attribute-flags;
              description
                "Path attributes flags.";
              reference
                "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels
                 RFC 4090: Fast Reroute Extensions to RSVP-TE for LSP
                 Tunnels
                 RFC 4561: Definition of a Record Route Object (RRO)
                 Node-Id Sub-Object";
            }
          }
          description
            "Numbered node route hop.";
        }
        case numbered-link-hop {
          container numbered-link-hop {
            description
              "Numbered link route hop container.";
            leaf link-tp-id {
              type te-tp-id;
```

```
            mandatory true;
            description
              "Numbered TE LTP identifier.";
          }
          leaf-list flags {
            type path-attribute-flags;
            description
              "Path attributes flags.";
            reference
              "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels
               RFC 4090: Fast Reroute Extensions to RSVP-TE for LSP
               Tunnels
               RFC 4561: Definition of a Record Route Object (RRO)
               Node-Id Sub-Object";
          }
        }
        description
          "Numbered link route hop.";
      }
      case unnumbered-link-hop {
        container unnumbered-link-hop {
          leaf link-tp-id {
            type te-tp-id;
            mandatory true;
            description
              "TE LTP identifier.  The combination of the TE link ID
               and the TE node ID is used to identify an unnumbered
               TE link.";
          }
          leaf node-id {
            type te-node-id;
            description
              "The identifier of a node in the TE topology.";
          }
          leaf-list flags {
            type path-attribute-flags;
            description
              "Path attributes flags.";
            reference
              "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels
               RFC 4090: Fast Reroute Extensions to RSVP-TE for LSP
               Tunnels
               RFC 4561: Definition of a Record Route Object (RRO)
               Node-Id Sub-Object";
          }
          description
            "Unnumbered link Record Route hop.";
          reference
```

```
              "RFC 3477: Signalling Unnumbered Links in Resource
               ReSerVation Protocol - Traffic Engineering (RSVP-TE)";
          }
          description
            "Unnumbered link route hop.";
        }
        case label {
          container label-hop {
            description
              "Label route hop type.";
            uses te-label;
            leaf-list flags {
              type path-attribute-flags;
              description
                "Path attributes flags.";
              reference
                "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels
                 RFC 4090: Fast Reroute Extensions to RSVP-TE for LSP
                 Tunnels
                 RFC 4561: Definition of a Record Route Object (RRO)
                 Node-Id Sub-Object";
            }
          }
          description
            "The label Record Route entry types.";
        }
      }
    }

    grouping label-restriction-info {
      description
        "Label set item information.";
      leaf restriction {
        type enumeration {
          enum inclusive {
            description
              "The label or label range is inclusive.";
          }
          enum exclusive {
            description
              "The label or label range is exclusive.";
          }
        }
        default "inclusive";
        description
          "Indicates whether the list item is inclusive or exclusive.";
      }
      leaf index {
```

```
        type uint32;
        description
          "The index of the label restriction list entry.";
      }
      container label-start {
        must "(not(../label-end/te-label/direction) and"
           + " not(te-label/direction))"
           + " or "
           + "(../label-end/te-label/direction = te-label/direction)"
           + " or "
           + "(not(te-label/direction) and"
           + " (../label-end/te-label/direction = 'forward'))"
           + " or "
           + "(not(../label-end/te-label/direction) and"
           + " (te-label/direction = 'forward'))" {
          error-message "'label-start' and 'label-end' must have the "
                      + "same direction.";
        }
        description
          "This is the starting label if a label range is specified.
           This is the label value if a single label is specified,
           in which case the 'label-end' attribute is not set.";
        uses te-label;
      }
      container label-end {
        must "(not(../label-start/te-label/direction) and"
           + " not(te-label/direction))"
           + " or "
           + "(../label-start/te-label/direction = te-label/direction)"
           + " or "
           + "(not(te-label/direction) and"
           + " (../label-start/te-label/direction = 'forward'))"
           + " or "
           + "(not(../label-start/te-label/direction) and"
           + " (te-label/direction = 'forward'))" {
          error-message "'label-start' and 'label-end' must have the "
                      + "same direction.";
        }
        description
          "This is the ending label if a label range is specified.
           This attribute is not set if a single label is specified.";
        uses te-label;
      }
      container label-step {
        description
          "The step increment between labels in the label range.
           The label start/end values will have to be consistent
           with the sign of label step.  For example,
```

```
           'label-start' < 'label-end' enforces 'label-step' > 0
           'label-start' > 'label-end' enforces 'label-step' < 0.";
        choice technology {
          default "generic";
          description
            "Data-plane technology type.";
          case generic {
            leaf generic {
              type int32;
              default "1";
              description
                "Label range step.";
            }
          }
        }
      }
      leaf range-bitmap {
        type yang:hex-string;
        description
          "When there are gaps between 'label-start' and 'label-end',
           this attribute is used to specify the positions
           of the used labels.  This is represented in big endian as
           'hex-string'.
           The most significant byte in the hex-string is the farthest
           to the left in the byte sequence.  Leading zero bytes in the
           configured value may be omitted for brevity.
           Each bit position in the 'range-bitmap' 'hex-string' maps
           to a label in the range derived from 'label-start'.

           For example, assuming that 'label-start' = 16000 and
           'range-bitmap' = 0x01000001, then:

           - bit position (0) is set, and the corresponding mapped
             label from the range is 16000 + (0 * 'label-step') or
             16000 for default 'label-step' = 1.
           - bit position (24) is set, and the corresponding mapped
             label from the range is 16000 + (24 * 'label-step') or
             16024 for default 'label-step' = 1.";
      }
    }

    grouping label-set-info {
      description
        "Grouping for the list of label restrictions specifying what
         labels may or may not be used.";
      container label-restrictions {
        description
          "The label restrictions container.";
```

```
        list label-restriction {
          key "index";
          description
            "The absence of the label restrictions container implies
             that all labels are acceptable; otherwise, only restricted
             labels are available.";
          reference
            "RFC 7579: General Network Element Constraint Encoding
             for GMPLS-Controlled Networks";
          uses label-restriction-info;
        }
      }
    }

    grouping optimization-metric-entry {
      description
        "Optimization metrics configuration grouping.";
      leaf metric-type {
        type identityref {
          base path-metric-type;
        }
        description
          "Identifies the 'metric-type' that the path computation
           process uses for optimization.";
      }
      leaf weight {
        type uint8;
        default "1";
        description
          "TE path metric normalization weight.";
      }
      container explicit-route-exclude-objects {
        when "../metric-type = "
           + "'te-types:path-metric-optimize-excludes'";
        description
          "Container for the 'exclude route' object list.";
        uses path-route-exclude-objects;
      }
      container explicit-route-include-objects {
        when "../metric-type = "
           + "'te-types:path-metric-optimize-includes'";
        description
          "Container for the 'include route' object list.";
        uses path-route-include-objects;
      }
    }

    grouping common-constraints {
```

```
      description
        "Common constraints grouping that can be set on
         a constraint set or directly on the tunnel.";
      uses te-bandwidth {
        description
          "A requested bandwidth to use for path computation.";
      }
      leaf link-protection {
        type identityref {
          base link-protection-type;
        }
        default "te-types:link-protection-unprotected";
        description
          "Link protection type required for the links included
           in the computed path.";
        reference
          "RFC 4202: Routing Extensions in Support of
           Generalized Multi-Protocol Label Switching (GMPLS)";
      }
      leaf setup-priority {
        type uint8 {
          range "0..7";
        }
        default "7";
        description
          "TE LSP requested setup priority.";
        reference
          "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels";
      }
      leaf hold-priority {
        type uint8 {
          range "0..7";
        }
        default "7";
        description
          "TE LSP requested hold priority.";
        reference
          "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels";
      }
      leaf signaling-type {
        type identityref {
          base path-signaling-type;
        }
        default "te-types:path-setup-rsvp";
        description
          "TE tunnel path signaling type.";
      }
    }
```

```
    grouping tunnel-constraints {
      description
        "Tunnel constraints grouping that can be set on
         a constraint set or directly on the tunnel.";
      uses te-topology-identifier;
      uses common-constraints;
    }

    grouping path-constraints-route-objects {
      description
        "List of route entries to be included or excluded when
         performing the path computation.";
      container explicit-route-objects-always {
        description
          "Container for the 'exclude route' object list.";
        list route-object-exclude-always {
          key "index";
          ordered-by user;
          description
            "List of route objects to always exclude from the path
             computation.";
          leaf index {
            type uint32;
            description
              "Explicit Route Object index.  The index is used to
               identify an entry in the list.  The order of entries
               is defined by the user without relying on key values.";
          }
          uses explicit-route-hop;
        }
        list route-object-include-exclude {
          key "index";
          ordered-by user;
          description
            "List of route objects to include or exclude in the path
             computation.";
          leaf explicit-route-usage {
            type identityref {
              base route-usage-type;
            }
            default "te-types:route-include-object";
            description
              "Indicates whether to include or exclude the
               route object.  The default is to include it.";
          }
          leaf index {
            type uint32;
            description
```

```
              "Route object include-exclude index.  The index is used
               to identify an entry in the list.  The order of entries
               is defined by the user without relying on key values.";
            }
            uses explicit-route-hop {
              augment "type" {
                case srlg {
                  container srlg {
                    description
                      "SRLG container.";
                    leaf srlg {
                      type uint32;
                      description
                        "SRLG value.";
                    }
                  }
                  description
                    "An SRLG value to be included or excluded.";
                }
                description
                  "Augmentation for a generic explicit route for SRLG
                   exclusion.";
              }
            }
          }
        }
      }

      grouping path-route-include-objects {
        description
          "List of route objects to be included when performing
           the path computation.";
        list route-object-include-object {
          key "index";
          ordered-by user;
          description
            "List of Explicit Route Objects to be included in the
             path computation.";
          leaf index {
            type uint32;
            description
              "Route object entry index.  The index is used to
               identify an entry in the list.  The order of entries
               is defined by the user without relying on key values.";
          }
          uses explicit-route-hop;
        }
      }
```

```
    grouping path-route-exclude-objects {
      description
        "List of route objects to be excluded when performing
         the path computation.";
      list route-object-exclude-object {
        key "index";
        ordered-by user;
        description
          "List of Explicit Route Objects to be excluded in the
           path computation.";
        leaf index {
          type uint32;
          description
            "Route object entry index.  The index is used to
             identify an entry in the list.  The order of entries
             is defined by the user without relying on key values.";
        }
        uses explicit-route-hop {
          augment "type" {
            case srlg {
              container srlg {
                description
                  "SRLG container.";
                leaf srlg {
                  type uint32;
                  description
                    "SRLG value.";
                }
              }
              description
                "An SRLG value to be included or excluded.";
            }
            description
              "Augmentation for a generic explicit route for SRLG
               exclusion.";
          }
        }
      }
    }

    grouping generic-path-metric-bounds {
      description
        "TE path metric bounds grouping.";
      container path-metric-bounds {
        description
          "TE path metric bounds container.";
        list path-metric-bound {
          key "metric-type";
```

```
          description
            "List of TE path metric bounds.";
          leaf metric-type {
            type identityref {
              base path-metric-type;
            }
            description
              "Identifies an entry in the list of 'metric-type' items
               bound for the TE path.";
          }
          leaf upper-bound {
            type uint64;
            default "0";
            description
              "Upper bound on the end-to-end TE path metric.  A zero
               indicates an unbounded upper limit for the specific
               'metric-type'.";
          }
        }
      }
    }

    grouping generic-path-optimization {
      description
        "TE generic path optimization grouping.";
      container optimizations {
        description
          "The objective function container that includes
           attributes to impose when computing a TE path.";
        choice algorithm {
          description
            "Optimizations algorithm.";
          case metric {
            if-feature "path-optimization-metric";
            /* Optimize by metric */
            list optimization-metric {
              key "metric-type";
              description
                "TE path metric type.";
              uses optimization-metric-entry;
            }
            /* Tiebreakers */
            container tiebreakers {
              description
                "Container for the list of tiebreakers.";
              list tiebreaker {
                key "tiebreaker-type";
                description
```

```
                  "The list of tiebreaker criteria to apply on an
                   equally favored set of paths, in order to pick
                   the best.";
                leaf tiebreaker-type {
                  type identityref {
                    base path-metric-type;
                  }
                  description
                    "Identifies an entry in the list of tiebreakers.";
                }
              }
            }
          }
          case objective-function {
            if-feature "path-optimization-objective-function";
            /* Objective functions */
            container objective-function {
              description
                "The objective function container that includes
                 attributes to impose when computing a TE path.";
              leaf objective-function-type {
                type identityref {
                  base objective-function-type;
                }
                default "te-types:of-minimize-cost-path";
                description
                  "Objective function entry.";
              }
            }
          }
        }
      }
    }

    grouping generic-path-affinities {
      description
        "Path affinities grouping.";
      container path-affinities-values {
        description
          "Path affinities represented as values.";
        list path-affinities-value {
          key "usage";
          description
            "List of named affinity constraints.";
          leaf usage {
            type identityref {
              base resource-affinities-type;
            }
```

```
            description
              "Identifies an entry in the list of value affinity
               constraints.";
          }
          leaf value {
            type admin-groups;
            default "";
            description
              "The affinity value.  The default is empty.";
          }
        }
      }
      container path-affinity-names {
        description
          "Path affinities represented as names.";
        list path-affinity-name {
          key "usage";
          description
            "List of named affinity constraints.";
          leaf usage {
            type identityref {
              base resource-affinities-type;
            }
            description
              "Identifies an entry in the list of named affinity
               constraints.";
          }
          list affinity-name {
            key "name";
            leaf name {
              type string;
              description
                "Identifies a named affinity entry.";
            }
            description
              "List of named affinities.";
          }
        }
      }
    }

    grouping generic-path-srlgs {
      description
        "Path SRLG grouping.";
      container path-srlgs-lists {
        description
          "Path SRLG properties container.";
        list path-srlgs-list {
```

```
          key "usage";
          description
            "List of SRLG values to be included or excluded.";
          leaf usage {
            type identityref {
              base route-usage-type;
            }
            description
              "Identifies an entry in a list of SRLGs to either
               include or exclude.";
          }
          leaf-list values {
            type srlg;
            description
              "List of SRLG values.";
          }
        }
      }
      container path-srlgs-names {
        description
          "Container for the list of named SRLGs.";
        list path-srlgs-name {
          key "usage";
          description
            "List of named SRLGs to be included or excluded.";
          leaf usage {
            type identityref {
              base route-usage-type;
            }
            description
              "Identifies an entry in a list of named SRLGs to either
               include or exclude.";
          }
          leaf-list names {
            type string;
            description
              "List of named SRLGs.";
          }
        }
      }
    }

    grouping generic-path-disjointness {
      description
        "Path disjointness grouping.";
      leaf disjointness {
        type te-path-disjointness;
        description
```

```
          "The type of resource disjointness.
           When configured for a primary path, the disjointness level
           applies to all secondary LSPs.  When configured for a
           secondary path, the disjointness level overrides the level
           configured for the primary path.";
      }
    }

    grouping common-path-constraints-attributes {
      description
        "Common path constraints configuration grouping.";
      uses common-constraints;
      uses generic-path-metric-bounds;
      uses generic-path-affinities;
      uses generic-path-srlgs;
    }

    grouping generic-path-constraints {
      description
        "Global named path constraints configuration grouping.";
      container path-constraints {
        description
          "TE named path constraints container.";
        uses common-path-constraints-attributes;
        uses generic-path-disjointness;
      }
    }

    grouping generic-path-properties {
      description
        "TE generic path properties grouping.";
      container path-properties {
        config false;
        description
          "The TE path properties.";
        list path-metric {
          key "metric-type";
          description
            "TE path metric type.";
          leaf metric-type {
            type identityref {
              base path-metric-type;
            }
            description
              "TE path metric type.";
          }
          leaf accumulative-value {
            type uint64;
```

```
            description
              "TE path metric accumulative value.";
          }
        }
        uses generic-path-affinities;
        uses generic-path-srlgs;
        container path-route-objects {
          description
            "Container for the list of route objects either returned by
             the computation engine or actually used by an LSP.";
          list path-route-object {
            key "index";
            ordered-by user;
            description
              "List of route objects either returned by the computation
               engine or actually used by an LSP.";
            leaf index {
              type uint32;
              description
                "Route object entry index.  The index is used to
                 identify an entry in the list.  The order of entries
                 is defined by the user without relying on key
                 values.";
            }
            uses explicit-route-hop;
          }
        }
      }
    }

    grouping encoding-and-switching-type {
      description
        "Common grouping to define the LSP encoding and
        switching types";
      leaf encoding {
        type identityref {
          base te-types:lsp-encoding-types;
        }
        description
          "LSP encoding type.";
        reference
          "RFC3945";
      }
      leaf switching-type {
        type identityref {
          base te-types:switching-capabilities;
        }
        description
```

```
          "LSP switching type.";
        reference
          "RFC3945";
      }
    }
  }
  <CODE ENDS>
```

Acknowledgements

This document was prepared using kramdown.

Authors' Addresses

Italo Busi
Huawei
Email: italo.busi@huawei.com

Aihua Guo
Futurewei Technologies
Email: aihuaguo.ietf@gmail.com

Xufeng Liu
IBM Corporation
Email: xufeng.liu.ietf@gmail.com

Tarek Saad
Juniper Networks
Email: tsaad@juniper.net

Rakesh Gandhi
Cisco Systems, Inc.
Email: rgandhi@cisco.com

Vishnu Pavan Beeram
Juniper Networks
Email: vbeeram@juniper.net

Igor Bryskin
Individual
Email: i_bryskin@yahoo.com

NETMOD Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: 25 August 2022

Q. Wu
B. Claise
Huawei
P. Liu
Z. Du
China Mobile
M. Boucadair
Orange

21 February 2022

Self-Describing Data Object Tags in YANG Data Models
draft-ietf-netmod-node-tags-06

Abstract

This document defines a method to tag data objects that are associated with operation and management data in YANG modules. This method for tagging YANG data objects is meant to be used for classifying data objects from different YANG modules and identifying their characteristics data. Tags may be registered as well as assigned during the module definition, assigned by implementations, or dynamically defined and set by users.

This document also provides guidance to future YANG data model writers; as such, this document updates RFC 8407.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 August 2022.

Table of Contents

1. Introduction

The use of tags for classification and organization purposes is fairly ubiquitous, not only within IETF protocols, but globally in the Internet (e.g., "#hashtags"). For the specific case of YANG data models, a module tag is defined as a string that is associated with a module name at the module level [RFC8819].

Many data models have been specified by various Standards Developing Organizations (SDOs) and the Open Source community, and it is likely that many more will be specified. These models cover many of the networking protocols and techniques. However, the data objects defined by these technology-specific data models might represent a portion of fault, configuration, accounting, performance, and security (FCAPS) management information ([FCAPS]) at different levels and network locations, but also categorised in various different ways. Furthermore, there is no consistent classification criteria or representation for a specific service, feature, or data source.

This document defines self-describing data object tags and associates them with data objects within a YANG module, which:

* Provide dictionary meaning for specific targeted data objects.

* Indicate a relationship between data objects within the same YANG module or from different YANG modules.

* Identify key performance metric data objects and the absolute XPath expression identifying the element path to the node.

The self-describing data object tags can be used by the NETCONF [RFC6241] or RESTCONF [RFC8040] client to classify data objects from different YANG modules and identify characteristic data. In addition, these tags can provide input, instructions, or indications to selection filters and filter queries of configuration or operational state on a server based on these data object tags (e.g., return specific objects containing operational state related to system-management). NETCONF clients can discover data objects with self-describing data object tags supported by a NETCONF server by means of the <get-schema> operation (Section 3.1 of [RFC6022]). The self-describing data object tag capability can also be advertised using the capability notification model [RFC9196] by the NETCONF server or some websites where offline documents are kept. Similar to YANG module tags defined in [RFC8819], these data object tags may be registered or assigned during the module definition, assigned by implementations, or dynamically defined and set by users.

This document defines a YANG module [RFC7950] that augments the module tag model [RFC8819] and provides a list of data object entries to add or remove self-describing tags as well as to view the set of self-describing tags associated with specific data objects within YANG modules.

This document defines three extension statements to indicate self-describing tags that should be added by the module implementation automatically (i.e., outside of configuration).

This document also defines an IANA registry for tag prefixes and a set of globally assigned tags (Section 9).

Section 8 provides guidelines for authors of YANG data models. This document updates [RFC8407].

The YANG data model in this document conforms to the Network Management Datastore Architecture defined in [RFC8342].

2. Terminology

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.

The meanings of the symbols in tree diagrams are defined in [RFC8340].

3. Self-Describing Data Object Tags: Massive Data Object Collection Use Case

Among data object tags, the 'opm' (object, property, metric) tags can be used to tackle massive data object collections, indicate relationships between data objects, and capture YANG data objects associated with YANG-modelled performance metrics data. An example is depicted in Figure 1.

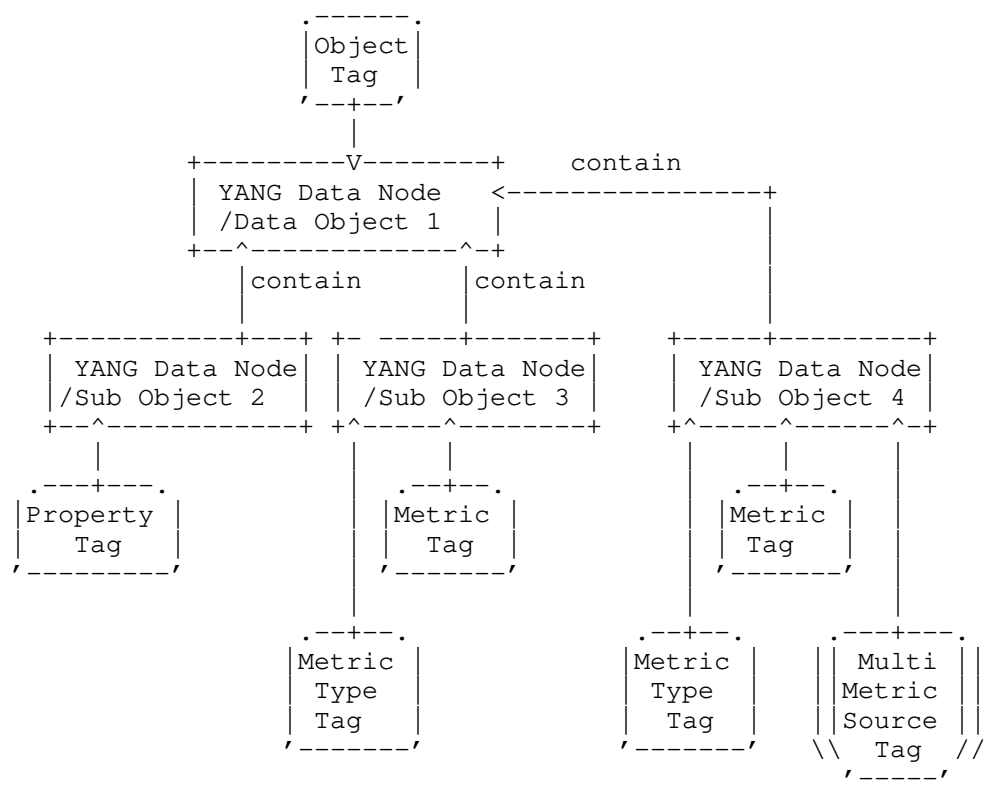


Figure 1: The Relation between Object, Property, and Metric

In Figure 1, data objects can contain other data objects called 'subobjects'. Both object and subobjects can be modeled as YANG data nodes [RFC7950].

Data objects that contain other data objects can be one of type 'container', 'leaf-list', or 'list' and are tagged with the object tag.

A subobject tagged with the property tag is a 'leaf' node.

Subobjects tagged with the metric tag can be one of 'container', 'leaf-list', 'list', or 'leaf' data node.

A data object may contain one single object tag, or one single property tag, or one single metric tag. The data object tagged with the metric tag also can have one or multiple Metric Type tags and/or one single multi-source tag.

The use of 'opm' tags is meant to help filter discrete categories of YANG data objects scattered across the same or different YANG modules that are supported by a device and capture all network performance data or all property data in a single view of the data. In the example shown in Figure 2, the 'tunnel-svc' data object is a container node defined in a 'tunnel-pm' module and can be seen as the root object for property tagged subobjects (e.g., 'tunnel-svc'/'create-time') and metric tagged subobjects (e.g., 'tunnel-svc'/'avg-latency'). The 'name', 'create-time', and 'modified-time' are property tagged subobjects under 'tunnel-svc' container. The 'avg-latency' and 'packet-loss' metrics are tagged subobjects under 'tunnel-svc' container node. Consider the 'tunnel-svc' data object and the 'tunnel-svc/name' data object as an example: the 'tunnel-svc' data object has one single object tag (i.e., 'ietf:object'), while the 'tunnel-svc/name' data object has one single property subobject tag (i.e., 'ietf:property'). In addition, not all metric subobjects need to be tagged (e.g., define specific categories, such as loss-related metric subobjects need to be tagged with a metric-type tag which can further reduce amount data to be fetched).

Data Object	Object Tag	Property Tag	Metric Tag	Multi-Source Tag
tunnel-svc	ietf: object			
tunnel-svc/name		ietf: property		
tunnel-svc/create-time		ietf: property		
tunnel-svc/modified-time		ietf: property		
tunnel-svc/avg-latency			ietf: metric	non-agg
tunnel-svc/packet-loss			ietf: metric	non-agg
tunnel-svc/min-latency			ietf: metric	non-agg
tunnel-svc/max-latency			ietf: metric	non-agg

Figure 2: Example of OPM Tags Used in the YANG Module

If data objects in YANG modules are adequately tagged and learnt by the client from a server, the client can retrieve paths to all targeted data objects and then use an XPath query defined in [RFC8639][RFC8641] to list all tagged data objects which reflect the network characteristics.

4. Data Object Tag Values

All data object tags SHOULD begin with a prefix indicating who owns their definition. An IANA registry (Section 9.1) is used to register data object tag prefixes. Initially, three prefixes are defined.

No further structure is imposed by this document on the value following the registered prefix, and the value can contain any YANG type 'string' characters except carriage returns, newlines, tabs, and spaces.

Except for the conflict-avoiding prefix, this document is purposefully not specifying any structure on (i.e., restricting) the tag values. The intent is to avoid arbitrarily restricting the values that designers, implementers, and users can use. As a result of this choice, designers, implementers, and users are free to add or not add any structure they may require to their own tag values.

4.1. IETF Tags

An IETF tag is a data object tag that has the prefix "ietf:".

All IETF data object tags are registered with IANA in the registry defined in Section 9.2.

4.2. Vendor Tags

A vendor tag is a tag that has the prefix "vendor:".

These tags are defined by the vendor that implements the module, and are not registered with IANA. However, it is RECOMMENDED that the vendor includes extra identification in the tag to avoid collisions, such as using the enterprise or organization name following the "vendor:" prefix (e.g., vendor:example.com:vendor-defined-classifier).

4.3. User Tags

A user tag is any tag that has the prefix "user:". For the avoidance of confusion, the colon (":") when it appears for the first time, is always assumed to be the separator between a prefix and the rest of the tag. And so, when a user tag does not have a prefix, it MUST NOT contain a colon.

These tags are defined by a user/administrator and are not meant to be registered with IANA. Users are not required to use the "user:" prefix; however, doing so is RECOMMENDED as it helps avoid collisions.

4.4. Reserved Tags

Any tag not starting with the prefix "ietf:", "vendor:", or "user:" is reserved for future use (Section 9.1).

These tag values are not invalid, but simply reserved in the context of specifications (e.g., RFCs).

5. Data Object Tag Management

Tags may be associated with a data object within a YANG module in a number of ways. Typically, tags may be defined and associated at the module design time, at implementation time without the need of a live server, or via user administrative control. As the main consumers of data object tags are users, users may also remove any tag from a live server, no matter how the tag became associated with a data object within a YANG module.

5.1. Module Design Tagging

A data object definition MAY indicate a set of data object tags to be added by a module's implementer. These design time tags are indicated using a set of extension statements which include:

opm-tag extension statement: Classifies management and operation data into object, property subobject, and metric subobject categories.

Both objects and subobjects can be modeled as YANG data nodes [RFC7950]. Data objects that contain other data objects can be one of type 'container', 'leaf-list', and 'list' and are tagged with object tag. A subobject tagged with the property tag is a 'leaf' node. Subobjects tagged with the metric tag can be one of 'container', 'leaf-list', 'list', or 'leaf' data node. A data object contains one single object tag, one single property tag, or one single metric tag.

A data object tagged with metric tag also can have one or multiple Metric type tag and/or one single multi-source tag. See the examples depicted in Figure 2 and Figure 4.

metric-type extension statement: Provides metric data objects classifications (e.g., loss, jitter, delay, counter, gauge, summary, unknown) within the YANG module.

multi-source-tag extension statement: Identifies multi-source aggregation type (e.g., aggregated, non-aggregated) related to a metric subobject.

The 'aggregated' multi-source aggregation type allows a large number of measurements on metric subobjects from different sources of the same type (e.g., line card, each subinterface of aggregated Ethernet interface) to be combined into aggregated statistics and report as one metric subobject.

The 'non-aggregated' multi-source aggregation type allows measurement from each source of the same type (e.g., line card, each subinterface of aggregated Ethernet interface) to be reported separately.

Among these extension statements, the 'metric-type' and 'multi-source' tag extension statements are context information that can be used to correlate data objects from the different modules.

If the data node is defined in an IETF Standards Track document, the data object tags MUST be IETF Tags (Section 4.1). Thus, new data objects can drive the addition of new IETF tags to the IANA registry defined in Section 9.2, and the IANA registry can serve as a check against duplication.

5.2. Implementation Tagging

An implementation MAY include additional tags associated with data objects within a YANG module. These tags SHOULD be IETF (Section 4.1) or vendor tags (Section 4.2).

5.3. User Tagging

Data object tags of any kind, with or without a prefix, can be assigned and removed by the user from a server using normal configuration mechanisms. In order to remove a data object tag from the operational datastore, the user adds a matching "masked-tag" entry for a given data object within the 'ietf-data-object-tags' module.

6. Data Object Tags Module Structure

6.1. Data Object Tags Module Tree

The tree associated with the "ietf-data-object-tags" module is as follows:

```
module: ietf-data-object-tags
augment /tags:module-tags/tags:module:
  +--rw data-object-tags
     +--rw data-object* [name]
        +--rw name  nacm:node-instance-identifier
        +--rw tag*         tags:tag
        +--rw masked-tag*  tags:tag
```

Figure 3: YANG Module Tags Tree Diagram

7. YANG Module

This module imports types from [RFC8819],[RFC8341].

```
<CODE BEGINS> file "ietf-data-object-tags@2022-02-04.yang"
module ietf-data-object-tags {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-data-object-tags";
  prefix ntags;

  import ietf-netconf-acm {
    prefix nacm;
    reference
    "RFC 8341: Network Configuration Access Control
               Model";
  }
  import ietf-module-tags {
    prefix tags;
    reference
    "RFC 8819: YANG Module Tags ";
  }
```

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

    Editor: Qin Wu
            <mailto:bill.wu@huawei.com>

    Editor: Benoit Claise
            <mailto:benoit.claise@huawei.com>

    Editor: Peng Liu
            <mailto:liupengyjy@chinamobile.com>

    Editor: Zongpeng Du
            <mailto:duzongpeng@chinamobile.com>

    Editor: Mohamed Boucadair
            <mailto:mohamed.boucadair@orange.com>";
  description
    "This module describes a mechanism associating self-describing
     tags with YANG data object within YANG modules. Tags may be IANA
     assigned or privately defined.

     Copyright (c) 2022 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://datatracker.ietf.org/html/rfcXXXX); see the RFC itself
     for full legal notices.";

  revision 2022-02-04 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Self-Describing Data Object Tags in YANG Data
                 Models";
  }

  extension opm-tag {
```

```
      argument tag;
      description
        "The argument 'tag' is of type 'tag'. This extension statement
         is used by module authors to indicate the opm tags that should
         be added automatically by the system. 'opm-tag' is used to
         classify operation and management data objects into the three
         categories, object, property, and metric. Data Object can
         contain other data objects called subobjects. Both object and
         subobjects can be modeled as data nodes. A data object
         tagged with object tag can be one of container, leaf-list, or
         list. A data object tagged is with the property tag is a leaf
         node. The data object tagged with the metric tag can be one of
         container, leaf-list, list, or leaf. A data objects tagged
         with either property tag or metric tag are subobjects
         belonging to a specific root data object. Each data object may
         contain one single object tag, or one single property tag,
         or one   single metric tag (these tags are mutually
         exclusive). As such,  the origin of the value for the
         pre-defined tags should be set to 'system'.";
    }

    extension metric-type {
      argument tag;
      description
        "The argument 'tag' is of type 'tag'. The metric type can be
         used to provide metric data object classification
         (e.g., loss, jitter, packet loss, counter, gauge,
         summary, unknown) within a YANG module.";
    }

    extension multi-source-tag {
      argument tag;
      description
        "The argument 'tag' is of type 'tag'. The multi-source-tag can
        be used to identify multi-source aggregation type
        (e.g., aggregated,  non-aggregated) related to a metric
        subobject.

        The 'aggregated' multi-source aggregation type allows a large
        number of measurements on metric subobjects from different
        sources of the same type (e.g., line card, each subinterface of
        an aggregated Ethernet interface) to be combined into
        aggregated statistics and reported as one metric subobject
        value.

        The 'non-aggregated'  multi-source aggregation type allows
        measurement from each source of the same type (e.g., line
        card, each subinterface of an aggregated Ethernet interface) to
```

```
        be reported separately.";
      }

    augment "/tags:module-tags/tags:module" {
      description
        "Augment the Module Tags module with data object tag
         attributes.";
      container data-object-tags {
        description
          "Contains the list of data objects and their associated data
           object tags.";
        list data-object {
          key "name";
          description
            "Includes a list of data objects and their associated data
             object tags.";
          leaf name {
            type nacm:node-instance-identifier;
            mandatory true;
            description
              "The YANG data object name.";
          }
          leaf-list tag {
            type tags:tag;
            description
              "Lists the tags associated with the data object within
               the YANG  module.

               See the IANA 'YANG Data Object Tag Prefixes' registry
               for reserved prefixes and the IANA 'IETF YANG Data
               Object Tags' registry for IETF tags.

               The 'operational' state view of this list is
               constructed using the following steps:

               1) System tags (i.e., tags of 'system' origin) are
                  added.
               2) User configured tags (i.e., tags of 'intended'
                  origin) are added.
               3) Any tag that is equal to a masked-tag is removed.";
            reference
              "RFC XXXX: Self-Describing Data Object Tags in YANG Data
                         Models, Section 9";
          }
          leaf-list masked-tag {
            type tags:tag;
            description
              "The list of tags that should not be associated with the
```

```
                data object within the YANG module. The user can remove
                (mask) tags from the operational state datastore by
                adding them to this list. It is not an error to add tags
                to this list that are not associated with the data
                object within YANG  module, but they have no operational
                effect.";
            }
          }
        }
      }
    }
    <CODE ENDS>
```

8. Guidelines to Model Writers

This section updates [RFC8407].

8.1. Define Standard Tags

A module MAY indicate, using data object tag extension statements, a set of data object tags that are to be automatically associated with data object within the module (i.e., not added through configuration).

```
module example-module-A {
  //...
  import ietf-data-node-tags { prefix ntags; }

  container top {
    ntags:opm-tag "ietf:object";
    list X {
      leaf foo {
         ntags:opm-tag "ietf:property";
      }
      leaf bar {
        ntags:opm-tag "ietf:metric";
      }
    }
  }
  // ...
}
```

Figure 4: An Example of Data Object Tag

The module writer can use existing standard data object tags, or use new data object tags defined in the data object definition, as appropriate. For IETF standardized modules, new data object tags MUST be assigned in the IANA registry defined in Section Section 9.2.

9. IANA Considerations

9.1. YANG Data Object Tag Prefixes Registry

This document requests IANA to create "YANG Data Object Tag Prefixes" subregistry in "YANG Data Object Tag" registry.

This registry allocates tag prefixes. All YANG Data Object Tags should begin with one of the prefixes in this registry.

Prefix entries in this registry should be short strings consisting of lowercase ASCII alpha-numeric characters and a final ":" character.

The allocation policy for this registry is Specification Required [RFC8126]. The Reference and Assignee values should be sufficient to identify and contact the organization that has been allocated the prefix. There is no specific guidance for the Designated Expert and there is a presumption that a code point should be granted unless there is a compelling reason to the contrary.

The initial values for this registry are as follows:

Prefix	Description	Reference	Assignee
ietf:	IETF Tags allocated in the IANA IETF YANG Data Object Tags registry	[This document]	IETF
vendor:	Non-registered tags allocated by the module's implementer.	[This document]	IETF
user:	Non-registered tags allocated by and for the user.	[This document]	IETF

Figure 5: Table 1

Other standards organizations (SDOs) wishing to allocate their own set of tags should request the allocation of a prefix from this registry.

9.2. IETF YANG Data Object Tags Registry

This document requests IANA to create "IETF OPM Tags","IETF Metric Type Tags","IETF Multiple Source Tags" three subregistries in "YANG Data Object Tag" registry. These 3 subregistries appear below "YANG Data Object Tag Prefixes" registry.

Three subregistries allocate tags that have the registered prefix "ietf:". New values should be well considered and not achievable through a combination of already existing IETF tags.

The allocation policy for these three subregistries is IETF Review [RFC8126]. The Designated Expert is expected to verify that IANA assigned tags conform to Net-Unicode as defined in [RFC5198], and shall not need normalization.

The initial values for these three subregistries are as follows:

OPM Tag	Description	Reference
ietf:object	Represents Root object containing other data objects (e.g., interfaces)	[This document]
ietf:property	Represents a property data object(e.g., ifindex) associated with a specific root object (e.g., interfaces)	[This document]
ietf:metric	Represent metric data object(e.g., ifstatistics) associated with specific root object(e.g., interfaces)	[This document]

Metric Type Tag	Description	Reference
ietf:delay	Represents the delay metric group to which the metric data objects belong to.	[This document]
ietf:jitter	Represents the jitter metric group to which the metric data objects belong to.	[This document]
ietf:loss	Represents the loss metric group to which the metric data objects belong to.	[This document]
ietf:counter	Represents any metric value associated with a metric data object that monotonically	[This

	increases over time, starting from zero.	document]
ietf:gauge	Represents current measurements associated with a metric data object that may increase, decrease or stay constant.	[This document]
ietf:summary	Represents the metric value associated with a metric data object that measures distributions of discrete events without knowing predefined range.	[This document
ietf:unknown	Represents the metric value associated with metric data object that can not determine the type of metric.	[This document

Multiple Source Tag	Description	Reference
ietf:agg	Relates to multiple sources aggregation type (i.e., aggregated statistics)	[This document]
ietf:non-agg	Relates to multiple sources aggregation type (i.e., non-aggregated statistics)	[This document]

Figure 6: Table 2

9.3. Updates to the IETF XML Registry

This document registers the following namespace URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-data-object-tags
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

9.4. Updates to the YANG Module Names Registry

This document registers the following YANG module in the YANG Module Names registry [RFC6020] within the "YANG Parameters" registry:

```
name: ietf-data-object-tags
namespace: urn:ietf:params:xml:ns:yang:ietf-data-object-tags
prefix: ntags
reference: RFC XXXX
maintained by IANA: N
```

10. Security Considerations

The YANG module specified in this document defines schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content, e.g., the presence of tags may reveal information about the way in which data objects are used and therefore providing access to private information or revealing an attack vector should be restricted. Note that appropriate privilege and security levels need to be applied to the addition and removal of user tags to ensure that a user receives the correct data.

This document adds the ability to associate data object tag meta-data with data object within the YANG modules. This document does not define any actions based on these associations, and none are yet defined, and therefore it does not by itself introduce any new security considerations.

Users of the data object tag meta-data may define various actions to be taken based on the data object tag meta-data. These actions and their definitions are outside the scope of this document. Users will need to consider the security implications of any actions they choose to define, including the potential for a tag to get ’masked’ by another user.

11. Acknowledgements

The authors would like to thank Ran Tao for his major contributions to the initial modeling and use cases.

The authors would also like to acknowledge the comments and suggestions received from Juergen Schoenwaelder, Andy Bierman, Lou Berger,Jaehoon Paul Jeong, Wei Wang, Yuan Zhang, Ander Liu, YingZhen Qu, Boyuan Yan, Adrian Farrel, and Mahesh Jethanandani.

12. Contributors

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

13. References

13.1. 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/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8819] Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", RFC 8819, DOI 10.17487/RFC8819, January 2021, <https://www.rfc-editor.org/info/rfc8819>.

13.2. Informative References

[FCAPS] International Telecommunication Union, "X.700 : Management framework for Open Systems Interconnection (OSI) for CCITT applications", , September 1992, <http://www.itu.int/rec/T-REC-X.700-199209-I/en>.

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <https://www.rfc-editor.org/info/rfc5198>.

[RFC6022] Scott, M. and M. Bjorklund, "YANG Module for NETCONF Monitoring", RFC 6022, DOI 10.17487/RFC6022, October 2010, <https://www.rfc-editor.org/info/rfc6022>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[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/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[RFC9195] Lengyel, B. and B. Claise, "A File Format for YANG Instance Data", RFC 9195, DOI 10.17487/RFC9195, February 2022, <https://www.rfc-editor.org/info/rfc9195>.

[RFC9196] Lengyel, B., Clemm, A., and B. Claise, "YANG Modules Describing Capabilities for Systems and Datastore Update Notifications", RFC 9196, DOI 10.17487/RFC9196, February 2022, <https://www.rfc-editor.org/info/rfc9196>.

Appendix A. NETCONF Example

The following is a NETCONF example result from a query of the data object tags list. For the sake of brevity only a few module and associated data object results are provided. The example uses the folding defined in [RFC8792].

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<ns0:data xmlns:ns0="urn:ietf:params:xml:ns:netconf:base:1.0">
 <t:module-tags xmlns:t="urn:ietf:params:xml:ns:yang:ietf-module-tags">
    <t:module>
       <t:name>ietf-interfaces</t:name>
       <s:data-object-tags xmlns:s="urn:ietf:params:xml:ns:yang:ietf-\
data-object-tags">
        <s:data-object>
         <s:name>/if:interfaces/if:interface</s:name>
         <s:tag>ietf:object</s:tag>
        </s:data-object>
       <s:data-object>
```

```
         <s:name>/if:interfaces/if:interface/if:last-change</\
 s:name>
         <s:tag>ietf:property</s:tag>
        </s:data-object>
         <s:data-object>
          <s:name>
           /if:interfaces/if:interface/if:statistics/if:in-errors
          </s:name>
          <s:tag>ietf:metric</s:tag>
          <s:tag>ietf:loss</s:tag>
          <s:tag>ietf:non-agg</s:tag>
         </s:data-object>
       </s:data-object-tags>
     </t:module>
     <t:module>
        <t:name>ietf-ip</t:name>
        <s:data-object-tags xmlns:s="urn:ietf:params:xml:ns:yang:ietf\
 -data-object-tags">
        <s:data-object>
          <s:name>/if:interfaces/if:interface/ip:ipv4</s:name>
          <s:tag>ietf:object</s:tag>
        </s:data-object>
        <s:data-object>
          <s:name>/if:interfaces/if:interface/ip:ipv4/ip:enable\
 </s:name>
          <s:tag>ietf:property</s:tag>
        </s:data-object>
         <s:data-object>
         <s:name>/if:interfaces/if:interface/ip:ipv4/ip:mtu</s:name>
         <s:tag>ietf:metric</s:tag>
         <s:tag>ietf:non-agg</s:tag>
        </s:data-object>
       </s:data-object-tags>
     </t:module>
   </t:module-tags>
 </ns0:data>
```

Figure 7: Example NETCONF Query Output

Appendix B. Non-NMDA State Module

As per [RFC8407], the following is a non-NMDA module to support viewing the operational state for non-NMDA compliant servers.

```
<CODE BEGINS> file "ietf-data-object-tags-state@2022-02-03.yang"
module ietf-data-object-tags-state {
  yang-version 1.1;
  namespace
        "urn:ietf:params:xml:ns:yang:ietf-data-object-tags-state";
  prefix ntags-s;

  import ietf-netconf-acm {
    prefix nacm;
    reference
    "RFC 8341: Network Configuration Access Control
               Model";
  }
  import ietf-module-tags {
    prefix tags;
    reference
    "RFC 8819: YANG Module Tags ";
  }
  organization
    "IETF NetMod Working Group (NetMod)";

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

    Editor: Qin Wu
            <mailto:bill.wu@huawei.com>

    Editor: Benoit Claise
            <mailto:benoit.claise@huawei.com>

    Editor: Peng Liu
            <mailto:liupengyjy@chinamobile.com>

    Editor: Zongpeng Du
            <mailto:duzongpeng@chinamobile.com>

    Editor: Mohamed Boucadair
            <mailto:mohamed.boucadair@orange.com>";

  description
    "This module describes a mechanism associating self-describing
     tags with YANG data object within YANG modules. Tags may be
     IANA assigned or privately defined.

     Copyright (c) 2022 IETF Trust and the persons identified as
     authors of the code. All rights reserved.
```

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

        This version of this YANG module is part of RFC XXXX
        (https://datatracker.ietf.org/html/rfcXXXX); see the RFC
        itself for full legal notices.";

    revision 2022-02-04 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: Self-Describing Data Object Tags in YANG Data
                   Models";
    }

    extension opm-tag {
      argument tag;
      description
        "The argument 'tag' is of type 'tag'. This extension
         statement is used by module authors to indicate the opm tags
         that should be added automatically by the system. 'opm-tag'
         is used to classify operation and management data objects
         into the three categories, object, property, and metric.
         Data Object can contain other data objects called subobjects.
         Both object and subobjects can be modeled as data nodes. The
         Data Object tagged with object tag can be one of container,
         leaf-list and list. The Data Object tagged with the Property
         tag is a leaf node. The Data Object tagged with the Metric
         tag can be one of type container, leaf-list, list, leaf. The
         Data objects tagged with either property tag or metric tag
         are subobjects belonging to a specific root data object. Each
         Data Object may contain one single object tag, or one single
         property tag, or one single metric tag (these tags are
         mutually exclusive). As such, the origin of the value for the
         pre-defined tags should be set to 'system'.";
    }
    extension metric-type {
      argument tag;
      description
        "The argument 'tag' is of type 'tag'.The metric-type can be
         used to provide metric subobject classification
         (e.g., loss, jitter, packet loss, guage, counter, histogram,
         unknow, etc.) within the YANG module.";
    }
```

```
  extension multi-source-tag {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'. The multi-source tag can
       be used to identify multi-source aggregation type (e.g.,
       aggregated, non-aggregated) related to a metric subobject.

      The 'aggregated' multi-source aggregation type allows a large
      number of measurements on metric subobjects from different
      sources of the same type (e.g., line card, each subinterface
      of aggregated Ethernet interface) to be combined into
      aggregated statistics and reported as one metric subobject
      value.

      The 'non-aggregated'multi-source aggregation type
      allows measurement from each source of the same type
      (e.g., line card, each subinterface of aggregated Ethernet
      interface) to be reported separately.";
   }

  augment "/tags:module-tags/tags:module" {
    description
      "Augments the Module Tags module with data object tag
       attributes.";
    container data-object-tags {
      config false;
      status deprecated;
      description
        "Contains the list of data objects and their
         associated self describing tags.";
      list data-object {
        key "name";
        status deprecated;
        description
          "Lists the data objects and their associated self
           describing tags.";
        leaf name {
          type nacm:node-instance-identifier;
          mandatory true;
          status deprecated;
          description
            "The YANG data object name.";
        }
        leaf-list tag {
          type tags:tag;
          status deprecated;
          description
            "Tags associated with the data object within the
```

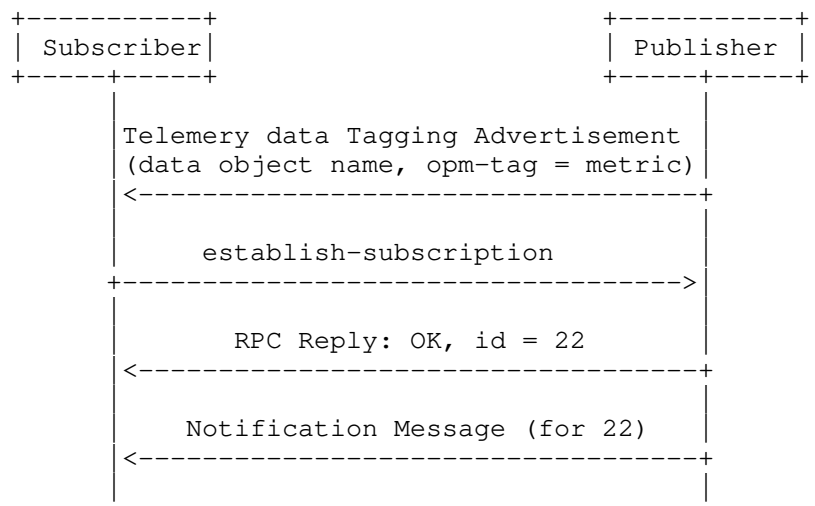
```
                YANG module. See the IANA 'YANG Data Object Tag
                Prefixes' registry for reserved prefixes and the
                IANA 'IETF YANG Data Object Tags'registry for
                IETF tags.

                The 'operational' state view of this list is
                constructed using the following steps:

                1) System tags (i.e., tags of 'system' origin) are
                   added.
                2) User configured tags (i.e., tags of 'intended'
                   origin) are added.
                3) Any tag that is equal to a masked-tag is removed.";
              reference
                "RFC XXXX: Self-Describing Data Object Tags in YANG Data
                           Models, Section 9";
            }
            leaf-list masked-tag {
              type tags:tag;
              status deprecated;
              description
                "The list of tags that should not be associated with the
                 data object within the YANG module. The user can remove
                 (mask) tags from the operational state datastore by
                 adding them to this list. It is not an error to add
                 tags to this list that are not associated with the
                 data object within YANG module, but they have no
                 operational effect.";
            }
          }
        }
      }
    }
   }
   <CODE ENDS>
```

Appendix C. Targeted data object collection example

The following provides targeted data object collection example which helps reduce amount of data to be fetched. The subscription "id" values of 22 used below is just an example. In production, the actual values of "id" might not be small integers.



The publisher advertises telemetry data object capability to the subscriber to instruct the receiver to subscribe tagged data object (e.g., performance metric data object) using standard subscribed notification mechanism [RFC8639]. The corresponding telemetry data object capability model is created based on ietf-data-object-tags module defined in this document.

Figure 8 illustrates the advertisment of the list of available target objects using the YANG instance file format [RFC9195]:

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=\
"urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilities</name>
  <content-schema>
    <module>ietf-system-capabilities@2020-03-23</module>
    <module>ietf-notification-capabilities@2020-03-23</module>
    <module>ietf-data-export-capabilities@2020-03-23</module>
  </content-schema>
  <!-- revision date, contact, etc. -->
  <description>Defines the notification capabilities of an
    acme-router.The router only has running, and operational
    datastores. Every change can be reported on-change from
    running, but only config=true nodes and some config=false data
    from operational. Statistics are not reported based on timer
    based trigger and counter threshold based trigger.
  </description>
  <content-data>
    <system-capabilities \
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities" \
xmlns:inc=\
"urn:ietf:params:xml:ns:yang:ietf-notification-capabilities" \
xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      <datastore-capabilities>
        <datastore>ds:operational</datastore>
        <per-node-capabilities>
          <node-selector>\
/if:interfaces/if:interface/if:statistics/if:in-errors\
</node-selector>
          <sec:self-describing-capabilities>
            <sec:opm-tag>metric</sec:opm-tag>
            <sec:metric-type>loss</sec:metric-type>
          </sec:self-describing-capabilities>
        </per-node-capabilities>
      </datastore-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

Figure 8: List of Available Target Objects

With telemetry data tagging information carried in the telemetry data tagging Advertisement, the subscriber identifies targeted data object and associated data path to the datastore node and sends a standard establish-subscription RPC [RFC8639] to subscribe tagged data objects

that are interests to the client application from the publisher. Alternatively, the subscriber can query data object tag list from somewhere (e.g., the network device, or offline document) using ietf-data-object-tags module defined in this document and fetch tagged data objects and associated data path to the datastore node and sends a standard establish-subscription RPC [RFC8639] to subscribe tagged data objects that are interests to the client application from the publisher.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

 <netconf:rpc message-id="101"
     xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
   <establish-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifica\
tions"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <yp:datastore
         xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:operational
     </yp:datastore>
     <yp:datastore-xpath-filter
        xmlns:ex="https://example.com/sample-data/1.0">
       /if:interfaces/if:interface/if:statistics/if:in-errors
     </yp:datastore-xpath-filter>
     <yp:periodic>
       <yp:period>500</yp:period>
     </yp:periodic>
   </establish-subscription>
 </netconf:rpc>
```

The publisher returns specific object types of operational state (e.g., in-errors statistics data) subscribed by the client.

Appendix D. Changes between Revisions

Editorial Note (To be removed by RFC Editor)

v05 - v06

* Additional Editorial changes;

* Use the folding defined in [RFC8792].

v04 - v05

* Add user tag formating clarification;

* Provide guidance to the Designated Expert for evaluation of YANG Data Object Tag registry and YANG Data Object Tag prefix registry.

* Update the figure 1 and figure 2 with additional tags.

* Security section enhancement for user tag managment.

* Change data object name into name in the module.

* Other Editorial changes to address Adrian's comments and comments during YANG docotor review.

* Open issue: Are there any risks associated with an attacker adding or removing tags so that a requester gets the wrong data?

v03 - v04

* Remove histogram metric type tag from metric type tags.

* Clarify the object tag and property tag,metric tag are mutual exlusive.

* Clarify to have two optional node tags (i.e.,object tag and property tag) to indicate relationship between data objects.

* Update targeted data object collection example.

v02 - v03

* Additional Editorial changes.

* Security section enhancement.

* Nits fixed.

v01 - v02

* Clarify the relation between data object, object tag, property tag and metric tag in figure 1 and figure 2 and related description;

* Change Metric Group into Metric Type in the YANG model;

* Add 5 metric types in section 7.2;

v00 - v01

* Merge self-describing data object tag use case section into introduction section as a subsection;

* Add one glossary section;

* Clarify the relation between data object, object tag, property tag and metric tag in Self-Describing Data Object Tags Use Case section;

* Add update to RFC8407 in the front page.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: bill.wu@huawei.com

Benoit Claise
Huawei
De Kleetlaan 6a b1
1831 Diegem
Belgium
Email: benoit.claise@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing
Email: liupengyjy@chinamobile.com

Zongpeng Du
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing
Email: duzongpeng@chinamobile.com

Mohamed Boucadair
Orange
35000 Rennes
France
Email: mohamed.boucadair@orange.com

Network Working Group
Internet-Draft
Updates: 6020,7950,8407,8525 (if approved)
Intended status: Standards Track
Expires: May 12, 2022

R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman, Ed.

B. Lengyel, Ed.
Ericsson
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia

November 8, 2021

Updated YANG Module Revision Handling
draft-ietf-netmod-yang-module-versioning-05

Abstract

This document specifies a new YANG module update procedure that can document when non-backwards-compatible changes have occurred during the evolution of a YANG module. It extends the YANG import statement with an earliest revision filter to better represent inter-module dependencies. It provides guidelines for managing the lifecycle of YANG modules and individual schema nodes. It provides a mechanism, via the revision-label YANG extension, to specify a revision identifier for YANG modules and submodules. This document updates RFC 7950, RFC 6020, RFC 8407 and RFC 8525.

Table of Contents

1. Introduction

This document defines the foundational pieces of a solution to the YANG module lifecycle problems described in [I-D.ietf-netmod-yang-versioning-reqs]. Complementary documents provide other parts of the solution, with the overall relationship of the solution drafts described in [I-D.ietf-netmod-yang-solutions].

Specifically, this document recognises a need (within standards organizations, vendors, and the industry) to sometimes allow YANG modules to evolve with non-backwards-compatible changes, which could cause breakage to clients and importing YANG modules. Accepting that non-backwards-compatible changes do sometimes occur, it is important to have mechanisms to report where these changes occur, and to manage their effect on clients and the broader YANG ecosystem.

The document comprises five parts:

Refinements to the YANG 1.1 module revision update procedure, supported by new extension statements to indicate when a revision contains non-backwards-compatible changes, and an optional revision label.

A YANG extension statement allowing YANG module imports to specify an earliest module revision that may satisfy the import dependency.

Updates and augmentations to ietf-yang-library to include the revision label in the module and submodule descriptions, to report how "deprecated" and "obsolete" nodes are handled by a server, and to clarify how module imports are resolved when multiple revisions could otherwise be chosen.

Considerations of how versioning applies to YANG instance data.

Guidelines for how the YANG module update rules defined in this document should be used, along with examples.

Note to RFC Editor (To be removed by RFC Editor)

Open issues are tracked at <https://github.com/netmod-wg/yang-ver-dt/issues>.

1.1. Updates to YANG RFCs

This document updates [RFC7950] section 11 and [RFC6020] section 10. Section 3 describes modifications to YANG revision handling and update rules, and Section 4 describes a YANG extension statement to do import by derived revision.

This document updates [RFC7950] section 5.2 and [RFC6020] section 5.2. Section 3.4.1 describes the use of a revision label in the name of a file containing a YANG module or submodule.

This document updates [RFC7950] section 5.6.5 and [RFC8525]. Section 5.1 defines how a client of a YANG library datastore schema resolves ambiguous imports for modules which are not "import-only".

This document updates [RFC8407] section 4.7. Section 7 provides guidelines on managing the lifecycle of YANG modules that may contain non-backwards-compatible changes and a branched revision history.

This document updates [RFC8525] with augmentations to include revision labels in the YANG library data and two boolean leafs to indicate whether status deprecated and status obsolete schema nodes are implemented by the server.

2. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

In addition, this document uses the following terminology:

- YANG module revision: An instance of a YANG module, uniquely identified with a revision date, with no implied ordering or backwards compatibility between different revisions of the same module.

- Backwards-compatible (BC) change: A backwards-compatible change between two YANG module revisions, as defined in Section 3.1.1

- Non-backwards-compatible (NBC) change: A non-backwards-compatible change between two YANG module revisions, as defined in Section 3.1.2

3. Refinements to YANG revision handling

[RFC7950] and [RFC6020] assume, but do not explicitly state, that the revision history for a YANG module or submodule is strictly linear, i.e., it is prohibited to have two independent revisions of a YANG module or submodule that are both directly derived from the same parent revision.

This document clarifies [RFC7950] and [RFC6020] to explicitly allow non-linear development of YANG module and submodule revisions, so that they MAY have multiple revisions that directly derive from the same parent revision. As per [RFC7950] and [RFC6020], YANG module and submodule revisions continue to be uniquely identified by their revision date, and hence all revisions of a given module or submodule MUST have unique revision dates.

A corollary to the above is that the relationship between two module or submodule revisions cannot be determined by comparing the module or submodule revision date alone, and the revision history, or revision label, must also be taken into consideration.

A module's name and revision date identifies a specific immutable definition of that module within its revision history. Hence, if a module includes submodules then to ensure that the module's content is uniquely defined, the module's "include" statements SHOULD use "revision-date" substatements to specify the exact revision date of each included submodule. When a module does not include its submodules by revision-date, the revision of submodules used cannot be derived from the including module. Mechanisms such as YANG packages [I-D.ietf-netmod-yang-packages], and YANG library [RFC8525], MAY be used to specify the exact submodule revisions used when the submodule revision date is not constrained by the "include" statement.

[RFC7950] section 11 and [RFC6020] section 10 require that all updates to a YANG module are BC to the previous revision of the module. This document introduces a method to indicate that an NBC change has occurred between module revisions: this is done by using a new "non-backwards-compatible" YANG extension statement in the module revision history.

Two revisions of a module or submodule MAY have identical content except for the revision history. This could occur, for example, if a module or submodule has a branched history and identical changes are applied in multiple branches.

3.1. Updating a YANG module with a new revision

This section updates [RFC7950] section 11 and [RFC6020] section 10 to refine the rules for permissible changes when a new YANG module revision is created.

Where pragmatic, updates to YANG modules SHOULD be backwards-compatible, following the definition in Section 3.1.1.

A new module revision MAY contain NBC changes, e.g., the semantics of an existing data-node definition MAY be changed in an NBC manner without requiring a new data-node definition with a new identifier. A YANG extension, defined in Section 3.2, is used to signal the potential for incompatibility to existing module users and readers.

As per [RFC7950] and [RFC6020], all published revisions of a module are given a new unique revision date. This applies even for module revisions containing (in the module or included submodules) only changes to any whitespace, formatting, comments or line endings (e.g., DOS vs UNIX).

3.1.1. Backwards-compatible rules

A change between two module revisions is defined as being "backwards-compatible" if the change conforms to the module update rules specified in [RFC7950] section 11 and [RFC6020] section 10, updated by the following rules:

- A "status" "deprecated" statement MAY be added, or changed from "current" to "deprecated", but adding or changing "status" to "obsolete" is not a backwards-compatible change.

- YANG schema nodes with a "status" "obsolete" substatement MAY be removed from published modules, and are classified as backwards-compatible changes. In some circumstances it may be helpful to retain the obsolete definitions since their identifiers may still be referenced by other modules and to ensure that their identifiers are not reused with a different meaning.

- In statements that have any data definition statements as substatements, those data definition substatements MAY be reordered, as long as they do not change the ordering of any "input" or "output" data definition substatements of "rpc" or

"action" statements. If new data definition statements are added, they can be added anywhere in the sequence of existing substatements.

- A statement that is defined using the YANG "extension" statement MAY be added, removed, or changed, if it does not change the semantics of the module. Extension statement definitions SHOULD specify whether adding, removing, or changing statements defined by that extension are backwards-compatible or non-backwards-compatible.

- Any changes (including whitespace or formatting changes) that do not change the semantic meaning of the module are backwards compatible.

3.1.2. Non-backwards-compatible changes

Any changes to YANG modules that are not defined by Section 3.1.1 as being backwards-compatible are classified as "non-backwards-compatible" changes.

3.2. non-backwards-compatible revision extension statement

The "rev:non-backwards-compatible" extension statement is used to indicate YANG module revisions that contain NBC changes.

If a revision of a YANG module contains changes, relative to the preceding revision in the revision history, that do not conform to the module update rules defined in Section 3.1.1, then a "rev:non-backwards-compatible" extension statement MUST be added as a substatement to the "revision" statement.

3.3. Removing revisions from the revision history

Authors may wish to remove revision statements from a module or submodule. Removal of revision information may be desirable for a number of reasons including reducing the size of a large revision history, or removing a revision that should no longer be used or imported. Removing revision statements is allowed, but can cause issues and SHOULD NOT be done without careful analysis of the potential impact to users of the module or submodule. Doing so can lead to import breakages when import by revision-or-derived is used. Moreover, truncating history may cause loss of visibility of when non-backwards-compatible changes were introduced.

An author MAY remove a contiguous sequence of entries from the end (i.e., oldest entries) of the revision history. This is acceptable

even if the first remaining (oldest) revision entry in the revision history contains a rev:non-backwards-compatible substatement.

An author MAY remove a contiguous sequence of entries in the revision history as long as the presence or absence of any existing rev:non-backwards-compatible substatements on all remaining entries still accurately reflect the compatibility relationship to their preceding entries remaining in the revision history.

The author MUST NOT remove the first (i.e., newest) revision entry in the revision history.

Example revision history:

```
revision 2020-11-11 {
  rev:revision-label 4.0.0;
  rev:non-backwards-compatible;
}

revision 2020-08-09 {
  rev:revision-label 3.0.0;
  rev:non-backwards-compatible;
}

revision 2020-06-07 {
  rev:revision-label 2.1.0;
}

revision 2020-02-10 {
  rev:revision-label 2.0.0;
  rev:non-backwards-compatible;
}

revision 2019-10-21 {
  rev:revision-label 1.1.3;
}

revision 2019-03-04 {
  rev:revision-label 1.1.2;
}

revision 2019-01-02 {
  rev:revision-label 1.1.1;
}
```

In the revision history example above, removing the revision history entry for 2020-02-10 would also remove the rev:non-backwards-

compatible annotation and hence the resulting revision history would incorrectly indicate that revision 2020-06-07 is backwards-compatible with revisions 2019-01-02 through 2019-10-21 when it is not, and so this change cannot be made. Conversely, removing one or more revisions out of 2019-03-04, 2019-10-21 and 2020-08-09 from the revision history would still retain a consistent revision history, and is acceptable, subject to an awareness of the concerns raised in the first paragraph of this section.

3.4. Revision label

Each revision entry in a module or submodule MAY have a revision label associated with it, providing an alternative alias to identify a particular revision of a module or submodule. The revision label could be used to provide an additional versioning identifier associated with the revision.

A revision label scheme is a set of rules describing how a particular type of revision-label operates for versioning YANG modules and submodules. For example, YANG Semver [I-D.ietf-netmod-yang-semver] defines a revision label scheme based on Semver 2.0.0 [semver]. Other documents may define other YANG revision label schemes.

Submodules MAY use a revision label scheme. When they use a revision label scheme, submodules MAY use a revision label scheme that is different from the one used in the including module.

The revision label space of submodules is separate from the revision label space of the including module. A change in one submodule MUST result in a new revision label of that submodule and the including module, but the actual values of the revision labels in the module and submodule could be completely different. A change in one submodule does not result in a new revision label in another submodule. A change in a module revision label does not necessarily mean a change to the revision label in all included submodules.

If a revision has an associated revision label, then it may be used instead of the revision date in a "rev:revision-or-derived" extension statement argument.

A specific revision-label identifies a specific revision of the module. If two YANG modules contain the same module name and the same revision-label (and hence also the same revision-date) in their latest revision statement, then the file contents of the two modules, including the revision history, MUST be identical.

3.4.1. File names

This section updates [RFC7950] section 5.2 and [RFC6020] section 5.2.

If a revision has an associated revision label, then the revision-label MAY be used instead of the revision date in the filename of a YANG file, where it takes the form:

```
module-or-submodule-name [['@' revision-date]|['#' revision-label]]
    ( '.yang' / '.yin' )

  E.g., acme-router-module@2018-01-25.yang
  E.g., acme-router-module#2.0.3.yang
```

YANG module (or submodule) files MAY be identified using either revision-date or revision-label. Typically, only one file name SHOULD exist for the same module (or submodule) revision. Two file names, one with the revision date and another with the revision label, MAY exist for the same module (or submodule) revision, e.g., when migrating from one scheme to the other.

3.4.2. Revision label scheme extension statement

The optional "rev:revision-label-scheme" extension statement is used to indicate which revision-label scheme a module or submodule uses. There MUST NOT be more than one revision label scheme in a module or submodule. The mandatory argument to this extension statement:

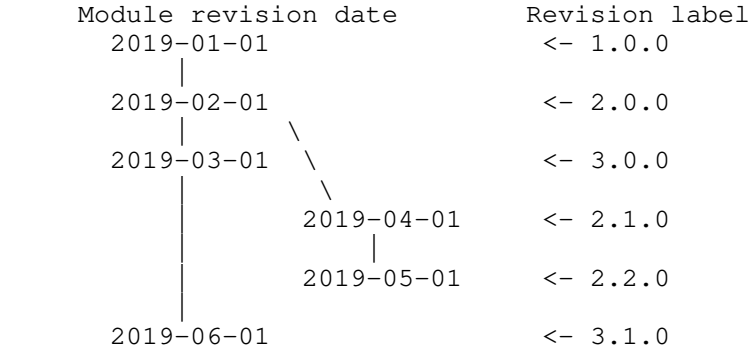
- specifies the revision-label scheme used by the module or submodule
- is defined in the document which specifies the revision-label scheme
- MUST be an identity derived from "revision-label-scheme-base".

The revision-label scheme used by a module or submodule SHOULD NOT change during the lifetime of the module or submodule. If the revision-label scheme used by a module or submodule is changed to a new scheme, then all revision-label statements that do not conform to the new scheme MUST be replaced or removed.

3.5. Examples for updating the YANG module revision history

The following diagram, explanation, and module history illustrates how the branched revision history, "non-backwards-compatible" extension statement, and "revision-label" extension statement could be used:

Example YANG module with branched revision history.



The tree diagram above illustrates how an example module's revision history might evolve, over time. For example, the tree might represent the following changes, listed in chronological order from the oldest revision to the newest revision:

Example module, revision 2019-06-01:

```
    module example-module {

      namespace "urn:example:module";
      prefix "prefix-name";
      rev:revision-label-scheme "yangver:yang-semver";

      import ietf-yang-revisions { prefix "rev"; }
      import ietf-yang-semver { prefix "yangver"; }

      description
        "to be completed";

      revision 2019-06-01 {
        rev:revision-label 3.1.0;
        description "Add new functionality.";
      }

      revision 2019-03-01 {
        rev:revision-label 3.0.0;
        rev:non-backwards-compatible;
        description
          "Add new functionality. Remove some deprecated nodes.";
      }

      revision 2019-02-01 {
        rev:revision-label 2.0.0;
        rev:non-backwards-compatible;
        description "Apply bugfix to pattern statement";
      }

      revision 2019-01-01 {
        rev:revision-label 1.0.0;
        description "Initial revision";
      }

      //YANG module definition starts here
    }
```

Example module, revision 2019-05-01:

```
    module example-module {

      namespace "urn:example:module";
      prefix "prefix-name";
      rev:revision-label-scheme "yangver:yang-semver";

      import ietf-yang-revisions { prefix "rev"; }
      import ietf-yang-semver { prefix "yangver"; }

      description
        "to be completed";

      revision 2019-05-01 {
        rev:revision-label 2.2.0;
        description "Backwards-compatible bugfix to enhancement.";
      }

      revision 2019-04-01 {
        rev:revision-label 2.1.0;
        description "Apply enhancement to older release train.";
      }

      revision 2019-02-01 {
        rev:revision-label 2.0.0;
        rev:non-backwards-compatible;
        description "Apply bugfix to pattern statement";
      }

      revision 2019-01-01 {
        rev:revision-label 1.0.0;
        description "Initial revision";
      }

      //YANG module definition starts here
    }
```

4. Import by derived revision

[RFC7950] and [RFC6020] allow YANG module "import" statements to optionally require the imported module to have a particular revision date. In practice, importing a module with an exact revision date is often too restrictive because it requires the importing module to be updated whenever any change to the imported module occurs. The alternative choice of using an import statement without any revision date statement is also not ideal because the importing module may not work with all possible revisions of the imported module.

Instead, it is desirable for an importing module to specify a "minimum required revision" of a module that it is compatible with, based on the assumption that later revisions derived from that "minimum required revision" are also likely to be compatible. Many possible changes to a YANG module do not break importing modules, even if the changes themselves are not strictly backwards-compatible. E.g., fixing an incorrect pattern statement or description for a leaf would not break an import, changing the name of a leaf could break an import but frequently would not, but removing a container would break imports if that container is augmented by another module.

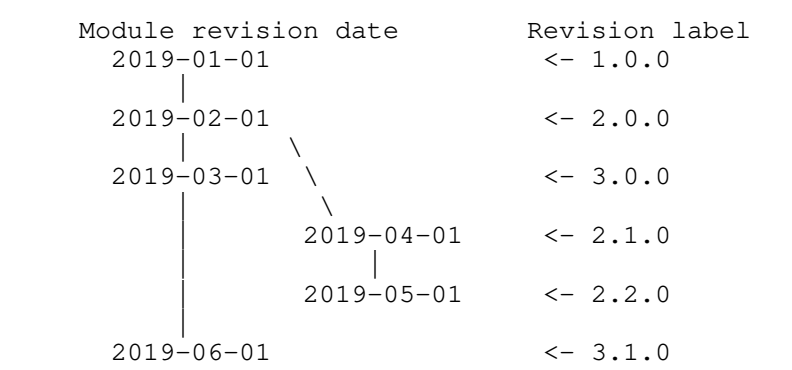
The ietf-revisions module defines the "revision-or-derived" extension statement, a substatement to the YANG "import" statement, to allow for a "minimum required revision" to be specified during import:

> The argument to the "revision-or-derived" extension statement is a revision date or a revision label.
>
> A particular revision of an imported module satisfies an import's "revision-or-derived" extension statement if the imported module's revision history contains a revision statement with a matching revision date or revision label.
>
> An "import" statement MUST NOT contain both a "revision-or-derived" extension statement and a "revision-date" statement.
>
> The "revision-or-derived" extension statement MAY be specified multiple times, allowing the import to use any module revision that satisfies at least one of the "revision-or-derived" extension statements.
>
> The "revision-or-derived" extension statement does not guarantee that all module revisions that satisfy an import statement are necessarily compatible; it only gives an indication that the revisions are more likely to be compatible. Hence, NBC changes to an imported module may also require new revisions of any importing modules, updated to accommodation those changes, along with updated import "revision-or-derived" extension statements to depend on the updated imported module revision.
>
> Adding, modifying or removing a "revision-or-derived" extension statement is considered to be a BC change.

4.1. Module import examples

Consider the example module "example-module" from Section 3.5 that is hypothetically available in the following revision/label pairings: 2019-01-01/1.0.0, 2019-02-01/2.0.0, 2019-03-01/3.0.0,

2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0. The relationship between the revisions is as before:



4.1.1. Example 1

This example selects module revisions that match, or are derived from the revision 2019-02-01. E.g., this dependency might be used if there was a new container added in revision 2019-02-01 that is augmented by the importing module. It includes revisions/labels: 2019-02-01/2.0.0, 2019-03-01/3.0.0, 2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0.

```
import example-module {
  rev:revision-or-derived 2019-02-01;
}
```

Alternatively, the first example could have used the revision label "2.0.0" instead, which selects the same set of revisions/labels.

```
import example-module {
  rev:revision-or-derived 2.0.0;
}
```

4.1.2. Example 2

This example selects module revisions that are derived from 2019-04-01 by using the revision label 2.1.0. It includes revisions/labels: 2019-04-01/2.1.0 and 2019-05-01/2.2.0. Even though 2019-06-01/3.1.0 has a higher revision label number than 2019-04-01/2.1.0 it is not a derived revision, and hence it is not a valid revision for import.

```
import example-module {
  rev:revision-or-derived 2.1.0;
}
```

4.1.3. Example 3

This example selects revisions derived from either 2019-04-01 or 2019-06-01. It includes revisions/labels: 2019-04-01/2.1.0, 2019-05-01/2.2.0, and 2019-06-01/3.1.0.

```
import example-module {
  rev:revision-or-derived 2019-04-01;
  rev:revision-or-derived 2019-06-01;
}
```

5. Updates to ietf-yang-library

This document updates YANG 1.1 [RFC7950] and YANG library [RFC8525] to clarify how ambiguous module imports are resolved. It also defines the YANG module, ietf-yang-library-revisions, that augments YANG library [RFC8525] with revision labels and two leafs to indicate how a server implements deprecated and obsolete schema nodes.

5.1. Resolving ambiguous module imports

A YANG datastore schema, defined in [RFC8525], can specify multiple revisions of a YANG module in the schema using the "import-only" list, with the requirement from [RFC7950] section 5.6.5 that only a single revision of a YANG module may be implemented.

If a YANG module import statement does not specify a specific revision within the datastore schema then it could be ambiguous as to which module revision the import statement should resolve to. Hence, a datastore schema constructed by a client using the information contained in YANG library may not exactly match the datastore schema actually used by the server.

The following two rules remove the ambiguity:

If a module import statement could resolve to more than one module revision defined in the datastore schema, and one of those revisions is implemented (i.e., not an "import-only" module), then the import statement MUST resolve to the revision of the module that is defined as being implemented by the datastore schema.

If a module import statement could resolve to more than one module revision defined in the datastore schema, and none of those revisions are implemented, then the import MUST resolve to the module revision with the latest revision date.

5.2. YANG library versioning augmentations

The "ietf-yang-library-revisions" YANG module has the following structure (using the notation defined in [RFC8340]):

```
module: ietf-yang-library-revisions
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module
            /yanglib:submodule:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:module-set
            /yanglib:import-only-module/yanglib:submodule:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:schema:
    +--ro deprecated-nodes-implemented?   boolean
    +--ro obsolete-nodes-absent?          boolean
```

5.2.1. Advertising revision-label

The ietf-yang-library-revisions YANG module augments the "module" and "submodule" lists in ietf-yang-library with "revision-label" leafs to optionally declare the revision label associated with each module and submodule.

5.2.2. Reporting how deprecated and obsolete nodes are handled

The ietf-yang-library-revisions YANG module augments YANG library with two boolean leafs to allow a server to report how it implements status "deprecated" and status "obsolete" schema nodes. The leafs are:

deprecated-nodes-implemented: If set to "true", this leaf indicates that all schema nodes with a status "deprecated" are implemented equivalently as if they had status "current"; otherwise deviations MUST be used to explicitly remove "deprecated" nodes from the schema. If this leaf is set to "false" or absent, then the behavior is unspecified.

obsolete-nodes-absent: If set to "true", this leaf indicates that the server does not implement any status "obsolete" schema nodes. If this leaf is set to "false" or absent, then the behaviour is unspecified.

Servers SHOULD set both the "deprecated-nodes-implemented" and "obsolete-nodes-absent" leafs to "true".

If a server does not set the "deprecated-nodes-implemented" leaf to "true", then clients MUST NOT rely solely on the "rev:non-backwards-compatible" statements to determine whether two module revisions are backwards-compatible, and MUST also consider whether the status of any nodes has changed to "deprecated" and whether those nodes are implemented by the server.

6. Versioning of YANG instance data

Instance data sets [I-D.ietf-netmod-yang-instance-file-format] do not directly make use of the updated revision handling rules described in this document, as compatibility for instance data is undefined.

However, instance data specifies the content-schema of the data-set. This schema SHOULD make use of versioning using revision dates and/or revision labels for the individual YANG modules that comprise the schema or potentially for the entire schema itself (e.g., [I-D.ietf-netmod-yang-packages]).

In this way, the versioning of a content-schema associated with an instance data set may help a client to determine whether the instance data could also be used in conjunction with other revisions of the YANG schema, or other revisions of the modules that define the schema.

7. Guidelines for using the YANG module update rules

The following text updates section 4.7 of [RFC8407] to revise the guidelines for updating YANG modules.

7.1. Guidelines for YANG module authors

All IETF YANG modules MUST include revision-label statements for all newly published YANG modules, and all newly published revisions of existing YANG modules. The revision-label MUST take the form of a YANG semantic version number [I-D.ietf-netmod-yang-semver].

NBC changes to YANG modules may cause problems to clients, who are consumers of YANG models, and hence YANG module authors SHOULD minimize NBC changes and keep changes BC whenever possible.

When NBC changes are introduced, consideration should be given to the impact on clients and YANG module authors SHOULD try to mitigate that impact.

A "rev:non-backwards-compatible" statement MUST be added if there are NBC changes relative to the previous revision.

Removing old revision statements from a module's revision history could break import by revision, and hence it is RECOMMENDED to retain them. If all dependencies have been updated to not import specific revisions of a module, then the corresponding revision statements can be removed from that module. An alternative solution, if the revision section is too long, would be to remove, or curtail, the older description statements associated with the previous revisions.

The "rev:revision-or-derived" extension SHOULD be used in YANG module imports to indicate revision dependencies between modules in preference to the "revision-date" statement, which causes overly strict import dependencies and SHOULD NOT be used.

A module that includes submodules SHOULD use the "revision-date" statement to include specific submodule revisions. The revision of the including module MUST be updated when any included submodule has changed.

In some cases a module or submodule revision that is not strictly NBC by the definition in Section 3.1.2 of this specification may include the "non-backwards-compatible" statement. Here is an example when adding the statement may be desirable:

- A "config false" leaf had its value space expanded (for example, a range was increased, or additional enum values were added) and the author or server implementor feels there is a significant compatibility impact for clients and users of the module or submodule

7.1.1. Making non-backwards-compatible changes to a YANG module

There are various valid situations where a YANG module has to be modified in an NBC way. Here are the different ways in which this can be done:

- NBC changes can be sometimes be done incrementally using the "deprecated" status to provide clients time to adapt to NBC changes.

- NBC changes are done at once, i.e. without using "status" statements. Depending on the change, this may have a big impact on clients.

- If the server can support multiple revisions of the YANG module or of YANG packages (as specified in [I-D.ietf-netmod-yang-packages]), and allows the client to select the revision (as per [I-D.ietf-netmod-yang-ver-selection]), then NBC changes MAY be done without using "status" statements.

Clients would be required to select the revision which they support and the NBC change would have no impact on them.

Here are some guidelines on how non-backwards-compatible changes can be made incrementally, with the assumption that deprecated nodes are implemented by the server, and obsolete nodes are not:

1. The changes should be made gradually, e.g., a data node's status SHOULD NOT be changed directly from "current" to "obsolete" (see Section 4.7 of [RFC8407]), instead the status SHOULD first be marked "deprecated". At some point in the future, when support is removed for the data node, there are two options. The first, and preferred, option is to keep the data node definition in the model and change the status to "obsolete". The second option is to simply remove the data node from the model, but this has the risk of breaking modules which import the modified module, and the removed identifier may be accidently reused in a future revision.

2. For deprecated data nodes the "description" statement SHOULD also indicate until when support for the node is guaranteed (if known). If there is a replacement data node, rpc, action or notification for the deprecated node, this SHOULD be stated in the "description". The reason for deprecating the node can also be included in the "description" if it is deemed to be of potential interest to the user.

3. For obsolete data nodes, it is RECOMMENDED to keep the above information, from when the node had status "deprecated", which is still relevant.

4. When obsoleting or deprecating data nodes, the "deprecated" or "obsolete" status SHOULD be applied at the highest possible level in the data tree. For clarity, the "status" statement SHOULD also be applied to all descendent data nodes, but the additional status related information does not need to be repeated if it does not introduce any additional information.

5. NBC changes which can break imports SHOULD be avoided because of the impact on the importing module. The importing modules could get broken, e.g., if an augmented node in the importing module has been removed from the imported module. Alternatively, the schema of the importing modules could undergo an NBC change due to the NBC change in the imported module, e.g., if a node in a grouping has been removed. As described in Appendix B.1, instead of removing a node, that node SHOULD first be deprecated and then obsoleted.

See Appendix B for examples on how NBC changes can be made.

7.2. Versioning Considerations for Clients

Guidelines for clients of modules using the new module revision update procedure:

- Clients SHOULD be liberal when processing data received from a server. For example, the server may have increased the range of an operational node causing the client to receive a value which is outside the range of the YANG model revision it was coded against.

- Clients SHOULD monitor changes to published YANG modules through their revision history, and use appropriate tooling to understand the specific changes between module revision. In particular, clients SHOULD NOT migrate to NBC revisions of a module without understanding any potential impact of the specific NBC changes.

- Clients SHOULD plan to make changes to match published status changes. When a node's status changes from "current" to "deprecated", clients SHOULD plan to stop using that node in a timely fashion. When a node's status changes to "obsolete", clients MUST stop using that node.

8. Module Versioning Extension YANG Modules

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

```
<CODE BEGINS> file "ietf-yang-revisions@2021-11-04.yang"
module ietf-yang-revisions {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-revisions";
  prefix rev;

  // RFC Ed.: We need the bis version to get the new type revision-identifier
  // If 6991-bis is not yet an RFC we need to copy the definition here
  import ietf-yang-types {
    prefix yang;
    reference
      "XXXX [ietf-netmod-rfc6991-bis]: Common YANG Data Types";
  }

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

```
    Author:   Joe Clarke
              <mailto:jclarke@cisco.com>

    Author:   Reshad Rahman
              <mailto:reshad@yahoo.com>

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

    Author:   Balazs Lengyel
              <mailto:balazs.lengyel@ericsson.com>

    Author:   Jason Sterne
              <mailto:jason.sterne@nokia.com>";
  description
    "This YANG 1.1 module contains definitions and extensions to
    support updated YANG revision handling.

    Copyright (c) 2021 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

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

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

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

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

  revision 2021-11-04 {
    rev:revision-label 1.0.0-draft-ietf-netmod-yang-module-versioning-05;
    description
      "Initial version.";
    reference
      "XXXX: Updated YANG Module Revision Handling";
```

```
  }

  typedef revision-label {
    type string {
      length "1..255";
      pattern '[a-zA-Z0-9,\-_.+]+';
      pattern '\d{4}-\d{2}-\d{2}' {
        modifier invert-match;
      }
    }
    description
      "A label associated with a YANG revision.

      Alphanumeric characters, comma, hyphen, underscore, period
      and plus are the only accepted characters. MUST NOT match
      revision-date.";
    reference
      "XXXX: Updated YANG Module Revision Handling;
      Section 3.3, Revision label";
  }

  typedef revision-date-or-label {
    type union {
      type yang:revision-identifier;
      type revision-label;
    }
    description
      "Represents either a YANG revision date or a revision label";
  }

  extension non-backwards-compatible {
    description
      "This statement is used to indicate YANG module revisions that
      contain non-backwards-compatible changes.

      The statement MUST only be a substatement of the 'revision'
      statement.  Zero or one 'non-backwards-compatible' statements
      per parent statement is allowed.  No substatements for this
      extension have been standardized.

      If a revision of a YANG module contains changes, relative to
      the preceding revision in the revision history, that do not
      conform to the backwards compatible module update rules defined
      in RFC-XXX, then the 'non-backwards-compatible' statement MUST
      be added as a substatement to the revision statement.

      Conversely, if a revision does not contain a
      'non-backwards-compatible' statement then all changes,
```

```
      relative to the preceding revision in the revision history,
      MUST be backwards-compatible.

      A new module revision that only contains changes that are
      backwards compatible SHOULD NOT include the
      'non-backwards-compatible' statement.  An example of when
      an author might add the 'non-backwards-compatible' statement
      is if they believe a change could negatively impact clients
      even though the backwards compatibility rules defined in
      RFC-XXXX classify it as a backwards-compatible change.

      Add, removing, or changing a 'non-backwards-compatible'
      statement is a backwards-compatible version change.";

    reference
      "XXXX: Updated YANG Module Revision Handling;
      Section 3.2, non-backwards-compatible revision extension statement";
  }

  extension revision-label {
    argument revision-label;
    description
      "The revision label can be used to provide an additional
      versioning identifier associated with a module or submodule
      revision.  One such scheme that
      could be used is [XXXX: ietf-netmod-yang-semver].

      The format of the revision-label argument MUST conform to the
      pattern defined for the revision-label typedef in this module.

      The statement MUST only be a substatement of the revision
      statement.  Zero or one revision-label statements per parent
      statement are allowed.  No substatements for this extension
      have been standardized.

      Revision labels MUST be unique amongst all revisions of a
      module or submodule.

      Adding a revision label is a backwards-compatible version
      change.  Changing or removing an existing revision label in
      the revision history is a non-backwards-compatible version
      change, because it could impact any references to that
      revision label.";

    reference
      "XXXX: Updated YANG Module Revision Handling;
      Section 3.3, Revision label";
  }
```

```
  extension revision-label-scheme {
    argument revision-label-scheme-base;
    description
      "The revision label scheme specifies which revision-label scheme
      the module or submodule uses.

      The mandatory revision-label-scheme-base argument MUST be an
      identity derived from revision-label-scheme-base.

      This extension is only valid as a top-level statement, i.e.,
      given as as a substatement to 'module' or 'submodule'.  No
      substatements for this extension have been standardized.

      This extension MUST be used if there is a revision-label
      statement in the module or submodule.

      Adding a revision label scheme is a backwards-compatible version
      change.  Changing a revision label scheme is a
      non-backwards-compatible version change, unless the new revision
      label scheme is backwards-compatible with the replaced revision
      label scheme.  Removing a revision label scheme is a
      non-backwards-compatible version change.";

    reference
      "XXXX: Updated YANG Module Revision Handling;
      Section 3.3.1, Revision label scheme extension statement";
  }

  extension revision-or-derived {
    argument revision-date-or-label;
    description
      "Restricts the revision of the module that may be imported to
      one that matches or is derived from the specified
      revision-date or revision-label.

      The argument value MUST conform to the
      'revision-date-or-label' defined type.

      The statement MUST only be a substatement of the import
      statement.  Zero, one or more 'revision-or-derived' statements
      per parent statement are allowed.  No substatements for this
      extension have been standardized.

      If specified multiple times, then any module revision that
      satisfies at least one of the 'revision-or-derived' statements
      is an acceptable revision for import.

      An 'import' statement MUST NOT contain both a
```

```
      'revision-or-derived' extension statement and a
      'revision-date' statement.

      A particular revision of an imported module satisfies an
      import's 'revision-or-derived' extension statement if the
      imported module's revision history contains a revision
      statement with a matching revision date or revision label.

      The 'revision-or-derived' extension statement does not
      guarantee that all module revisions that satisfy an import
      statement are necessarily compatible, it only gives an
      indication that the revisions are more likely to be
      compatible.

      Adding, removing or updating a 'revision-or-derived'
      statement to an import is a backwards-compatible change.
      ";

    reference
      "XXXX: Updated YANG Module Revision Handling;
      Section 4, Import by derived revision";
  }

  identity revision-label-scheme-base {
    description
      "Base identity from which all revision label schemes are
      derived.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
        Section 3.3.1, Revision label scheme extension statement";

  }
}
<CODE ENDS>
```

YANG module with augmentations to YANG Library to revision labels

```
<CODE BEGINS> file "ietf-yang-library-revisions@2021-11-04.yang"
module ietf-yang-library-revisions {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-library-revisions";
  prefix yl-rev;

  import ietf-yang-revisions {
    prefix rev;
    reference
```

```
      "XXXX: Updated YANG Module Revision Handling";
  }

  import ietf-yang-library {
    prefix yanglib;
    reference "RFC 8525: YANG Library";
  }

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

     Author:   Joe Clarke
               <mailto:jclarke@cisco.com>

     Author:   Reshad Rahman
               <mailto:reshad@yahoo.com>

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

     Author:   Balazs Lengyel
               <mailto:balazs.lengyel@ericsson.com>

     Author:   Jason Sterne
               <mailto:jason.sterne@nokia.com>";
  description
    "This module contains augmentations to YANG Library to add module
     level revision label and to provide an indication of how
     deprecated and obsolete nodes are handled by the server.

     Copyright (c) 2021 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
```

```
     'MAY', and 'OPTIONAL' in this document are to be interpreted as
     described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.";

  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  // RFC Ed.: replace XXXX (including in the imports above) with
  // actual RFC number and remove this note.
  // RFC Ed.: please replace revision-label version with 1.0.0 and
  // remove this note.
  revision 2021-11-04 {
    rev:revision-label 1.0.0-draft-ietf-netmod-yang-module-versioning-05;
    description
      "Initial revision";
    reference
      "XXXX: Updated YANG Module Revision Handling";
  }

  // library 1.0 modules-state is not augmented with revision-label

  augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
    description
      "Add a revision label to module information";
    leaf revision-label {
      type rev:revision-label;
      description
        "The revision label associated with this module revision.
         The label MUST match the rev:revision-label value in the specific
         revision of the module loaded in this module-set.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 5.2.1, Advertising revision-label";
    }
  }

  augment "/yanglib:yang-library/yanglib:module-set/yanglib:module/"
          + "yanglib:submodule" {
    description
      "Add a revision label to submodule information";
    leaf revision-label {
      type rev:revision-label;
      description
        "The revision label associated with this submodule revision.
         The label MUST match the rev:revision-label value in the specific
         revision of the submodule included by the module loaded in
         this module-set.";
```

```
      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 5.2.1, Advertising revision-label";
    }
  }

  augment "/yanglib:yang-library/yanglib:module-set/"
          + "yanglib:import-only-module" {
    description
      "Add a revision label to module information";
    leaf revision-label {
      type rev:revision-label;
      description
        "The revision label associated with this module revision.
         The label MUST match the rev:revision-label value in the specific
         revision of the module included in this module-set.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 5.2.1, Advertising revision-label";
    }
  }

  augment "/yanglib:yang-library/yanglib:module-set/"
          + "yanglib:import-only-module/yanglib:submodule" {
    description
      "Add a revision label to submodule information";
    leaf revision-label {
      type rev:revision-label;
      description
        "The revision label associated with this submodule revision.
         The label MUST match the rev:label value in the specific
         revision of the submodule included by the
         import-only-module loaded in this module-set.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 5.2.1, Advertising revision-label";
    }
  }

  augment "/yanglib:yang-library/yanglib:schema" {
    description
      "Augmentations to the ietf-yang-library module to indicate how
       deprecated and obsoleted nodes are handled for each datastore
       schema supported by the server.";

    leaf deprecated-nodes-implemented {
```

```
      type boolean;
      description
        "If set to true, this leaf indicates that all schema nodes with
         a status 'deprecated' are implemented
         equivalently as if they had status 'current'; otherwise
         deviations MUST be used to explicitly remove deprecated
         nodes from the schema.  If this leaf is absent or set to false,
         then the behavior is unspecified.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 5.2.2, Reporting how deprecated and obsolete nodes
         are handled";
    }

    leaf obsolete-nodes-absent {
      type boolean;
      description
        "If set to true, this leaf indicates that the server does not
         implement any status 'obsolete' schema nodes.  If this leaf is
         absent or set to false, then the behaviour is unspecified.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 5.2.2, Reporting how deprecated and obsolete nodes
         are handled";
    }
  }
}
<CODE ENDS>
```

9. Contributors

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

- Balazs Lengyel
- Benoit Claise
- Bo Wu
- Ebben Aries
- Jan Lindblad

- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu
- Reshad Rahman
- Rob Wilton

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update]. We would like to thank Kevin D'Souza and Benoit Claise for their initial work in this problem space.

Discussons on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models. We would like to thank both Anees Shaikh and Rob Shakir for their input into this problem space.

We would also like to thank Lou Berger, Andy Bierman, Martin Bjorklund, Italo Busi, Tom Hill, Scott Mansfield, Kent Watsen for their contributions and review comments.

10. Security Considerations

The document does not define any new protocol or data model. There are no security considerations beyond those specified in [RFC7950] and [RFC6020].

11. IANA Considerations

11.1. YANG Module Registrations

This document requests IANA to registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations are requested.

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

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

The following YANG module is requested to be registred in the "IANA Module Names" [RFC6020]. Following the format in RFC 6020, the following registrations are requested:

The ietf-yang-revisions module:

Name: ietf-yang-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-revisions

Prefix: rev

Reference: [RFCXXXX]

The ietf-yang-library-revisions module:

Name: ietf-yang-library-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-library-revisions

Prefix: yl-rev

Reference: [RFCXXXX]

11.2. Guidance for versioning in IANA maintained YANG modules

Note for IANA (to be removed by the RFC editor): Please check that the registries and IANA YANG modules are referenced in the appropriate way.

IANA is responsible for maintaining and versioning YANG modules that are derived from other IANA registries. For example, "iana-if-type.yang" [IfTypeYang] is derived from the "Interface Types (ifType) IANA registry" [IfTypesReg], and "iana-routing-types.yang" [RoutingTypesYang] is derived from the "Address Family Numbers" [AddrFamilyReg] and "Subsequent Address Family Identifiers (SAFI) Parameters" [SAFIReg] IANA registries.

Normally, updates to the registries cause any derived YANG modules to be updated in a backwards-compatible way, but there are some cases where the registry updates can cause non-backward-compatible updates to the derived YANG module. An example of such an update is the 2020-12-31 revision of iana-routing-types.yang

[RoutingTypesDecRevision], where the enum name for two SAFI values was changed.

In all cases, IANA MUST follow the versioning guidance specified in Section 3.1, and MUST include a "rev:non-backwards-compatible" substatement to the latest revision statement whenever an IANA maintained module is updated in a non-backwards-compatible way, as described in Section 3.2.

Note: For published IANA maintained YANG modules that contain non-backwards-compatible changes between revisions, a new revision should be published with the "rev:non-backwards-compatible" substatement retrospectively added to any revisions containing non-backwards-compatible changes.

Non-normative examples of updates to enumeration types in IANA maintained modules that would be classified as non-backwards-compatible changes are: Changing the status of an enumeration typedef to obsolete, changing the status of an enum entry to obsolete, removing an enum entry, changing the identifier of an enum entry, or changing the described meaning of an enum entry.

Non-normative examples of updates to enumeration types in IANA maintained modules that would be classified as backwards-compatible changes are: Adding a new enum entry to the end of the enumeration, changing the status or an enum entry to deprecated, or improving the description of an enumeration that does not change its defined meaning.

Non-normative examples of updates to identity types in IANA maintained modules that would be classified as non-backwards-compatible changes are: Changing the status of an identity to obsolete, removing an identity, renaming an identity, or changing the described meaning of an identity.

Non-normative examples of updates to identity types in IANA maintained modules that would be classified as backwards-compatible changes are: Adding a new identity, changing the status or an identity to deprecated, or improving the description of an identity that does not change its defined meaning.

12. References

12.1. Normative References

[I-D.ietf-netmod-rfc6991-bis]
Schoenwaelder, J., "Common YANG Data Types", draft-ietf-netmod-rfc6991-bis-08 (work in progress), November 2021.

[I-D.ietf-netmod-yang-semver]
Clarke, J., Wilton, R., Rahman, R., Lengyel, B., Sterne, J., and B. Claise, "YANG Semantic Versioning", draft-ietf-netmod-yang-semver-04 (work in progress), October 2021.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

12.2. Informative References

[AddrFamilyReg]
"Address Family Numbers IANA Registry", <https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>.

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", draft-clacla-netmod-yang-model-update-06 (work in progress), July 2018.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-21 (work in progress), October 2021.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", draft-ietf-netmod-yang-packages-02 (work in progress), October 2021.

[I-D.ietf-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-ietf-netmod-yang-solutions-01 (work in progress), November 2020.

[I-D.ietf-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Schema Selection", draft-ietf-netmod-yang-ver-selection-00 (work in progress), March 2020.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-ietf-netmod-yang-versioning-reqs-05 (work in progress), July 2021.

[IfTypesReg]
"Interface Types (ifType) IANA Registry", <https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-5>.

[IfTypeYang]
"iana-if-type YANG Module", <https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml>.

[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/info/rfc8340>.

[RoutingTypesDecRevision]
"2020-12-31 revision of iana-routing-types.yang", <https://www.iana.org/assignments/yang-parameters/iana-routing-types@2020-12-31.yang>.

[RoutingTypesYang]
"iana-routing-types YANG Module", <https://www.iana.org/assignments/iana-routing-types/iana-routing-types.xhtml>.

[SAFIReg] "Subsequent Address Family Identifiers (SAFI) Parameters IANA Registry", <https://www.iana.org/assignments/safi-namespace/safi-namespace.xhtml>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

Appendix A. Examples of changes that are NBC

Examples of NBC changes include:

- Deleting a data node, or changing it to status obsolete.
- Changing the name, type, or units of a data node.
- Modifying the description in a way that changes the semantic meaning of the data node.
- Any changes that change or reduce the allowed value set of the data node, either through changes in the type definition, or the addition or changes to "must" statements, or changes in the description.
- Adding or modifying "when" statements that reduce when the data node is available in the schema.
- Making the statement conditional on if-feature.

Appendix B. Examples of applying the NBC change guidelines

The following sections give steps that could be taken for making NBC changes to a YANG module or submodule using the incremental approach described in section Section 7.1.1.

The examples are all for "config true" nodes.

Alternatively, the NBC changes MAY be done non-incrementally and without using "status" statements if the server can support multiple revisions of the YANG module or of YANG packages. Clients would be required to select the revision which they support and the NBC change would have no impact on them.

B.1. Removing a data node

Removing a leaf or container from the data tree, e.g., because support for the corresponding feature is being removed:

1. The schema node's status is changed to "deprecated" and the node is supported for some period of time (e.g. one year). This is a BC change.

2. When the schema node is not supported anymore, its status is changed to "obsolete" and the "description" updated. This is an NBC change.

B.2. Changing the type of a leaf node

Changing the type of a leaf node. e.g., a "vpn-id" node of type integer being changed to a string:

1. The status of schema node "vpn-id" is changed to "deprecated" and the node is supported for some period of time (e.g. one year). This is a BC change. The description is updated to indicate that "vpn-name" is replacing this node.

2. A new schema node, e.g., "vpn-name", of type string is added to the same location as the existing node "vpn-id". This new node has status "current" and its description explains that it is replacing node "vpn-id".

3. During the period of time when both schema nodes are supported, the interactions between the two nodes is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server may prevent the new node from being set if the old node is already set (and vice-versa). A "choice" construction could be used, or the new node may have a "when" statement to achieve this. The old node must not have a "when" statement since this would be an NBC change, but the server could reject the old node from being set if the new node is already set.

    2. If the new node is set and a client does a get or get-config operation on the old node, the server could map the value. For example, if the new node "vpn-name" has value "123" then the server could return integer value 123 for the old node "vpn-id". However, if the value can not be mapped then the configuration would be incomplete. The behavior in this case is outside the scope of this document.

4. When the schema node "vpn-id" is not supported anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

B.3. Reducing the range of a leaf node

Reducing the range of values of a leaf-node, e.g., consider a "vpn-id" schema node of type uint32 being changed from range 1..5000 to range 1..2000:

1. If all values which are being removed were never supported, e.g., if a vpn-id of 2001 or higher was never accepted, this is a BC change for the functionality (no functionality change). Even if it is an NBC change for the YANG model, there should be no impact for clients using that YANG model.

2. If one or more values being removed was previously supported, e.g., if a vpn-id of 3333 was accepted previously, this is an NBC change for the YANG model. Clients using the old YANG model will be impacted, so a change of this nature should be done carefully, e.g., by using the steps described in Appendix B.2

B.4. Changing the key of a list

Changing the key of a list has a big impact to the client. For example, consider a "sessions" list which has a key "interface" and there is a need to change the key to "dest-address". Such a change can be done in steps:

1. The status of list "sessions" is changed to "deprecated" and the list is supported for some period of time (e.g. one year). This is a BC change. The description is updated to indicate the new list that is replacing this list.

2. A new list is created in the same location with the same descendant schema nodes but with "dest-address" as key. Finding an appropriate name for the new list can be difficult. In this case the new list is called "sessions-address", has status "current" and its description should explain that it is replacing list "session".

3. During the period of time when both lists are supported, the interactions between the two lists is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server could prevent entries in the new list from being created if the old list already has entries (and vice-versa).

    2. If the new list has entries created and a client does a get or get-config operation on the old list, the server could map the entries. However, if the new list has entries which

would lead to duplicate keys in the old list, the mapping can not be done.

4. When list "sessions" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

If the server can support NBC revisions of the YANG module simultaneously using version selection [I-D.ietf-netmod-yang-ver-selection], then the changes can be done immediately:

1. The new revision of the YANG module has the list "sessions" modified to have "dest-address" as key, this is an NBC change.

2. Clients which require the previous functionality select the older module revision

B.5. Renaming a node

A leaf or container schema node may be renamed, either due to a spelling error in the previous name or because of a better name. For example a node "ip-adress" could be renamed to "ip-address":

1. The status of the existing node "ip-adress" is changed to "deprecated" and is supported for some period of time (e.g. one year). This is a BC change. The description is updated to indicate the node that is replacing this node.

2. The new schema node "ip-address" is added to the same location as the existing node "ip-adress". This new node has status "current" and its description should explain that it is replacing node "ip-adress".

3. During the period of time when both nodes are available, the interactions between the two nodes is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server may prevent the new node from being set if the old node is already set (and vice-versa). A "choice" construction could be used, or the new node may have a "when" statement to achieve this. The old node must not have a "when" statement since this would be an NBC change, but the server could reject the old node from being set if the new node is already set.

2. If the new node is set and a client does a get or get-config operation on the old node, the server could use the value of the new node. For example, if the new node "ip-address" has value X then the server may return value X for the old node "ip-adress".

4. When node "ip-adress" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Reshad Rahman (editor)

Email: reshad@yahoo.com

Balazs Lengyel (editor)
Ericsson

Email: balazs.lengyel@ericsson.com

Joe Clarke
Cisco Systems, Inc.

Email: jclarke@cisco.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2022

R. Wilton, Ed.
R. Rahman
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia
B. Wu, Ed.
Huawei

4 March 2022

YANG Packages
draft-ietf-netmod-yang-packages-03

Abstract

This document defines YANG packages; a versioned organizational structure used to manage schema and conformance of YANG modules as a cohesive set instead of individually.

It describes how packages: are represented on a server, can be defined in offline YANG instance data files, and can be used to define the content schema associated with YANG instance data files.

Table of Contents

1. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

This document also makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

* datastore schema

This document also makes of the following terminology introduced in the YANG 1.1 Data Modeling Language [RFC7950]:

* data node

* schema node

In addition, this document defines the following terminology:

* YANG package: a versioned organizational structure used to manage a set of YANG modules that collectively define a package schema. YANG packages are defined in Section 5.

* package schema: The combined set of schema nodes defined by a YANG package. Package schema can be used to define datastore schema.

* backwards-compatible (BC) change: When used in the context of a YANG module, it follows the definition in Section 3.1.1 of [I-D.ietf-netmod-yang-module-versioning]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.2.

* non-backwards-compatible (NBC) change: When used in the context of a YANG module, it follows the definition in Section 3.1.2 of [I-D.ietf-netmod-yang-module-versioning]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.2.

* editorial change: When used in the context of a YANG module, it follows the definition of an 'editorial change' in 3.2 of [I-D.ietf-netmod-yang-module-versioning]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.3.

2. Introduction

This document defines and describes the YANG [RFC7950] constructs that are used to define and use YANG packages.

A YANG package is a versioned organizational structure used to manage a set of YANG modules that collectively define a package schema. For example, a YANG package could contain the set of YANG modules required to implement an L2VPN service on a network device.

Non-normative examples of YANG packages are provided in the appendices.

3. Background on YANG packages

It has long been acknowledged within the YANG community that network management using YANG requires a unit of organization and conformance that is broader in scope than individual YANG modules.

'The YANG Package Statement' [I-D.bierman-netmod-yang-package] proposed a YANG package mechanism based on new YANG language statements, where a YANG package is defined in a file similar to how YANG modules are defined, and would require enhancements to YANG compilers to understand the new statements used to define packages.

OpenConfig [openconfigsemver] describes an approach to versioning 'bundle releases' based on git tags. I.e. a set of modules, at particular versions, can be marked with the same release tag to indicate that they are known to interoperate together.

The NETMOD WG in general, and the YANG versioning design team in particular, are exploring solutions [I-D.ietf-netmod-yang-solutions] to the YANG versioning requirements, [I-D.ietf-netmod-yang-versioning-reqs]. Solutions to the versioning requirements can be split into several distinct areas. [I-D.ietf-netmod-yang-module-versioning] is focused on YANG

versioning scoped to individual modules. The overall solution must also consider YANG versioning and conformance scoped to sets of modules. YANG packages provide part of the solution for versioning sets of modules.

4. Objectives

The main goals of YANG package definitions include, but are not restricted to:

* To provide an alternative, simplified, YANG conformance mechanism. Rather than conformance being performed against a set of individual YANG module revisions, features, and deviations, conformance can be more simply stated in terms of YANG packages, with a set of modifications (e.g. additional modules, deviations, or features).

* To allow datastore schema to be specified in a concise way rather than having each server explicitly list all modules, revisions, and features. YANG package definitions can be defined in documents that are available offline, and accessible via a URL, rather than requiring explicit lists of modules to be shared between client and server. Hence, a YANG package must contain sufficient information to allow a client or server to precisely construct the schema associated with the package.

* To define a mainly linear versioned history of sets of modules versions that are known to work together. I.e. to help mitigate the problem where a client must manage devices from multiple vendors, and vendor A implements version 1.0.0 of module foo and version 2.0.0 of module bar, and vendor B implements version 2.0.0 of module foo and version 1.0.0 of module bar. For a client, trying to interoperate with multiple vendors, and many YANG modules, finding a consistent lowest common denominator set of YANG module versions may be difficult, if not impossible.

Protocol mechanisms of how clients can negotiate which packages or package versions are to be used for NETCONF/RESTCONF communications are outside the scope of this document, and are defined in [I-D.ietf-netmod-yang-ver-selection].

Finally, the package definitions proposed by this document are intended to be relatively basic in their definition and the functionality that they support. As industry gains experience using YANG packages, the standard YANG mechanisms of updating, or augmenting YANG modules could also be used to extend the functionality supported by YANG packages, if required.

5. YANG Package Definition

This document specifies an approach to defining YANG packages that is different to either of the approaches described in the background.

A YANG package is a versioned organizational structure used to manage a set of YANG modules that collectively define a package schema.

Each YANG package has a name that SHOULD end with the suffix "-pkg". Package names are normally expected to be globally unique, but in some cases the package name may be locally scoped to a server or device, as described in Section 5.5.

YANG packages are versioned using the same approaches described in [I-D.ietf-netmod-yang-module-versioning] and [I-D.ietf-netmod-yang-semver]. This is described in further detail in Section 5.2.

Each YANG package version, defines:

* some metadata about the package, e.g., description, tags, scoping, referential completeness, location information.

* a set of YANG modules, at particular revisions, that are implemented by servers that implement the package. The modules may contain deviations.

* a set of import-only YANG modules, at particular revisions, that are used 'import-only' by the servers that implement the package.

* a set of included YANG packages, at particular revisions, that are also implemented by servers that implement the package.

* a set of YANG module features that must be supported by servers that implement the package.

The structure for YANG package definitions uses existing YANG language statements, YANG Data Structure Extensions [I-D.ietf-netmod-yang-data-ext], and YANG Instance Data File Format [I-D.ietf-netmod-yang-instance-file-format].

YANG package definitions are available offline in YANG instance data files. Client applications can be designed to support particular package versions that they expect to interoperate with.

YANG package definitions are available from the server via augmentations to YANG Library [RFC8525]. Rather than client applications downloading the entire contents of YANG library to

confirm that the server's datastore schema are compatible with the client, they can simply check the names and versions of the packages advertised in YANG library to know what schema to expect in the server datastores.

YANG package definitions can also be used to define the content schema associated with YANG instance data files holding other, e.g., non packages related, instance data.

5.1. Package definition rules

Packages are defined using the following rules:

1. A YANG package MAY represent a referentially complete set of modules or MAY represent a set of modules with some module import dependencies missing, as described in Section 5.4.

2. Packages definitions are hierarchical. A package can include other packages. Only a single version of a package can be included, and conflicting package includes (e.g. from descendant package includes) MUST be explicitly resolved by indicating which version takes precedence, and which versions are being replaced.

3. YANG packages definitions MAY include modules containing deviation statements, but those deviation statements MUST only be used in an [RFC7950] compatible way to indicate where a server, or class of servers, deviates from a published standard. Deviations MUST NOT be included in a package definition that is part of a published standard. See section Section 5.8.1 for further guidance on the use of deviations in YANG packages.

4. For each module implemented by a package, only a single revision of that module MUST be implemented. Multiple revisions of a module MAY be listed as import-only dependencies.

5. The revision of a module listed in the package 'module' list supersedes any 'implemented' revision of the module listed in an included package module list. The 'replaces-revision' leaf-list is used to indicate which 'implemented' or 'import-only' module revisions are replaces by this module revision. This allows a package to explicitly resolve conflicts between implemented module revisions in included packages.

6. The 'replaces-revision' leaf-list in the 'import-only-module' list can be used to exclude duplicate revisions of import-only modules from included packages. Otherwise, the import-only-modules for a package are the import-only-modules from all included packages combined with any modules listed in the packages import-only-module list.

5.2. Package versioning

Individual versions of a YANG package are versioned using the "revision-label" scheme defined in section 3.3 of [I-D.ietf-netmod-yang-module-versioning].

5.2.1. Updating a package with a new version

Package compatibility is fundamentally defined by how the package schema between two package versions has changed.

When a package definition is updated, the version associated with the package MUST be updated appropriately, taking into consideration the scope of the changes as defined by the rules below.

5.2.1.1. Non-Backwards-compatible changes

The following changes classify as non-backwards-compatible changes to a package definition:

* Changing an 'included-package' list entry to select a package version that is non-backwards-compatible to the prior package version, or removing a previously included package.

* Changing a 'module' or 'import-only-module' list entry to select a module revision that is non-backwards-compatible to the prior module revision, or removing a previously implemented module.

* Removing a feature from the 'mandatory-feature' leaf-list.

* Adding, changing, or removing a module containing one or more deviations, that when applied to the target module would create a change that is considered a non-backwards-compatible change to the affected data node in the schema associated with the prior package version.

5.2.1.2. Backwards-compatible changes

The following changes classify as backwards-compatible changes to a package definition:

* Changing an 'included-package' list entry to select a package version that is backwards-compatible to the prior package version, or including a new package that does not conflict with any existing included package or module.

* Changing a 'module' or 'import-only-module' list entry to select a module revision that is backwards-compatible to the prior module revision, or including a new module to the package definition.

* Adding a feature to the 'mandatory-feature' leaf-list.

* Adding, changing, or removing a module containing one or more deviations, that when applied to the target module would create a change that is considered a backwards-compatible change to the affected data node in the schema associated with the prior package version.

5.2.1.3. Editorial changes

The following changes classify as editorial changes to a package definition:

* Changing a 'included-package' list entry to select a package version that is classified as an editorial change relative to the prior package version.

* Changing a 'module' or 'import-only-module' list entry to select a module revision that is classified as an editorial change relative to the prior module revision.

* Any change to any metadata associated with a package definition.

5.2.2. YANG Semantic Versioning for packages

YANG Semantic Versioning [I-D.ietf-netmod-yang-semver] MAY be used as an appropriate type of revision-label for the package version leaf.

If the format of the leaf matches the 'ysver:version' type specified in ietf-yang-semver.yang, then the package version leaf MUST be interpreted as a YANG semantic version number.

For YANG packages defined by the IETF, YANG semantic version numbers MUST be used as the version scheme for YANG packages.

The rules for incrementing the YANG package version number are equivalent to the semantic versioning rules used to version individual YANG modules, defined in section 3.2 of [I-D.ietf-netmod-yang-semver], but use the rules defined previously

in Section 5.2.1 to determine whether a change is classified as non-backwards-compatible, backwards-compatible, or editorial. Where available, the semantic version number of the referenced elements in the package (included packages or modules) can be used to help determine the scope of changes being made.

5.3. Package conformance

YANG packages allows for conformance to be checked at a package level rather than requiring a client to download all modules, revisions, and deviations from the server to ensure that the datastore schema used by the server is compatible with the client.

YANG package conformance is analogous to how YANG [RFC7950] requires that servers either implement a module faithfully, or otherwise use deviations to indicate areas of non-conformance.

For a top level package representing a datastore schema, servers MUST implement the package definition faithfully, including all mandatory features.

Package definitions MAY modify the schema for directly or hierarchically included packages through the use of different module revisions or module deviations.

5.3.1. Use of YANG semantic versioning

Using the YANG semantic versioning scheme for package version numbers and module revision labels can help with conformance. In the general case, clients should be able to determine the nature of changes between two package versions by comparing the version number.

This usually means that a client does not have to be restricted to working only with servers that advertise exactly the same version of a package in YANG library. Instead, reasonable clients should be able to interoperate with any server that supports a package version that is backwards compatible to version that the client is designed for, assuming that the client is designed to ignore operational values for unknown data nodes.

For example, a client coded to support 'foo' package at version 1.0.0 should interoperate with a server implementing 'foo' package at version 1.3.5, because the YANG semantic versioning rules require that package version 1.3.5 is backwards compatible to version 1.0.0.

This also has a relevance on servers that are capable of supporting version selection because they need not support every version of a YANG package to ensure good client compatibility. Choosing suitable

minor versions within each major version number should generally be sufficient, particular if they can avoid non-backwards-compatible patch level changes.

5.3.2. The relationship between packages and datastores

As defined by NMDA [RFC8342], each datastore has an associated datastore schema. Sections 5.1 and 5.3 of NMDA defines further constraints on the schema associated with datastores. These constraints can be summarized thus:

* The schema for all conventional datastores is the same.

* The schema for non conventional configuration datastores (e.g., dynamic datastores) may completely differ (i.e. no overlap at all) from the schema associated with the conventional configuration datastores, or may partially or fully overlap with the schema of the conventional configuration datastores. A dynamic datastore, for example, may support different modules than conventional datastores, or may support a subset or superset of modules, features, or data nodes supported in the conventional configuration datastores. Where a data node exists in multiple datastore schema it has the same type, properties and semantics.

* The schema for the operational datastore is intended to be a superset of all the configuration datastores (i.e. includes all the schema nodes from the conventional configuration datastores), but data nodes can be omitted if they cannot be accurately reported. The operational datastore schema can include additional modules containing only config false data nodes, but there is no harm in including those modules in the configuration datastore schema as well.

Given that YANG packages represent a schema, it follows that each datastore schema can be represented using packages. In addition, the schema for most datastores on a server are often closely related. Given that there are many ways that a datastore schema could be represented using packages, the following guidance provides a consistent approach to help clients understand the relationship between the different datastore schema supported by a device (e.g., which parts of the schema are common and which parts have differences):

* Any datastores (e.g., conventional configuration datastores) that have exactly the same datastore schema MUST use the same package definitions. This is to avoid, for example, the creation of a 'running-cfg' package and a separate 'intended-cfg' package that have identical schema.

* Common package definitions SHOULD be used for those parts of the datastore schema that are common between datastores, when those datastores do not share exactly the same datastore schema. E.g., if a substantial part of the schema is common between the conventional, dynamic, and operational datastores then a single common package can be used to describe the common parts, along with other packages to describe the unique parts of each datastore schema.

* YANG modules that do not contain any configuration data nodes SHOULD be included in the package for configuration datastores if that helps unify the package definitions.

* The packages for the operational datastore schema MUST include all packages for all configuration datastores, along with any required modules defining deviations to mark unsupported data nodes. The deviations MAY be defined directly in the packages defining the operational datastore schema, or in separate non referentially complete packages.

* The schema for a datastore MAY be represented using a single package or as the union of a set of compatible packages, i.e., equivalently to a set of non-conflicting packages being included together in an overarching package definition.

5.4. Schema referential completeness

A YANG package may represent a schema that is 'referentially complete', or 'referentially incomplete', indicated in the package definition by the 'complete' flag.

If all import statements in all YANG modules included in the package (either directly, or through included packages) can be resolved to a module revision defined with the YANG package definition, then the package is classified as referentially complete. Conversely, if one or more import statements cannot be resolved to a module specified as part of the package definition, then the package is classified as referentially incomplete.

A package that represents the exact contents of a datastore schema MUST always be referentially complete.

Referentially incomplete packages can be used, along with locally scoped packages, to represent an update to a device's datastore schema as part of an optional software hot fix. E.g., the base software is made available as a complete globally scoped package. The hot fix is made available as an incomplete globally scoped package. A device's datastore schema can define a local package that implements the base software package updated with the hot fix package.

Referentially incomplete packages could also be used to group sets of logically related modules together, but without requiring a fixed dependency on all imported 'types' modules (e.g., iana-if-types.yang), instead leaving the choice of specific revisions of 'types' modules to be resolved when the package definition is used.

5.5. Package name scoping and uniqueness

YANG package names can be globally unique, or locally scoped to a particular server or device.

5.5.1. Globally scoped packages

The name given to a package MUST be globally unique, and it MUST include an appropriate organization prefix in the name, equivalent to YANG module naming conventions.

Ideally a YANG instance data file defining a particular package version would be publicly available at one or more URLs.

5.5.2. Server scoped packages

Package definitions may be scoped to a particular server by setting the 'is-local' leaf to true in the package definition.

Locally scoped packages MAY have a package name that is not globally unique.

Locally scoped packages MAY have a definition that is not available offline from the server in a YANG instance data file.

5.6. Submodules packages considerations

As defined in [RFC7950] and [I-D.ietf-netmod-yang-semver], YANG conformance and versioning is specified in terms of particular revisions of YANG modules rather than for individual submodules.

However, YANG package definitions also include the list of submodules included by a module, primarily to provide a location of where the submodule definition can be obtained from, allowing a schema to be fully constructed from a YANG package instance data file definition.

5.7. Package tags

[I-D.ietf-netmod-module-tags] defines YANG module tags as a mechanism to annotate a module definition with additional metadata. Tags MAY also be associated to a package definition via the 'tags' leaf-list. The tags use the same registry and definitions used by YANG module tags.

5.8. YANG Package Usage Guidance

It is RECOMMENDED that organizations that publish YANG modules also publish YANG package definition that group and version those modules into units of related functionality. This increases interoperability, by encouraging implementations to use the same collections of YANG modules versions. Using packages also makes it easier to understand relationship between modules, and enables functionality to be described on a more abstract level than individual modules.

5.8.1. Use of deviations in YANG packages

[RFC7950] section 5.6.3 defines deviations as the mechanism to allow servers to indicate where they do not conform to a published YANG module that is being implemented.

In cases where implementations contain deviations from published packages, then those implementations SHOULD define a package that includes both the published packages and all modules containing deviations. This implementation specific package accurately reflects the schema used by the device and allows clients to determine how the implementation differs from the published package schema in an offline consumable way, e.g., when published in an instance data file (see section 6).

Organizations may wish to reuse YANG modules and YANG packages published by other organizations for new functionality. Sometimes, they may desire to modify the published YANG modules. However, they MUST NOT use deviations in an attempt to achieve this because such deviations cause two problems:

- They prevent implementations from reporting their own deviations for the same nodes.

They fracture the ecosystem by preventing implementations from conforming to the standards specified by both organizations. This hurts the interoperability in the YANG community, promotes development of disconnected functional silos, and hurts creativity in the market.

5.8.2. Use of features in YANG modules and YANG packages

The YANG language supports feature statements as the mechanism to make parts of a schema optional. Published standard YANG modules SHOULD make use of appropriate feature statements to provide flexibility in how YANG modules may be used by implementations and used by YANG modules published by other organizations.

YANG packages support 'mandatory features' which allow a package to specify features that MUST be implemented by any conformant implementation of the package as a mechanism to simplify and manage the schema represented by a YANG package.

5.9. YANG package core definition

The ietf-yang-package-types.yang module defines a grouping to specify the core elements of the YANG package structure that is used within YANG package instance data files (ietf-yang-package-instance.yang) and also on the server (ietf-yang-packages.yang).

The "ietf-yang-package-types" YANG module has the following structure:

```
module: ietf-yang-package-types

  grouping yang-pkg-identification-leafs
    +-- name       pkg-name
    +-- version    pkg-version

  grouping yang-pkg-instance
    +-- name                   pkg-name
    +-- version                pkg-version
    +-- timestamp?             yang:date-and-time
    +-- organization?          string
    +-- contact?               string
    +-- description?           string
    +-- reference?             string
    +-- complete?              boolean
    +-- local?                 boolean
    +-- tag*                   tags:tag
    +-- mandatory-feature*     scoped-feature
    +-- included-package* [name version]
    |  +-- name                 pkg-name
    |  +-- version              pkg-version
    |  +-- replaces-version*    pkg-version
    |  +-- location*            inet:uri
    +-- module* [name]
    |  +-- name                  yang:yang-identifier
    |  +-- revision?             rev:revision-date-or-label
    |  +-- replaces-revision*    rev:revision-date-or-label
    |  +-- namespace?            inet:uri
    |  +-- location*             inet:uri
    |  +-- submodule* [name]
    |     +-- name?       yang:yang-identifier
    |     +-- revision    yang:revision-identifier
    |     +-- location*   inet:uri
    +-- import-only-module* [name revision]
       +-- name?                yang:yang-identifier
       +-- revision?            rev:revision-date-or-label
       +-- replaces-revision*   rev:revision-date-or-label
       +-- namespace?           inet:uri
       +-- location*            inet:uri
       +-- submodule* [name]
          +-- name?       yang:yang-identifier
          +-- revision    yang:revision-identifier
          +-- location*   inet:uri
```

6. Package Instance Data Files

YANG packages SHOULD be available offline from the server, defined as YANG instance data files [I-D.ietf-netmod-yang-instance-file-format] using the schema below to define the package data.

The following rules apply to the format of the YANG package instance files:

1. The file SHOULD be encoded in JSON.

2. The name of the file SHOULD follow the format "<package-name>@<version>.json".

3. The package name MUST be specified in both the instance-data-set 'name' and package 'name' leafs.

4. The 'description' field of the instance-data-set SHOULD be "YANG package definition".

5. The 'timestamp', "organization', 'contact' fields are defined in both the instance-data-set metadata and the YANG package metadata. Package definitions SHOULD only define these fields as part of the package definition. If any of these fields are populated in the instance-data-set metadata then they MUST contain the same value as the corresponding leaves in the package definition.

6. The 'revision' list in the instance data file SHOULD NOT be used, since versioning is handled by the package definition.

7. The instance data file for each version of a YANG package SHOULD be made available at one of more locations accessible via URLs. If one of the listed locations defines a definitive reference implementation for the package definition then it MUST be listed as the first entry in the list.

The "ietf-yang-package" YANG module has the following structure:

```
module: ietf-yang-package

  structure package:
    // Uses the yang-package-instance grouping defined in
    // ietf-yang-package-types.yang
    +-- name                  pkg-name
    +-- version               pkg-version
    ... remainder of yang-package-instance grouping ...
```

7. Package Definitions on a Server

7.1. Package List

A top level 'packages' container holds the list of all versions of all packages known to the server. Each list entry uses the common package definition, but with the addition of package location that cannot be contained within a offline package definition contained in an instance data file.

The '/packages/package' list MAY include multiple versions of a particular package. E.g. if the server is capable of allowing clients to select which package versions should be used by the server.

7.2. Tree diagram

The "ietf-yang-packages" YANG module has the following structure:

```
module: ietf-yang-packages
  +--ro packages
     +--ro package* [name version]
        // Uses the yang-package-instance grouping defined in
        // ietf-yang-package-types.yang, with location:
        +--ro name                 pkg-name
        +--ro version              pkg-version
        ... remainder of yang-package-instance grouping ...
        +--ro location*            inet:uri
```

8. YANG Library Package Bindings

The YANG packages module also augments YANG library to allow a server to optionally indicate that a datastore schema is defined by a package, or a union of compatible packages. Since packages can generally be made available offline in instance data files, it may be sufficient for a client to only check that a compatible version of the package is implemented by the server without fetching either the package definition, or downloading and comparing the full list of modules and enabled features.

If a server indicates that a datastore schema maps to a particular package, then it MUST exactly match the schema defined by that package, taking into account enabled features and any deviations.

If a server cannot faithfully implement a package then it can define a new package to accurately report what it does implement. The new package can include the original package as an included package, and the new package can define additional modules containing deviations to the modules in the original package, allowing the new package to accurately describe the server's behavior. There is no specific mechanism provided to indicate that a mandatory-feature in package definition is not supported on a server, but deviations MAY be used to disable functionality predicated by an if-feature statement.

The "ietf-yl-packages" YANG module has the following structure:

```
module: ietf-yl-packages
  augment /yanglib:yang-library/yanglib:schema:
    +--ro package* [name version]
       +--ro name        -> /pkgs:packages/package/name
       +--ro version     leafref
```

9. YANG packages as schema for YANG instance data document

YANG package definitions can be used as the content schema definition for YANG instance data files. When using a package-based content schema, the name and version of the package MUST be specified, a package URL to the package definition MAY also be provided.

The "ietf-yang-inst-data-pkg" YANG module has the following structure:

```
module: ietf-yang-inst-data-pkg

  augment-structure /yid:instance-data-set/yid:content-schema/yid:content-schema-
spec:
    +--:(pkg-schema)
      +-- pkg-schema
        +-- name        pkg-name
        +-- version     pkg-version
        +-- location*   inet:uri
```

10. YANG Modules

The YANG module definitions for the modules described in the previous sections.

<CODE BEGINS>
file "ietf-yang-package-types#0.3.0-draft-ietf-netmod-yang-packages-03.yang"

```
module ietf-yang-package-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package-types";
  prefix pkg-types;

  import ietf-yang-revisions {
    prefix rev;
    reference
      "XXXX: Updated YANG Module Revision Handling";
  }
  import ietf-yang-types {
    prefix yang;
    rev:revision-or-derived "2019-07-21";
    reference
      "RFC 6991bis: Common YANG Data Types.";
  }
  import ietf-inet-types {
    prefix inet;
    rev:revision-or-derived "2013-07-15";
    reference
      "RFC 6991: Common YANG Data Types.";
  }
  import ietf-module-tags {
    prefix tags;
    reference
      "RFC 8819: YANG Module Tags.";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
```

```
        WG List:  <mailto:netmod@ietf.org>

        Author:   Rob Wilton
                  <mailto:rwilton@cisco.com>";
    description
      "This module provides type and grouping definitions for YANG
       packages.

       Copyright (c) 2022 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; see
       the RFC itself for full legal notices.

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

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

    revision 2022-03-04 {
      rev:revision-label 0.3.0-draft-ietf-netmod-yang-packages-03;
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Packages";
    }

    /*
     * Typedefs
     */

    typedef pkg-name {
      type yang:yang-identifier;
      description
        "Package names are typed as YANG identifiers.";
```

```
  }

  typedef pkg-version {
    type rev:revision-date-or-label;
    description
      "Package versions SHOULD be a revision-label (e.g. perhaps a
       YANG Semver version string).  Package versions MAY also be a
       revision-date";
  }

  typedef pkg-identifier {
    type rev:name-revision;
    description
      "Package identifiers combine a pkg-name and a pkg-version";
  }

  typedef scoped-feature {
    type string {
      pattern '[a-zA-Z_][a-zA-Z0-9\-_.]*:[a-zA-Z_][a-zA-Z0-9\-_.]*';
    }
    description
      "Represents a feature name scoped to a particular module,
       identified as the '<module-name>:<feature-name>', where both
       <module-name> and <feature-name> are YANG identifier strings,
       as defiend by Section 12 or RFC 6020.";
    reference
      "RFC XXXX, YANG Packages.";
  }

  /*
   * Groupings
   */

  grouping yang-pkg-identification-leafs {
    description
      "Parameters for identifying a specific version of a YANG
       package";
    leaf name {
      type pkg-name;
      mandatory true;
      description
        "The YANG package name.";
    }
    leaf version {
      type pkg-version;
      mandatory true;
      description
        "Uniquely identies a particular version of a YANG package.
```

```
            Follows the definition for revision labels defined in
            draft-verdt-nemod-yang-module-versioning, section XXX";
        }
      }

      grouping yang-pkg-instance {
        description
          "Specifies the data node for a full YANG package instance
           represented either on a server or as a YANG instance data
           document.";
        uses yang-pkg-identification-leafs;
        leaf timestamp {
          type yang:date-and-time;
          description
            "An optional timestamp for when this package was created.
             This does not need to be unique across all versions of a
             package.";
        }
        leaf organization {
          type string;
          description
            "Organization responsible for this package";
        }
        leaf contact {
          type string;
          description
            "Contact information for the person or organization to whom
             queries concerning this package should be sent.";
        }
        leaf description {
          type string;
          description
            "Provides a description of the package";
        }
        leaf reference {
          type string;
          description
            "Allows for a reference for the package";
        }
        leaf complete {
          type boolean;
          default "true";
          description
            "Indicates whether the schema defined by this package is
             referentially complete.  I.e. all module imports can be
             resolved to a module explicitly defined in this package or
             one of the included packages.";
        }
```

```
      leaf local {
        type boolean;
        default "false";
        description
          "Defines that the package definition is local to the server,
           and the name of the package MAY not be unique, and the
           package definition MAY not be available in an offline file.

           Local packages can be used when the schema for the device
           can be changed at runtime through the addition or removal of
           software packages, or hot fixes.";
      }
      leaf-list tag {
        type tags:tag;
        description
          "Tags associated with a YANG package.  Module tags defined in
           XXX, ietf-netmod-module-tags can be used here but with the
           modification that the tag applies to the entire package
           rather than a specific module.  See the IANA 'YANG Module
           Tag Prefix' registry for reserved prefixes and the IANA
           'YANG Module IETF Tag' registry for IETF standard tags.";
      }
      leaf-list mandatory-feature {
        type scoped-feature;
        description
          "Lists features from any modules included in the package that
           MUST be supported by any server implementing the package.

           Features already specified in a 'mandatory-feature' list of
           any included package MUST also be supported by server
           implementations and do not need to be repeated in this list.

           All other features defined in modules included in the
           package are OPTIONAL to implement.

           Features are identified using <module-name>:<feature-name>";
      }
      list included-package {
        key "name version";
        description
          "An entry in this list represents a package that is included
           as part of the package definition, or an indirectly included
           package that is changed in a non backwards compatible way.

           It can be used to resolve inclusion of conflicting package
           versions by explicitly specifying which package version is
           used.
```

```
            If included packages implement different revisions
            of the same module, then an explicit entry in the
            module list MUST be provided to select the specific module
            revision 'implemented' by this package definition.

            For import-only modules, the 'replaces-revision' leaf-list
            can be used to select the specific module revisions used by
            this package.";
          reference
            "XXX";
          uses yang-pkg-identification-leafs;
          leaf-list replaces-version {
            type pkg-version;
            description
              "Gives the version of an included package version that
               is replaced by this included package version.";
          }
          leaf-list location {
            type inet:uri;
            description
              "Contains a URL that represents where an instance data file
               for this YANG package can be found.

               This leaf will only be present if there is a URL available
               for retrieval of the schema for this entry.

               If multiple locations are provided, then the first
               location in the leaf-list MUST be the definitive location
               that uniquely identifies this package";
          }
        }
        list module {
          key "name";
          description
            "An entry in this list represents a module that must be
             implemented by a server implementing this package, as per
             RFC 7950 section 5.6.5, with a particular set of supported
             features and deviations.

             A entry in this list overrides any module revision
             'implemented' by an included package.  Any replaced module
             revision SHOULD also be listed in the 'replaces-revision'
             list.";
          reference
            "RFC 7950: The YANG 1.1 Data Modeling Language.";
          leaf name {
            type yang:yang-identifier;
            mandatory true;
```

```
          description
            "The YANG module name.";
        }
        leaf revision {
          type rev:revision-date-or-label;
          description
            "The YANG module revision date or revision-label.

             If no revision statement is present in the YANG module,
             this leaf is not instantiated.";
        }
        leaf-list replaces-revision {
          type rev:revision-date-or-label;
          description
            "Gives the revision of an module (implemented or
             import-only) defined in an included package that is
             replaced by this implemented module revision.";
        }
        leaf namespace {
          type inet:uri;
          description
            "The XML namespace identifier for this module.";
        }
        leaf-list location {
          type inet:uri;
          description
            "Contains a URL that represents the YANG schema resource
             for this module.

             This leaf will only be present if there is a URL available
             for retrieval of the schema for this entry.";
        }
        list submodule {
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";
          leaf name {
            type yang:yang-identifier;
            description
              "The YANG submodule name.";
          }
          leaf revision {
            type rev:revision-date-or-label;
            mandatory true;
            description
              "The YANG submodule revision date or revision-label.
```

```
                If the parent module include statement for this submodule
                includes a revision date then it MUST match the revision
                date specified here or it MUST match the revision-date
                associated with the revision-label specified here.";
            }
            leaf-list location {
              type inet:uri;
              description
                "Contains a URL that represents the YANG schema resource
                 for this submodule.

                 This leaf will only be present if there is a URL
                 available for retrieval of the schema for this entry.";
            }
          }
        }
        list import-only-module {
          key "name revision";
          description
            "An entry in this list indicates that the server imports
             reusable definitions from the specified revision of the
             module, but does not implement any protocol accessible
             objects from this revision.

             Multiple entries for the same module name MAY exist.  This
             can occur if multiple modules import the same module, but
             specify different revision-dates in the import statements.";
          leaf name {
            type yang:yang-identifier;
            description
              "The YANG module name.";
          }
          leaf revision {
            type rev:revision-date-or-label;
            description
              "The YANG module revision date or revision-label.

               If no revision statement is present in the YANG module,
               this leaf is not instantiated.";
          }
          leaf-list replaces-revision {
            type rev:revision-date-or-label;
            description
              "Gives the revision of an import-only-module defined in an
               included package that is replaced by this
               import-only-module revision.";
          }
          leaf namespace {
```

```
          type inet:uri;
          description
            "The XML namespace identifier for this module.";
        }
        leaf-list location {
          type inet:uri;
          description
            "Contains a URL that represents the YANG schema resource
             for this module.

             This leaf will only be present if there is a URL available
             for retrieval of the schema for this entry.";
        }
        list submodule {
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";
          leaf name {
            type yang:yang-identifier;
            description
              "The YANG submodule name.";
          }
          leaf revision {
            type yang:revision-identifier;
            mandatory true;
            description
              "The YANG submodule revision date.  If the parent module
               include statement for this submodule includes a revision
               date then it MUST match this leaf's value.";
          }
          leaf-list location {
            type inet:uri;
            description
              "Contains a URL that represents the YANG schema resource
               for this submodule.

               This leaf will only be present if there is a URL
               available for retrieval of the schema for this entry.";
          }
        }
      }
    }
  }
  <CODE ENDS>
```

```
   <CODE BEGINS>
     file "ietf-yang-package-instance#0.3.0-draft-ietf-netmod-yang-packages-03.ya
ng"
   module ietf-yang-package-instance {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package-instance";
     prefix pkg-inst;

     import ietf-yang-revisions {
       prefix rev;
       reference
         "XXXX: Updated YANG Module Revision Handling";
     }
     import ietf-yang-package-types {
       prefix pkg-types;
       rev:revision-or-derived "0.2.0";
       reference
         "RFC XXX: this RFC.";
     }
     import ietf-yang-structure-ext {
       prefix sx;
       reference
         "RFC 8791: YANG Data Structure Extensions.";
     }

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

        Author:   Rob Wilton
                  <mailto:rwilton@cisco.com>";
     description
       "This module provides a definition of a YANG package, which is
        used as the content schema for an YANG instance data document specifying
        a YANG package.

        Copyright (c) 2022 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; see
```

```
        the RFC itself for full legal notices.

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

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

      revision 2022-03-04 {
        rev:revision-label 0.3.0-draft-ietf-netmod-yang-packages-03;
        description
          "Initial revision";
        reference
          "RFC XXXX: YANG Packages";
      }

      /*
       * Top-level structure
       */
      sx:structure "package" {
        description
          "Defines the YANG package structure for use in a YANG instance
           data document.";
        uses pkg-types:yang-pkg-instance;
      }
    }
    <CODE ENDS>

    <CODE BEGINS>
      file "ietf-yang-packages#0.3.0-draft-ietf-netmod-yang-packages-03.yang"
    module ietf-yang-packages {
      yang-version 1.1;
      namespace "urn:ietf:params:xml:ns:yang:ietf-yang-packages";
      prefix pkgs;

      import ietf-yang-revisions {
        prefix rev;
        reference
          "XXXX: Updated YANG Module Revision Handling";
      }
      import ietf-yang-package-types {
        prefix pkg-types;
        rev:revision-or-derived "0.2.0";
```

```
      reference
        "RFC XXX: this RFC.";
    }
    import ietf-inet-types {
      prefix inet;
      rev:revision-or-derived "2013-07-15";
      reference
        "RFC 6991: Common YANG Data Types.";
    }

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

       Author:   Rob Wilton
                 <mailto:rwilton@cisco.com>";
    description
      "This module defines YANG packages on a server implementation.

       Copyright (c) 2022 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; see
       the RFC itself for full legal notices.

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

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

    revision 2022-03-04 {
      rev:revision-label 0.3.0-draft-ietf-netmod-yang-packages-03;
      description
```

```
        "Initial revision";
      reference
        "RFC XXXX: YANG Packages";
    }

    /*
     * Groupings
     */

    grouping yang-pkg-ref {
      description
        "Defines the leaves used to reference a single YANG package";
      leaf name {
        type leafref {
          path "/pkgs:packages/pkgs:package/pkgs:name";
        }
        description
          "The name of the references package.";
      }
      leaf version {
        type leafref {
          path "/pkgs:packages"
             + '/pkgs:package[pkgs:name = current()/../name]'
             + "/pkgs:version";
        }
        description
          "The version of the referenced package.";
      }
    }

    grouping yang-ds-pkg-ref {
      description
        "Defines the list used to reference a set of YANG packages that
         collectively represent a datastore schema.";
      list package {
        key "name version";
        description
          "Identifies the YANG packages that collectively defines the
           schema for the associated datastore.

           The datastore schema is defined as the union of all
           referenced packages, that MUST represent a referentially
           complete schema.

           All of the referenced packages must be compatible with no
           conflicting module versions or dependencies.";
        uses yang-pkg-ref;
      }
```

```
    }

    /*
     * Top level data nodes.
     */

    container packages {
      config false;
      description
        "All YANG package definitions";
      list package {
        key "name version";
        description
          "YANG package instance";
        uses pkg-types:yang-pkg-instance;
        leaf-list location {
          type inet:uri;
          description
            "Contains a URL that represents where an instance data file
             for this YANG package can be found.

             This leaf will only be present if there is a URL available
             for retrieval of the schema for this entry.

             If multiple locations are provided, then the first
             location in the leaf-list MUST be the definitive location
             that uniquely identifies this package";
        }
      }
    }
  }
  <CODE ENDS>

  <CODE BEGINS>
    file "ietf-yl-package#0.3.0-draft-ietf-netmod-yang-packages-03.yang"
  module ietf-yl-packages {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yl-packages";
    prefix yl-pkgs;

    import ietf-yang-revisions {
      prefix rev;
      reference
        "XXXX: Updated YANG Module Revision Handling";
    }
    import ietf-yang-packages {
      prefix pkgs;
      rev:revision-or-derived "0.2.0";
```

```
      reference
        "RFC XXX: YANG Packages.";
    }
    import ietf-yang-library {
      prefix yanglib;
      rev:revision-or-derived "2019-01-04";
      reference
        "RFC 8525: YANG Library";
    }

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

       Author:   Rob Wilton
                 <mailto:rwilton@cisco.com>";
    description
      "This module provides defined augmentations to YANG library to
       allow a server to report YANG package information.

       Copyright (c) 2022 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; see
       the RFC itself for full legal notices.

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

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

    revision 2022-03-04 {
```

```
      rev:revision-label 0.3.0-draft-ietf-netmod-yang-packages-03;
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Packages";
    }

    /*
     * Augmentations
     */

    augment "/yanglib:yang-library/yanglib:schema" {
      description
        "Allow datastore schema to be related to a set of YANG
         packages";
      uses pkgs:yang-ds-pkg-ref;
    }
  }
  <CODE ENDS>

  <CODE BEGINS>
    file "ietf-yang-inst-data-pkg#0.3.0-draft-ietf-netmod-yang-packages-03.yang"
  module ietf-yang-inst-data-pkg {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-inst-data-pkg";
    prefix yid-pkg;

    import ietf-yang-revisions {
      prefix rev;
      reference
        "XXXX: Updated YANG Module Revision Handling";
    }
    import ietf-yang-package-types {
      prefix pkg-types;
      rev:revision-or-derived "0.2.0";
      reference
        "RFC XXX: this RFC.";
    }
    import ietf-yang-structure-ext {
      prefix sx;
      reference
        "RFC 8791: YANG Data Structure Extensions.";
    }
    import ietf-yang-instance-data {
      prefix yid;
      reference
        "RFC 9195: A File Format for YANG Instance Data.";
    }
```

```
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types.";
  }

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

     Author:   Rob Wilton
               <mailto:rwilton@cisco.com>";
  description
    "The module augments ietf-yang-instance-data to allow package
     definitions to be used to define content schema in YANG instance data
     documents.

     Copyright (c) 2022 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; see
     the RFC itself for full legal notices.

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

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

  revision 2022-03-04 {
    rev:revision-label 0.3.0-draft-ietf-netmod-yang-packages-03;
    description
      "Initial revision";
    reference
```

```
         "RFC XXXX: YANG Packages";
     }

     /*
      * Augmentations
      */
     sx:augment-structure "/yid:instance-data-set/yid:content-schema/yid:content-
schema-spec" {
       description
         "Add package reference to instance data set schema
          specification";
       case pkg-schema {
         container pkg-schema {
           uses pkg-types:yang-pkg-identification-leafs;
           leaf-list location {
             type inet:uri;
             description
               "Contains a URL that represents where an instance data
                file for this YANG package can be found.

                This leaf will only be present if there is a URL
                available for retrieval of the schema for this entry.

                If multiple locations are provided, then the first
                location in the leaf-list MUST be the definitive
                location that uniquely identifies this package";
           }
         }
       }
     }
   }
   <CODE ENDS>
```

11. Security Considerations

The YANG modules specified in this document defines a schema for data that is accessed by network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

Similarly to YANG library [I-D.ietf-netconf-rfc7895bis], some of the readable data nodes in these YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes.

One additional key different to YANG library, is that the 'ietf-yang-package' YANG module defines a schema to allow YANG packages to be defined in YANG instance data files, that are outside the security controls of the network management protocols. Hence, it is important to also consider controlling access to these package instance data files to restrict access to sensitive information.

As per the YANG library security considerations, the module, revision information in YANG packages may help an attacker identify the server capabilities and server implementations with known bugs since the set of YANG modules supported by a server may reveal the kind of device and the manufacturer of the device. Server vulnerabilities may be specific to particular modules, module revisions, module features, or even module deviations. For example, if a particular operation on a particular data node is known to cause a server to crash or significantly degrade device performance, then the YANG packages information will help an attacker identify server implementations with such a defect, in order to launch a denial-of-service attack on the device.

12. IANA Considerations

It is expected that a central registry of standard YANG package definitions is required to support this solution.

It is unclear whether an IANA registry is also required to manage specific package versions. It is highly desirable to have a specific canonical location, under IETF control, where the definitive YANG package versions can be obtained from.

This document requests IANA to registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations are requested.

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

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

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

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

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

This document requests that the following YANG modules are added in the "YANG Module Names" registry [RFC6020]:

Name: ietf-yang-package-types.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-package-types.yang
Prefix: pkg-types
Reference: RFC XXXX

Name: ietf-yang-package-instance.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-package-instance.yang
Prefix: pkg-inst
Reference: RFC XXXX

Name: ietf-yang-packages.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-packages.yang
Prefix: pkgs
Reference: RFC XXXX

Name: ietf-yl-packages.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yl-packages.yang
Prefix: yl-pkgs
Reference: RFC XXXX

Name: ietf-yang-inst-data-pkg.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-inst-data-pkg.yang
Prefix: yid-pkg
Reference: RFC XXXX

13. Open Questions/Issues

All issues, along with the draft text, are currently being tracked at https://github.com/rgwilton/YANG-Packages-Draft/issues/

14. Acknowledgements

Feedback helping shape this document has kindly been provided by Andy Bierman, James Cumming, Mahesh Jethanandani, Balazs Lengyel, Ladislav Lhotka,and Jan Lindblad.

15. References

15.1. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", Work in Progress, Internet-Draft, draft-ietf-netconf-rfc7895bis-07, 17 October 2018, <https://www.ietf.org/archive/id/draft-ietf-netconf-rfc7895bis-07.txt>.

[I-D.ietf-netmod-module-tags]
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", Work in Progress, Internet-Draft, draft-ietf-netmod-module-tags-10, 29 February 2020, <https://www.ietf.org/archive/id/draft-ietf-netmod-module-tags-10.txt>.

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Structure Extensions", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-data-ext-05, 9 December 2019, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-data-ext-05.txt>.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "A File Format for YANG Instance Data", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-instance-file-format-21, 8 October 2021, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-instance-file-format-21.txt>.

[I-D.ietf-netmod-yang-module-versioning]
Wilton, R., Rahman, R., Lengyel, B., Clarke, J., and J. Sterne, "Updated YANG Module Revision Handling", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-05, 8 November 2021, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-module-versioning-05.txt>.

[I-D.ietf-netmod-yang-semver]
Clarke, J., Wilton, R., Rahman, R., Lengyel, B., Sterne, J., and B. Claise, "YANG Semantic Versioning", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-semver-06, 30 November 2021, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-semver-06.txt>.

[I-D.ietf-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-solutions-01, 2 November 2020, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-solutions-01.txt>.

[I-D.ietf-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Schema Selection", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-ver-selection-00, 17 March 2020, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-ver-selection-00.txt>.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-versioning-reqs-06, 6 January 2022, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-versioning-reqs-06.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

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

15.2. Informative References

[I-D.bierman-netmod-yang-package]
Bierman, A., "The YANG Package Statement", Work in Progress, Internet-Draft, draft-bierman-netmod-yang-package-00, 6 July 2015, <https://www.ietf.org/archive/id/draft-bierman-netmod-yang-package-00.txt>.

[I-D.ietf-netmod-artwork-folding]
Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", Work in Progress, Internet-Draft, draft-ietf-netmod-artwork-folding-12, 20 January 2020, <https://www.ietf.org/archive/id/draft-ietf-netmod-artwork-folding-12.txt>.

[openconfigsemver]
"Semantic Versioning for OpenConfig Models", <http://www.openconfig.net/docs/semver/>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

Appendix A. Examples

This section provides various examples of YANG packages, and as such this text is non-normative. The purpose of the examples is to only illustrate the file format of YANG packages, and how package dependencies work. It does not imply that such packages will be defined by IETF, or which modules would be included in those packages even if they were defined. For brevity, the examples exclude namespace declarations, and use a shortened URL of "tiny.cc/ietf-yang" as a replacement for "https://raw.githubusercontent.com/YangModels/yang/master/standard/ietf/RFC".

A.1. Example IETF Network Device YANG package

This section provides an instance data file example of an IETF Network Device YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, to implement a basic network device without any dynamic routing or layer 2 services. E.g., it includes functionality such as system information, interface and basic IP configuration.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, the modules are referenced by revision date rather than revision number.

```
   <CODE BEGINS> file "example-ietf-network-device-pkg.json"
   ========= NOTE: '\' line wrapping per BCP XX (RFC XXXX) ===========

   {
     "ietf-yang-instance-data:instance-data-set": {
       "name": "example-ietf-network-device-pkg",
       "content-schema": {
         "pkg-schema": {
            "name": "ietf-yang-package-defn-pkg",
            "version": "0.1.0"
         }
       },
       "description": "YANG package definition",
       "content-data": {
         "ietf-yang-package-instance:yang-package": {
           "name": "example-ietf-network-device-pkg",
           "version": "1.1.2",
           "timestamp": "2018-12-13T17:00:00Z",
           "organization": "IETF NETMOD Working Group",
           "contact" : "WG Web:   <http://tools.ietf.org/wg/netmod/>, \
                        WG List:  <mailto:netmod@ietf.org>",
           "description": "Example IETF network device YANG package.\
              \
              This package defines a small sample set of \
              YANG modules that could represent the basic set of \
              modules that a standard network device might be expected \
              to support.",
           "reference": "XXX, draft-rwilton-netmod-yang-packages",
           "location": [ "file://example.org/yang/packages/\
                                    ietf-network-device@v1.1.2.json" ],
           "module": [
             {
               "name": "iana-crypt-hash",
               "revision": "2014-08-06",
               "location": [ "https://tiny.cc/ietf-yang/\
                                iana-crypt-hash%402014-08-06.yang" ],
             },
             {
               "name": "ietf-system",
               "revision": "2014-08-06",
               "location": [ "https://tiny.cc/ietf-yang/\
                                  ietf-system%402014-08-06.yang" ],
             },
             {
               "name": "ietf-interfaces",
               "revision": "2018-02-20",
               "location": [ "https://tiny.cc/ietf-yang/\
                                ietf-interfaces%402018-02-20.yang" ],
```

```
            },
            {
              "name": "ietf-netconf-acm",
              "revision": "2018-02-14",
              "location": [ "https://tiny.cc/ietf-yang/\
                                ietf-netconf-acm%402018-02-14.yang" ],
            },
            {
              "name": "ietf-key-chain",
              "revision": "2017-06-15",
              "location": [ "https://tiny.cc/ietf-yang/\
                                   ietf-key-chain@2017-06-15.yang" ],
            },
            {
              "name": "ietf-ip",
              "revision": "2018-02-22",
              "location": [ "https://tiny.cc/ietf-yang/\
                                        ietf-ip%402018-02-22.yang" ],
          }
          ],
          "import-only-module": [
            {
              "name": "ietf-yang-types",
              "revision": "2013-07-15",
              "location": [ "https://tiny.cc/ietf-yang/\
                                ietf-yang-types%402013-07-15.yang" ],
            },
            {
              "name": "ietf-inet-types",
              "revision": "2013-07-15",
              "location": [ "https://tiny.cc/ietf-yang/\
                                ietf-inet-types%402013-07-15.yang" ],
            }
          ]
        }
      }
    }
  }
  <CODE ENDS>
```

A.2. Example IETF Basic Routing YANG package

This section provides an instance data file example of a basic IETF Routing YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, that builds upon the example-ietf-network-device YANG package to add support for basic dynamic routing and ACLs.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, they modules are referenced by revision date rather than revision number. Locations have been excluded where they are not currently known, e.g., for YANG modules defined in IETF drafts. In a normal YANG package, locations would be expected to be provided for all YANG modules.

<CODE BEGINS> file "example-ietf-routing-pkg.json"
========== NOTE: '\' line wrapping per BCP XX (RFC XXXX) ==========

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ietf-routing-pkg",
    "content-schema": {
      "pkg-schema": {
         "name": "ietf-yang-package-defn-pkg",
         "version": "0.1.0"
      }
    },
    "description": "YANG package definition",
    "content-data": {
      "ietf-yang-package-instance:yang-package": {
        "name": "example-ietf-routing",
        "version": "1.3.1",
        "timestamp": "2018-12-13T17:00:00Z",
        "description": "This package defines a small sample set of \
          IETF routing YANG modules that could represent the set of \
          IETF routing functionality that a basic IP network device \
          might be expected to support.",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "imported-packages": [
          {
            "name": "ietf-network-device",
            "version": "1.1.2",
            "location": [ "http://example.org/yang/packages/\
                                  ietf-network-device@v1.1.2.json" ],
          }
        ],
        "module": [
          {
            "name": "ietf-routing",
            "revision": "2018-03-13",
```

```
      "location": [ "https://tiny.cc/ietf-yang/\
                                 ietf-routing@2018-03-13.yang" ],
    },
    {
      "name": "ietf-ipv4-unicast-routing",
      "revision": "2018-03-13",
      "location": [ "https://tiny.cc/ietf-yang/\
                  ietf-ipv4-unicast-routing@2018-03-13.yang" ],
    },
    {
      "name": "ietf-ipv6-unicast-routing",
      "revision": "2018-03-13",
      "location": [ "https://tiny.cc/ietf-yang/\
                  ietf-ipv6-unicast-routing@2018-03-13.yang" ],
    },
    {
      "name": "ietf-isis",
      "revision": "2018-12-11",
      "location": [ "https://tiny.cc/ietf-yang/\
                  " ],
    },
    {
      "name": "ietf-interfaces-common",
      "revision": "2018-07-02",
      "location": [ "https://tiny.cc/ietf-yang/\
                  " ],
    },
    {
      "name": "ietf-if-l3-vlan",
      "revision": "2017-10-30",
      "location": [ "https://tiny.cc/ietf-yang/\
                  " ],
    },
    {
      "name": "ietf-routing-policy",
      "revision": "2018-10-19",
      "location": [ "https://tiny.cc/ietf-yang/\
                  " ],
    },
    {
      "name": "ietf-bgp",
      "revision": "2018-05-09",
      "location": [ "https://tiny.cc/ietf-yang/\
                  " ],
    },
    {
      "name": "ietf-access-control-list",
      "revision": "2018-11-06",
```

```
              "location": [ "https://tiny.cc/ietf-yang/\
                          " ],
            }
          ],
          "import-only-module": [
            {
              "name": "ietf-routing-types",
              "revision": "2017-12-04",
              "location": [ "https://tiny.cc/ietf-yang/\
                          ietf-routing-types@2017-12-04.yang" ],
            },
            {
              "name": "iana-routing-types",
              "revision": "2017-12-04",
              "location": [ "https://tiny.cc/ietf-yang/\
                          iana-routing-types@2017-12-04.yang" ],
            },
            {
              "name": "ietf-bgp-types",
              "revision": "2018-05-09",
              "location": [ "https://tiny.cc/ietf-yang/\
                          " ],
            },
            {
              "name": "ietf-packet-fields",
              "revision": "2018-11-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                          " ],
            },
            {
              "name": "ietf-ethertypes",
              "revision": "2018-11-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                          " ],
            }
          ]
        }
      }
    }
  }
  <CODE ENDS>
```

A.3. Package import conflict resolution example

This section provides an example of how a package can resolve conflicting module revisions from imported packages.

In this example, YANG package 'example-3-pkg' imports both 'example-import-1' and 'example-import-2' packages. However, the two imported packages implement different revisions of 'example-module-A' so the 'example-3-pkg' package selects version '1.2.3' to resolve the conflict. Similarly, for import-only modules, the 'example-3-pkg' package does not require both revisions of example-types-module-C to be imported, so it indicates that it only imports revision '2018-11-26' and not '2018-01-01'.

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-import-1-pkg",
    "content-schema": {
      "pkg-schema": {
         "name": "ietf-yang-package-defn-pkg",
         "version": "0.1.0"
      }
    },
    "description": "First imported example package",
    "content-data": {
      "ietf-yang-package-instance:yang-package": {
        "name": "example-import-1",
        "version": "1.0.0",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-01-01",
        "module": [
          {
            "name": "example-module-A",
            "revision": "1.0.0"
          },
          {
            "name": "example-module-B",
            "revision": "1.0.0"
          }
        ],
        "import-only-module": [
          {
            "name": "example-types-module-C",
            "revision": "2018-01-01"
          },
          {
            "name": "example-types-module-D",
            "revision": "2018-01-01"
          }
        ]
      }
    }
```

```
    }
  }

  {
    "ietf-yang-instance-data:instance-data-set": {
      "name": "example-import-2-pkg",
      "content-schema": {
        "pkg-schema": {
           "name": "ietf-yang-package-defn-pkg",
           "version": "0.1.0"
        }
      },
      "description": "Second imported example package",
      "content-data": {
        "ietf-yang-package:yang-package": {
          "name": "example-import-2",
          "version": "2.0.0",
          "reference": "XXX, draft-rwilton-netmod-yang-packages",
          "revision-date": "2018-11-26",
          "module": [
            {
              "name": "example-module-A",
              "revision": "1.2.3"
            },
            {
              "name": "example-module-E",
              "revision": "1.1.0"
            }
          ],
          "import-only-module": [
            {
              "name": "example-types-module-C",
              "revision": "2018-11-26"
            },
            {
              "name": "example-types-module-D",
              "revision": "2018-11-26"
            }
          ]
        }
      }
    }
  }

  {
    "ietf-yang-instance-data:instance-data-set": {
      "name": "example-3-pkg",
      "content-schema": {
```

```
        "pkg-schema": {
           "name": "ietf-yang-package-defn-pkg",
           "version": "0.1.0"
        }
      },
      "description": "Importing example package",
      "content-data": {
        "ietf-yang-package:yang-package": {
          "name": "example-3",
          "version": "1.0.0",
          "reference": "XXX, draft-rwilton-netmod-yang-packages",
          "revision-date": "2018-11-26",
          "included-package": [
            {
              "name": "example-import-1",
              "version": "1.0.0"
            },
            {
              "name": "example-import-2",
              "version": "2.0.0"
            }
          ],
          "module": [
            {
              "name": "example-module-A",
              "revision": "1.2.3"
            }
          ],
          "import-only-module": [
            {
              "name": "example-types-module-C",
              "revision": "2018-11-26",
              "replaces-revision": [ "2018-01-01 "]
            }
          ]
        }
      }
    }
  }
```

Appendix B. Possible alternative solutions

This section briefly describes some alternative solutions. It can be removed if this document is adopted as a WG draft.

B.1. Using module tags

Module tags have been suggested as an alternative solution, and indeed that can address some of the same requirements as YANG packages but not all of them.

Module tags can be used to group or organize YANG modules. However, this raises the question of where this tag information is stored. Module tags either require that the YANG module files themselves are updated with the module tag information (creating another versioning problem), or for the module tag information to be hosted elsewhere, perhaps in a centralize YANG Catalog, or in instance data files similar to how YANG packages have been defined in this draft.

One of the principle aims of YANG packages is to be a versioned object that defines a precise set of YANG modules versions that work together. Module tags cannot meet this aim without an explosion of module tags definitions (i.e. a separate module tag must be defined for each package version).

Module tags cannot support the hierachical scheme to construct schema that is proposed in this draft.

B.2. Using YANG library

Another question is whether it is necessary to define new YANG modules to define YANG packages, and whether YANG library could just be reused in an instance data file. The use of YANG packages offers several benefits over just using YANG library:

1. Packages allow schema to be built in a hierarchical fashion. [I-D.ietf-netconf-rfc7895bis] only allows one layer of hierarchy (using module sets), and there must be no conflicts between module revisions in different module-sets.

2. Packages can be made available off the box, with a well defined unique name, avoiding the need for clients to download, and construct/check the entire schema for each datastore. YANG library's use of a 'content-id' is unique only to the device that generated them.

3. Packages may be versioned using a semantic versioning scheme, YANG library does not provide a schema level semantic version number.

4. For a YANG library instance data file to contain the necessary information, it probably needs both YANG library and various augmentations (e.g. to include each module's semantic version

number), unless a new version of YANG library is defined containing this information. The module definition for a YANG package is specified to contain all of the ncessary information to solve the problem without augmentations

5. YANG library is designed to publish information about the modules, datastores, and datastore schema used by a server. The information required to construct an off box schema is not precisely the same, and hence the definitions might deviate from each other over time.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems, Inc.
Email: rwilton@cisco.com

Reshad Rahman
Cisco Systems, Inc.
Email: rrahman@cisco.com

Joe Clarke
Cisco Systems, Inc.
Email: jclarke@cisco.com

Jason Sterne
Nokia
Email: jason.sterne@nokia.com

Bo Wu (editor)
Huawei
Email: lana.wubo@huawei.com

Network Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: 3 June 2022

J. Clarke, Ed.
R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman

B. Lengyel
Ericsson
J. Sterne
Nokia
B. Claise
Huawei

30 November 2021

YANG Semantic Versioning
draft-ietf-netmod-yang-semver-06

Abstract

This document specifies a scheme and guidelines for applying an extended set of semantic versioning rules to revisions of YANG artifacts (e.g., modules and packages). Additionally, this document defines an RFCAAAA-compliant revision-label-scheme for this YANG semantic versioning scheme.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 June 2022.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-netmod-yang-module-versioning] puts forth a number of concepts relating to modified rules for updating modules and submodules, a means to signal when a new revision of a module or submodule has non-backwards-compatible (NBC) changes compared to its previous revision, and a scheme that uses the revision history as a lineage for determining from where a specific revision of a YANG module or submodule is derived. Additionally, section 3.4 of

[I-D.ietf-netmod-yang-module-versioning] defines a revision-label which can be used as an alias to provide additional context or as a meaningful label to refer to a specific revision.

This document defines a revision-label scheme that uses extended [semver] rules for YANG artifacts (i.e., YANG modules, YANG submodules, and YANG packages [I-D.ietf-netmod-yang-packages]) as well as the revision label definition for using this scheme. The goal being to add a human readable revision label that provides compatibility information for the YANG artifact without needing to compare or parse its body. The label and rules defined herein represent the RECOMMENDED revision label scheme for IETF YANG artifacts.

Note that a specific revision of the Semver 2.0.0 specification is referenced here (from June 19, 2020) to provide an immutable version. This is because the 2.0.0 version of the specification has changed over time without any change to the semantic version itself. In some cases the text has changed in non-backwards-compatible ways.

2. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Additionally, this document uses the following terminology:

* YANG artifact: YANG modules, YANG submodules, and YANG packages [I-D.ietf-netmod-yang-packages] are examples of YANG artifacts for the purposes of this document.

* YANG Semver: A revision-label identifier that is consistent with the extended set of semantic versioning rules, based on [semver] , defined within this document.

3. YANG Semantic Versioning

This section defines YANG Semantic Versioning, explains how it is used with YANG artifacts, and describes the rules associated with changing an artifact's semantic version when its contents are updated.

3.1. YANG Semver Pattern

YANG artifacts that employ semantic versioning as defined in this document MUST use a version string (e.g., in revision-label or as a package version) that corresponds to the following pattern: 'X.Y.Z_COMPAT'. Where:

* X, Y and Z are mandatory non-negative integers that are each less than or equal to 2147483647 (i.e., the maximum signed 32-bit integer value) and MUST NOT contain leading zeroes,

* The '.' is a literal period (ASCII character 0x2e),

* The '_' is an optional single literal underscore (ASCII character 0x5f) and MUST only be present if the following COMPAT element is included,

* COMPAT, if specified, MUST be either the literal string "compatible" or the literal string "non_compatible".

Additionally, [semver] defines two specific types of metadata that may be appended to a semantic version string. Pre-release metadata MAY be appended to a semver string after a trailing '-' character. Build metadata MAY be appended after a trailing '+' character. If both pre-release and build metadata are present, then build metadata MUST follow pre-release metadata. While build metadata MUST be ignored when comparing YANG semantic versions, pre-release metadata MUST be used during module and submodule development as specified in Section 5 . Both pre-release and build metadata are allowed in order to support all the [semver] rules. Thus, a version lineage that follows strict [semver] rules is allowed for a YANG artifact.

To signal the use of this versioning scheme, modules and submodules MUST set the revision-label-scheme extension, as defined in [I-D.ietf-netmod-yang-module-versioning] , to the identity "yang-semver". That identity value is defined in the ietf-yang-semver module below.

Additionally, this ietf-yang-semver module defines a typedef that formally specifies the syntax of the YANG Semver.

3.2. Semantic Versioning Scheme for YANG Artifacts

This document defines the YANG semantic versioning scheme that is used for YANG artifacts that employ the YANG Semver label. The versioning scheme has the following properties:

* The YANG semantic versioning scheme is extended from version 2.0.0 of the semantic versioning scheme defined at semver.org [semver] to cover the additional requirements for the management of YANG artifact lifecyles that cannot be addressed using the semver.org 2.0.0 versioning scheme alone.

* Unlike the [semver] versioning scheme, the YANG semantic versioning scheme supports updates to older versions of YANG artifacts, to allow for bug fixes and enhancements to artifact versions that are not the latest. However, it does not provide for the unlimited branching and updating of older revisions which are documented by the general rules in [I-D.ietf-netmod-yang-module-versioning] .

* YANG artifacts that follow the [semver] versioning scheme are fully compatible with implementations that understand the YANG semantic versioning scheme defined in this document.

* If updates are always restricted to the latest revision of the artifact only, then the version numbers used by the YANG semantic versioning scheme are exactly the same as those defined by the [semver] versioning scheme.

Every YANG module and submodule versioned using the YANG semantic versioning scheme specifies the module's or submodule's semantic version as the argument to the 'rev:revision-label' statement.

Because the rules put forth in [I-D.ietf-netmod-yang-module-versioning] are designed to work well with existing versions of YANG and allow for artifact authors to migrate to this scheme, it is not expected that all revisions of a given YANG artifact will have a semantic version label. For example, the first revision of a module or submodule may have been produced before this scheme was available.

YANG packages that make use of this YANG Semver will reflect that in the package metadata.

As stated above, the YANG semantic version is expressed as a string of the form: 'X.Y.Z_COMPAT'.

* 'X' is the MAJOR version. Changes in the MAJOR version number indicate changes that are non-backwards-compatible to versions with a lower MAJOR version number.

* 'Y' is the MINOR version. Changes in the MINOR version number indicate changes that are backwards-compatible to versions with the same MAJOR version number, but a lower MINOR version number and no PATCH "_compatible" or "_non_compatible" modifier.

* 'Z_COMPAT' is the PATCH version and modifier. Changes in the PATCH version number can indicate editorial, backwards-compatible, or non-backwards-compatible changes relative to versions with the same MAJOR and MINOR version numbers, but lower PATCH version number, depending on what form modifier '_COMPAT' takes:

  - If the modifier string is absent, the change represents an editorial change. An editorial change is defined to be a change in the YANG artifact's content that does not affect the semantic meaning or functionality provided by the artifact in any way. Some examples include correcting a spelling mistake in the description of a leaf within a YANG module or submodule, non-significant whitespace changes (e.g., realigning description statements or changing indendation), or changes to YANG comments. Note: restructuring how a module uses, or does not use, submodules is treated as an editorial level change on the condition that there is no change in the module's semantic behavior due to the restructuring.

  - If, however, the modifier string is present, the meaning is described below:

  - "_compatible" - the change represents a backwards-compatible change

  - "_non_compatible" - the change represents a non-backwards-compatible change

The '_COMPAT' modifier string is "sticky". Once a revision of a module has a modifier in the revision label, then all descendants of that revision with the same X.Y version digits will also have a modifier. The modifier can change from "_compatible" to "_non_compatible" in a descendant revision, but the modifier MUST NOT change from "_non_compatible" to "_compatible" and MUST NOT be removed. The persistence of the "_non_compatible" modifier ensures that comparisions of revision labels do not give the false impression of compatibility between two potentially non-compatible revisions. If "_non_compatible" was removed, for example between revisions "3.3.2_non_compatible" and "3.3.3" (where "3.3.3" was simply an editorial change), then comparing revision labels of "3.3.3" back to an ancestor "3.0.0" would look like they are backwards compatible when they are not (since "3.3.2_non_compatible" was in the chain of ancestors and introduced a non-backwards-compatible change).

The YANG artifact name and YANG semantic version uniquely identify a revision of said artifact. There MUST NOT be multiple instances of a YANG artifact definition with the same name and YANG semantic version but different content (and in the case of modules and submodules, different revision dates).

There MUST NOT be multiple versions of a YANG artifact that have the same MAJOR, MINOR and PATCH version numbers, but different patch modifier strings. E.g., artifact version "1.2.3_non_compatible" MUST NOT be defined if artifact version "1.2.3" has already been defined.

3.2.1. YANG Semver with submodules

YANG Semver MAY be used to version submodules. Submodule version are separate of any version on the including module, but if a submodule has changed, then the version of the including module MUST also be updated.

The rules for determining the version change of a submodule are the same as those defined in Section 3.1 and Section 3.2 as applied to YANG modules, except they only apply to the part of the module schema defined within the submodule's file.

One interesting case is moving definitions from one submodule to another in a way that does not change the resultant schema of the including module. In this case:

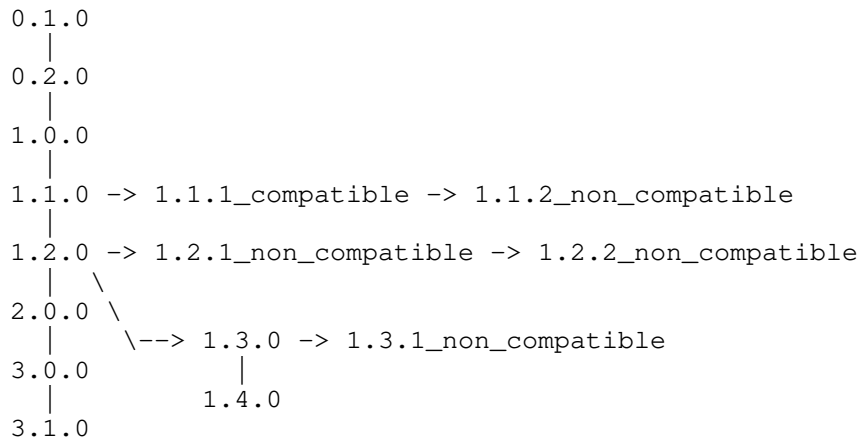
1. The including module has editorial changes
2. The submodule with the schema definition removed has non-backwards-compatible changes
3. The submodule with the schema definitions added has backwards-compatible changes

Note that the meaning of a submodule may change drastically despite having no changes in content or revision due to changes in other submodules belonging to the same module (e.g. groupings and typedefs declared in one submodule and used in another).

3.2.2. Examples for YANG semantic versions

The following diagram and explanation illustrate how YANG semantic versions work.

YANG Semantic versions for an example module:



The tree diagram above illustrates how the version history might evolve for an example module. The tree diagram only shows the parent/child ancestry relationships between the revisions. It does not describe the chronology of the revisions (i.e. when in time each revision was published relative to the other revisions).

The following description lists an example of what the chronological order of the revisions could look like, from oldest revision to newest:

0.1.0 - first pre-release module version

0.2.0 - second pre-release module version (with NBC changes)

1.0.0 - first release (may have NBC changes from 0.2.0)

1.1.0 - added new functionality, leaf "foo" (BC)

1.2.0 - added new functionality, leaf "baz" (BC)

2.0.0 - change existing model for performance reasons, e.g. re-key list (NBC)

1.3.0 - improve existing functionality, added leaf "foo-64" (BC)

1.1.1_compatible - backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0. This revision was created after 1.2.0 otherwise it may have been released as 1.2.0. (BC)

3.0.0 - NBC bugfix, rename "baz" to "bar"; also add new BC leaf "wibble"; (NBC)

1.3.1_non_compatible - backport NBC fix, rename "baz" to "bar" (NBC)

1.2.1_non_compatible - backport NBC fix, rename "baz" to "bar" (NBC)

1.1.2_non_compatible - NBC point bug fix, not required in 2.0.0 due to model changes (NBC)

1.4.0 - introduce new leaf "ghoti" (BC)

3.1.0 - introduce new leaf "wobble" (BC)

1.2.2_non_compatible - backport "wibble". This is a BC change but "non_compatible" modifier is sticky. (BC)

The partial ancestry relationships based on the semantic versioning numbers are as follows:

1.0.0 < 1.1.0 < 1.2.0 < 2.0.0 < 3.0.0 < 3.1.0

1.0.0 < 1.1.0 < 1.1.1_compatible < 1.1.2_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.2.1_non_compatible < 1.2.2_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 1.3.1_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 1.4.0

There is no ordering relationship between "1.1.1_non_compatible" and either "1.2.0" or "1.2.1_non_compatible", except that they share the common ancestor of "1.1.0".

Looking at the version number alone does not indicate ancestry. The module definition in "2.0.0", for example, does not contain all the contents of "1.3.0". Version "2.0.0" is not derived from "1.3.0".

3.3. YANG Semantic Version Update Rules

When a new revision of an artifact is produced, then the following rules define how the YANG semantic version for the new artifact revision is calculated, based on the changes between the two artifact revisions, and the YANG semantic version of the base artifact revision from which the changes are derived.

The following four rules specify the RECOMMENDED, and REQUIRED minimum, update to a YANG semantic version:

1. If an artifact is being updated in a non-backwards-compatible way, then the artifact version "X.Y.Z[_compatible|_non_compatible]" SHOULD be updated to "X+1.0.0" unless that version has already been used for this artifact but with different content, in which case the artifact version "X.Y.Z+1_non_compatible" SHOULD be used instead.

2. If an artifact is being updated in a backwards-compatible way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version SHOULD be updated to "X.Y+1.0", unless that version has already been used for this artifact but with different content, when the artifact version SHOULD be updated to "X.Y.Z+1_compatible" instead.

   ii "X.Y.Z_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_non_compatible".

3. If an artifact is being updated in an editorial way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version SHOULD be updated to "X.Y.Z+1"

   ii "X.Y.Z_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_non_compatible".

4. YANG artifact semantic version numbers beginning with 0, i.e., "0.X.Y", are regarded as pre-release definitions and need not follow the rules above. Either the MINOR or PATCH version numbers may be updated, regardless of whether the changes are non-backwards-compatible, backwards-compatible, or editorial. See Section 5 for more details on using this notation during module and submodule development.

5. Additional pre-release rules for modules that have had at least one release are specified in Section 5 .

Although artifacts SHOULD be updated according to the rules above, which specify the recommended (and minimum required) update to the version number, the following rules MAY be applied when choosing a new version number:

1. An artifact author MAY update the version number with a more significant update than described by the rules above. For example, an artifact could be given a new MAJOR version number (i.e., X+1.0.0), even though no non-backwards-compatible changes have occurred, or an artifact could be given a new MINOR version number (i.e., X.Y+1.0) even if the changes were only editorial.

2. An artifact author MAY skip version numbers. That is, an artifact's revision history could be 1.0.0, 1.1.0, and 1.3.0 where 1.2.0 is skipped. Note that skipping versions has an impact when importing modules by revision-or-derived. See Section 4 for more details on importing modules with revision-label version gaps.

Although YANG Semver always indicates when a non-backwards-compatible, or backwards-compatible change may have occurred to a YANG artifact, it does not guarantee that such a change has occurred, or that consumers of that YANG artifact will be impacted by the change. Hence, tooling, e.g., [I-D.ietf-netmod-yang-schema-comparison] , also plays an important role for comparing YANG artifacts and calculating the likely impact from changes.

[I-D.ietf-netmod-yang-module-versioning] defines the "rev:non-backwards-compatible" extension statement to indicate where non-backwards-compatible changes have occurred in the module revision history. If a revision entry in a module's revision history includes the "rev:non-backwards-compatible" statement then that MUST be reflected in any YANG semantic version associated with that revision. However, the reverse does not necessarily hold, i.e., if the MAJOR version has been incremented it does not necessarily mean that a "rev:non-backwards-compatible" statement would be present.

3.4. Examples of the YANG Semver Label

3.4.1. Example Module Using YANG Semver

Below is a sample YANG module that uses the YANG Semver revision-label based on the rules defined in this document.

```
    module example-versioned-module {
      yang-version 1.1;
      namespace "urn:example:versioned:module";
      prefix "exvermod";
      rev:revision-label-scheme "ysver:yang-semver";

      import ietf-yang-revisions { prefix "rev"; }
      import ietf-yang-semver { prefix "ysver"; }

      description
        "to be completed";

      revision 2017-08-30 {
        description "Backport 'wibble' leaf";
        rev:revision-label 1.2.2_non_compatible;
      }

      revision 2017-07-30 {
        description "Rename 'baz' to 'bar'";
        rev:revision-label 1.2.1_non_compatible;
        rev:non-backwards-compatible;
      }

      revision 2017-04-20 {
        description "Add new functionality, leaf 'baz'";
        rev:revision-label 1.2.0;
      }

      revision 2017-04-03 {
        description "Add new functionality, leaf 'foo'";
        rev:revision-label 1.1.0;
      }

      revision 2017-02-07 {
        description "First release version.";
        rev:revision-label 1.0.0;
      }

      // Note: semver rules do not apply to 0.X.Y labels.
      // The following pre-release revision statements would not
      // appear in any final published version of a module. They
      // are removed when the final version is published.
      // During the pre-release phase of development, only a
      // single one of these revision statements would appear

      // revision 2017-01-30 {
      //   description "NBC changes to initial revision";
      //   rev:revision-label 0.2.0;
```

```
      //   rev:non-backwards-compatible; // optional
          //                                 // (theoretically no
          //                                 // 'previous released version')
      // }

      // revision 2017-01-26 {
      //   description "Initial module version";
      //   rev:revision-label 0.1.0;
      // }

      //YANG module definition starts here
    }
```

3.4.2. Example of Package Using YANG Semver

Below is an example YANG package that uses the semver revision label based on the rules defined in this document.

```
      {
        "ietf-yang-instance-data:instance-data-set": {
          "name": "example-yang-pkg",
          "target-ptr": "TBD",
          "timestamp": "2018-09-06T17:00:00Z",
          "description": "Example IETF package definition",
          "content-data": {
            "ietf-yang-package:yang-package": {
              "name": "example-yang-pkg",
              "version": "1.3.1",
              ...
     }
```

4. Import Module by Semantic Version

[I-D.ietf-netmod-yang-module-versioning] allows for imports to be done based on a module or a derived revision of a module. The rev:revision-or-derived statement can specify either a revision date or a revision label. The YANG Semver revision-label value can be used as the argument to rev:revision-or-derived . When used as such, any module that contains exactly the same YANG semantic version in its revision history may be used to satisfy the import requirement. For example:

```
          import example-module {
            rev:revision-or-derived 3.0.0;
          }
```

Note: the import lookup does not stop when a non-backward-compatible change is encountered. That is, if module B imports a module A at or derived from version 2.0.0, resolving that import will pass through a revision of module A with version "2.1.0_non_compatible" in order to determine if the present instance of module A derives from "2.0.0".

If an import by revision-or-derived cannot locate the specified revision-label in a given module's revision history, that import will fail. This is noted in the case of version gaps. That is, if a module's history includes "1.0.0", "1.1.0", and "1.3.0", an import from revision-or-derived at "1.2.0" will be unable to locate the specified revision entry and thus the import cannot be satisfied.

5. Guidelines for Using Semver During Module Development

This section and the IETF-specific sub-section below provides YANG Semver-specific guidelines to consider when developing new YANG modules. As such this section updates [RFC8407] .

Development of a brand new YANG module or submodule outside of the IETF that uses YANG Semver as its revision-label scheme SHOULD begin with a 0 for the MAJOR version component. This allows the module or submodule to disregard strict semver rules with respect to non-backwards-compatible changes during its initial development. However, module or submodule developers MAY choose to use the semver pre-release syntax instead with a 1 for the MAJOR version component. For example, an initial module or submodule revision-label might be either 0.0.1 or 1.0.0-alpha.1. If the authors choose to use the 0 MAJOR version component scheme, they MAY switch to the pre-release scheme with a MAJOR version component of 1 when the module or submodule is nearing initial release (e.g., a module's or submodule's revision label may transition from 0.3.0 to 1.0.0-beta.1 to indicate it is more mature and ready for testing).

When using pre-release notation, the format MUST include at least one alphabetic component and MUST end with a '.' or '-' and then one or more digits. These alphanumeric components will be used when deciding pre-release precedence. The following are examples of valid pre-release versions

1.0.0-alpha.1

1.0.0-alpha.3

2.1.0-beta.42

3.0.0-202007.rc.1

When developing a new revision of an existing module or submodule using the YANG semver revision-label scheme, the intended target semver version MUST be used along with pre-release notation. For example, if a released module or submodule which has a current revision-label of 1.0.0 is being modified with the intent to make non-backwards-compatible changes, the first development MAJOR version component must be 2 with some pre-release notation such as -alpha.1, making the version 2.0.0-alpha.1. That said, every publicly available release of a module or submodule MUST have a unique YANG semver revision-label (where a publicly available release is one that could be implemented by a vendor or consumed by an end user). Therefore, it may be prudent to include the year or year and month development began (e.g., 2.0.0-201907-alpha.1). As a module or submodule undergoes development, it is possible that the original intent changes. For example, a 1.0.0 version of a module or submodule that was destined to become 2.0.0 after a development cycle may have had a scope change such that the final version has no non-backwards-compatible changes and becomes 1.1.0 instead. This change is acceptable to make during the development phase so long as pre-release notation is present in both versions (e.g., 2.0.0-alpha.3 becomes 1.1.0-alpha.4). However, on the next development cycle (after 1.1.0 is released), if again the new target release is 2.0.0, new pre-release components must be used such that every revision-label for a given module or submodule MUST be unique throughout its entire lifecycle (e.g., the first pre-release version might be 2.0.0-202005-alpha.1 if keeping the same year and month notation mentioned above).

5.1. Pre-release Version Precedence

As a module or submodule is developed, the scope of the work may change. That is, while a ratified module or submodule with revision-label 1.0.0 is initially intended to become 2.0.0 in its next ratified version, the scope of work may change such that the final version is 1.1.0. During the development cycle, the pre-release versions could move from 2.0.0-some-pre-release-tag to 1.1.0-some-pre-release-tag. This downwards changing of version numbers makes it difficult to evaluate semver rules between pre-release versions. However, taken independently, each pre-release version can be compared to the previously ratified version (e.g., 1.1.0-some-pre-release-tag and 2.0.0-some-pre-release-tag can each be compared to 1.0.0). Module and submodule developers SHOULD maintain only one revision statement in a pre-released module or submodule that reflects the latest revision. IETF authors MAY choose to include an appendix in the associated draft to track overall changes to the module or submodule.

5.2. YANG Semver in IETF Modules

All published IETF modules and submodules MUST use YANG semantic versions for their revision-labels.

Development of a new module or submodule within the IETF SHOULD begin with the 0 MAJOR number scheme as described above. When revising an existing IETF module or submodule, the revision-label MUST use the target (i.e., intended) MAJOR and MINOR version components with a 0 PATCH version component. If the intended ratified release will be non-backward-compatible with the current ratified release, the MINOR version component MUST be 0.

All IETF modules and submodules in development MUST use the whole document name as a pre-release version string, including the current document revision. For example, if a module or submodule which is currently released at version 1.0.0 is being revised to include non-backwards-compatible changes in draft-user-netmod-foo, its development revision-labels MUST include 2.0.0-draft-user-netmod-foo followed by the document's revision (e.g., 2.0.0-draft-user-netmod-foo-02). This will ensure each pre-release version is unique across the lifecycle of the module or submodule. Even when using the 0 MAJOR version for initial module or submodule development (where MINOR and PATCH can change), appending the draft name as a pre-release component helps to ensure uniqueness when there are perhaps multiple, parallel efforts creating the same module or submodule.

For IETF YANG modules and submodules that have already been published, revision-labels MUST be retroactively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver version rules specified in Section 3.3 . For example, if a module or submodule started out in the pre-NMDA ([RFC8342]) world, and then had NMDA support added without removing any legacy "state" branches -- and you are looking to add additional new features -- a sensible choice for the target YANG Semver would be 1.2.0 (since 1.0.0 would have been the initial, pre-NMDA release, and 1.1.0 would have been the NMDA revision).

See Appendix A for a detailed example of IETF pre-release versions.

6. YANG Module

This YANG module contains the typedef for the YANG semantic version and the identity to signal its use.

```
<CODE BEGINS> file "ietf-yang-semver@2021-11-04.yang"
module ietf-yang-semver {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-semver";
  prefix ysver;
  rev:revision-label-scheme "yang-semver";

  import ietf-yang-revisions {
    prefix rev;
  }

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

     Author:   Joe Clarke
               <mailto:jclarke@cisco.com>
     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>
     Author:   Reshad Rahman
               <mailto:reshad@yahoo.com>
     Author:   Balazs Lengyel
               <mailto:balazs.lengyel@ericsson.com>
     Author:   Jason Sterne
               <mailto:jason.sterne@nokia.com>
     Author:   Benoit Claise
               <mailto:benoit.claise@huawei.com>";
  description
    "This module provides type and grouping definitions for YANG
     packages.

     Copyright (c) 2021 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
```

```
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     // RFC Ed. update the rev:revision-label to "1.0.0".

     revision 2021-11-04 {
       rev:revision-label "1.0.0-draft-ietf-netmod-yang-semver-05";
       description
         "Initial revision";
       reference
         "RFC XXXX: YANG Semantic Versioning.";
     }

     /*
      * Identities
      */

     identity yang-semver {
       base rev:revision-label-scheme-base;
       description
         "The revision-label scheme corresponds to the YANG Semver scheme
          which is defined by the pattern in the 'version' typedef below.
          The rules governing this revision-label scheme are defined in the
          reference for this identity.";
       reference
         "RFC XXXX: YANG Semantic Versioning.";
     }

     /*
      * Typedefs
      */

     typedef version {
       type rev:revision-label {
         pattern '[0-9]+[.][0-9]+[.][0-9]+(_(non_)?compatible)?'
         + '(-[A-Za-z0-9.-]+[.-][0-9]+)?([+][A-Za-z0-9.-]+)?';
       }
       description
         "Represents a YANG semantic version.  The rules governing the
          use of this revision label scheme are defined in the reference for
          this typedef.";
       reference
         "RFC XXXX: YANG Semantic Versioning.";
     }
   }
   <CODE ENDS>
```

7. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The design team consists of the following members whom have worked on the YANG versioning project:

* Balazs Lengyel
* Benoit Claise
* Bo Wu
* Ebben Aries
* Jan Lindblad
* Jason Sterne
* Joe Clarke
* Juergen Schoenwaelder
* Mahesh Jethanandani
* Michael (Wangzitao)
* Qin Wu
* Reshad Rahman
* Rob Wilton

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update] . We would like the thank Kevin D'Souza for his initial work in this problem space.

Discussons on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models based on their own [openconfigsemver] . We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

8. Security Considerations

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

9. IANA Considerations

9.1. YANG Module Registrations

This document requests IANA to register a URI in the "IETF XML Registry" [RFC3688] . Following the format in RFC 3688, the following registration is requested.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-semver

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

The following YANG module is requested to be registred in the "IANA Module Names" [RFC6020] . Following the format in RFC 6020, the following registrations are requested:

The ietf-yang-semver module:

Name: ietf-yang-semver

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-semver

Prefix: yangver

Reference: [RFCXXXX]

9.2. Guidance for YANG Semver in IANA maintained YANG modules and submodules

Note for IANA (to be removed by the RFC editor): Please check that the registries and IANA YANG modules and submodules are referenced in the appropriate way.

IANA is responsible for maintaining and versioning some YANG modules and submodules, e.g., iana-if-types.yang [IfTypeYang] and iana-routing-types.yang [RoutingTypesYang] .

In addition to following the rules specified in the IANA Considerations section of [I-D.ietf-netmod-yang-module-versioning] , IANA maintained YANG modules and submodules MUST also include a YANG Semver revision label for all new revisions, as defined in Section 3 .

The YANG Semver version associated with the new revision MUST follow the rules defined in Section 3.3 .

Note: For IANA maintained YANG modules and submodules that have already been published, revision labels MUST be retroactively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver rules specified in Section 3.3 .

Most changes to IANA maintained YANG modules and submodules are expected to be backwards-compatible changes and classified as MINOR version changes. The PATCH version may be incremented instead when only editorial changes are made, and the MAJOR version would be incremented if non-backwards-compatible changes are made.

Given that IANA maintained YANG modules are versioned with a linear history, it is anticipated that it should not be necessary to use the "_compatible" or "_non_compatible" modifiers to the "Z_COMPAT" version element.

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/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[I-D.ietf-netmod-yang-module-versioning] Wilton, R., Rahman, R., Lengyel, B., Clarke, J., and J. Sterne, "Updated YANG Module Revision Handling", Work in

Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-05, 8 November 2021, <https://tools.ietf.org/html/draft-ietf-netmod-yang-module-versioning-05>.

10.2. Informative References

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

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-02, 25 October 2021, <https://tools.ietf.org/html/draft-ietf-netmod-yang-packages-02>.

[I-D.ietf-netmod-yang-schema-comparison]
Wilton, R., "YANG Schema Comparison", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-schema-comparison-01, 2 November 2020, <https://tools.ietf.org/html/draft-ietf-netmod-yang-schema-comparison-01>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[openconfigsemver]
"Semantic Versioning for Openconfig Models", <http://www.openconfig.net/docs/semver/>.

[semver] "Semantic Versioning 2.0.0 (text from June 19, 2020)", <https://github.com/semver/semver/blob/8b2e8eec394948632957639dfa99fc7ec6286911/semver.md>.

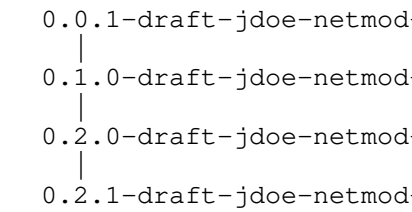
[IfTypeYang]
"iana-if-type YANG Module", <https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml>.

[RoutingTypesYang]
"iana-routing-types YANG Module",
<https://www.iana.org/assignments/iana-routing-types/iana-routing-types.xhtml>.

Appendix A. Example IETF Module Development

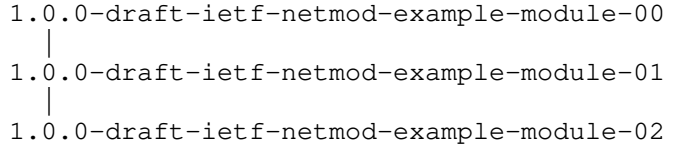
Assume a new YANG module is being developed in the netmod working group in the IETF. Initially, this module is being developed in an individual internet draft, draft-jdoe-netmod-example-module. The following represents the initial version tree (i.e., value of revision-label) of the module as it's being initially developed.

Version lineage for initial module development:



At this point, development stabilizes, and the workgroup adopts the draft. Thus now the draft becomes draft-ietf-netmod-example-module. The initial pre-release lineage continues as follows.

Continued version lineage after adoption:



At this point, the draft is ratified and becomes RFC12345 and the YANG module version becomes 1.0.0.

A time later, the module needs to be revised to add additional capabilities. Development will be done in a backwards-compatible way. Two new individual drafts are proposed to go about adding the capabilities in different ways: draft-jdoe-netmod-exmod-enhancements and draft-jadoe-netmod-exmod-changes. These are initially developed in parallel with the following versions.

Parallel development for next module revision:

```
      1.1.0-draft-jdoe-netmod-exmod-enhancements-00 || 1.1.0-draft-jadoe-netmod-e
xmod-changes-00
        |                                                |
      1.1.0-draft-jdoe-netmod-exmod-enhancements-01 || 1.1.0-draft-jadoe-netmod-e
xmod-changes-01
```

   At this point, the WG decides to merge some aspects of both and adopt the work in jadoe's draft as draft-ietf-netmod-exmod-changes. A single version lineage continues.

```
        1.1.0-draft-ietf-netmod-exmod-changes-00
          |
        1.1.0-draft-ietf-netmod-exmod-changes-01
          |
        1.1.0-draft-ietf-netmod-exmod-changes-02
          |
        1.1.0-draft-ietf-netmod-exmod-changes-03
```

   The draft is ratified, and the new module version becomes 1.1.0.

Authors' Addresses

   Joe Clarke (editor)
   Cisco Systems, Inc.
   7200-12 Kit Creek Rd
   Research Triangle Park, North Carolina
   United States of America

   Phone: +1-919-392-2867
   Email: jclarke@cisco.com

   Robert Wilton (editor)
   Cisco Systems, Inc.

   Email: rwilton@cisco.com

   Reshad Rahman

   Email: reshad@yahoo.com

   Balazs Lengyel
   Ericsson
   1117 Budapest
   Magyar Tudosok Korutja
   Hungary

   Phone: +36-70-330-7909

Email: balazs.lengyel@ericsson.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Benoit Claise
Huawei

Email: benoit.claise@huawei.com

NETMOD
Internet-Draft
Intended status: Standards Track
Expires: 14 August 2022

Q. Ma, Ed.
Q. Wu
Huawei
H. Li
HPE

10 February 2022

Immutable Metadata Annotation
draft-ma-netmod-immutable-flag-00

Abstract

This document defines a metadata annotation [RFC7952] named "immutable" to indicate the immutability of a particular instantiated data node. Any instantiated data node annotated with immutable="true" by the server is read-only to the clients of YANG-driven management protocols, such as NETCONF, RESTCONF and other management operations (e.g., SNMP and CLI requests).

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 August 2022.

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC7950] is a data modeling language used to model both state and configuration data, based on the "config" statement. However, there are some configuration data which may not be writable to clients, e.g., the interface name and type values created by the system due to the hardware currently present in the device cannot be modified by clients, while configurations such as MTU created by the system are free to be modified by the client.

Allowing some configuration modifiable while others not is inconsistent and introduces ambiguity to clients.

To address this issue, this document defines a metadata annotation [RFC7952] named "immutable" to indicate the immutability characteristic of a particular instantiated data node. Any instantiated data node marked with immutable="true" by the server is read-only to the clients of YANG-driven management protocols, such as NETCONF, RESTCONF and other management operations (e.g., SNMP and CLI requests).

1.1. Terminology

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.

The following terms are defined in [RFC6241] and [RFC8341] and are not redefined here:

* configuration data

* access operation

The following terms are defined in this document:

immutable: A metadata annotation indicating the immutability of a data node. An immutable data node is read-only to clients. Note that "immutable" is used to annotate instances of YANG data nodes rather than schema nodes. For instance, a "list" data node may exist in multiple instances in the data tree, "immutable" can annotate some of the instances as read-only, while others are not.

2. "Immutable" Metadata Annotation Definition

The "immutable" flag is used to indicate the immutability of a particular instantiated data node. It applies to the container, anydata, anyxml, leaf, list and leaf-list entries. The values are boolean types indicating whether the data node instance is immutable or not.

When the client retrieves a particular datastore, immutable data node instances MUST be annotated with immutable="true" by the server. If the "immutable" metadata annotation is not specified, the default is the same as the "immutable" value of the parent node. The default "immutable" value for a top level data node is false.

Any data node instance annotated with "immutable=true" is read-only to clients, which means that the following access operations for a particular instance are not allowed:

* Create: add descendant node instances for a container instance annotated with "immutable";

* Delete: delete a data node instance annotated with "immutable" or its child node instances from a datastore, e.g., delete an immutable list or leaf-list entry;

* Update: update an existing data node instance annotated with "immutable" or its child node instances in a datastore, e.g., modify the value of an immutable leaf data node;

However the following operations SHOULD be allowed:

* Create an immutable data node with a same value initially set by the system if it doesn't exist in the datastore, e.g., explicitly configure a system generated interface name in <running>;

* Delete the parent node of an immutable data node unless the parent node is also annotated with "immutable=true", e.g., /interfaces/interface/type leaf instance is immutable, but the deletion to the /interfaces/interface list entry is allowed;

Comment: Should we allow the client delete an "immutable" system instantiated node in <running> (see Appendix A.1.3)?

Comment: Annotations can only be attached to individual list/leaf-list entry, how would the client know if it's to apply to the whole list/leaf-list, the list/leaf-list entries, or both?

Servers MUST reject any attempt to the "create", "delete" and "update" access operations on an immutable data node. The error reporting is performed at various different time according to the selected target datastore. If the target datastore is "running" or "startup", the server should reply with an "invalid-value" at a <edit-config> operation time. If the target datastore is "candidate", the "invalid-value" error response to update an immutable data node is delayed until a <commit> or <validate> operation takes place. For an example of an "invalid-value" error response, see Appendix A.1.2.

3. YANG Module

```
<CODE BEGINS>
 file="ietf-immutable@2022-01-05.yang"
  module ietf-immutable {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-immutable";
    prefix im;

    import ietf-yang-metadata {
      prefix md;
    }

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

    contact
      "WG Web: <https://datatracker.ietf.org/wg/netmod/>

       WG List: <mailto:netmod@ietf.org>

       Author: Qiufang Ma
               <mailto:maqiufang1@huawei.com>

       Author: Qin Wu
               <mailto:bill.wu@huawei.com>

       Author: Hongwei Li
               <mailto:flycoolman@gmail.com>";

    description
      "This module defines a metadata annotation named 'immutable'
       to indicate the immutability of a particular instantiated
       data node. Any instantiated data node marked with
       immutable='true' by the server is read-only to the clients
       of YANG-driven management protocols, such as NETCONF,
       RESTCONF as well as SNMP and CLI requests.

       Copyright (c) 2021 IETF Trust and the persons identified
       as authors of the code. All rights reserved.

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

       This version of this YANG module is part of RFC HHHH
       (https://www.rfc-editor.org/info/rfcHHHH); 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 2022-01-05 {
         description
           "Initial revision.";
         reference
           "RFC XXX: Immutable Metadata Annotation";
       }

       md:annotation immutable {
         type boolean;
         description
           "The 'immutable' annotation indicates the immutability of an
            instantiated data node. Any data node instance marked as
            'immutable=true' is read-only to clients and cannot be
            created/deleted/changed through NETCONF, RESTCONF or CLI.
            It applies to the container, anydata, anyxml and leaf
            instances, list and leaf-list entries.
            If not specified for a given configuration data node
            instance, then the immutability is the same as its parent
            node instance in the data tree. The default for any top-level
            configuration data node instance is false if not specified.";
       }
     }
   <CODE ENDS>
```

4. IANA Considerations

4.1. The "IETF XML" Registry

This document registers one XML namespace URN in the 'IETF XML registry', following the format defined in [RFC3688].

```
   URI: urn:ietf:params:xml:ns:yang:ietf-immutable
   Registrant Contact: The IESG.
   XML: N/A, the requested URIs are XML namespaces.
```

4.2. The "YANG Module Names" Registry

This document registers one module name in the 'YANG Module Names' registry, defined in [RFC6020].

```
name: ietf-immutable
prefix: im
namespace: urn:ietf:params:xml:ns:yang:ietf-immutable
RFC: XXXX // RFC Ed.: replace XXXX and remove this comment
```

5. Security Considerations

The YANG module specified in this document defines a metadata annotation for data nodes that is designed to be accessed network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

Since immutable information is tied to applied configuration values, it is only accessible to clients that have the permissions to read the applied configuration values.

The security considerations for the Defining and Using Metadata with YANG (see Section 9 of [RFC7952]) apply to the metadata annotation defined in this document.

6. References

6.1. 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/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

6.2. Informative References

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Appendix A. Usage Examples

A.1. Interface Example

In this section, the following fragment of a fictional interface module is used:

```
      container interfaces {
        list interface {
          key "name";
          leaf name {
            type string;
          }
          leaf type {
            type identityref {
              base ianaift:iana-interface-type;
            }
          }
          leaf mtu {
            type uint16;
          }
          leaf-list ip-address {
            type inet:ip-address;
          }
        }
      }
```

In this example model, when an interface is physically present, the system will create an interface entry automatically with valid name and type values. Let's say that there is a system-defined interface "eth0" in <operational>:

```
<interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
        xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
        xmlns:im="urn:ietf:params:xml:ns:yang:ietf-immutable"
        or:origin="or:system">
  <interface>
    <name>eth0</name>
    <type im:immutable="true">ianaift:ethernetCsmacd</type>
    <mtu>1500</mtu>
  </interface>
</interfaces>
```

The "type" leaf instance is annotated with immutable="true" in the RPC response, which means that it is not allowed to be modified by client configuration operations. Note that although the "name" defined as a list key leaf isn't annotated with immutable="true", modification of the key value for a particular list entry is never allowed. Deletion to the whole list entry of interface "eth0" should be allowed in this case, since there is no annotation for the "type" leaf instance's parent node.

A.1.1. Creating an "immutable" Interface Type

The server should accept the configuration to set the leaf "type" to the value same as in <operational>:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     message-id="101">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interface xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"
            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
            xc:operation="create">
        <name>eth0</name>
        <type>ianaift:ethernetCsmacd</type>
      </interface>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

A.1.2. Changing an "immutable" Interface Type

If a client tries to change the type of an interface to a value that doesn’t match the real type of the interface used by the system, the server must reject the request:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interface xc:operation="merge"
            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
        <name>eth0</name>
        <type>ianaift:tunnel</type>
      </interface>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-path xmlns:t="http://example.com/schema/1.2/config">
      /interfaces/interface[name="eth0"]/type
    </error-path>
    <error-message xml:lang="en">
      Invalid type for interface eth0
    </error-message>
  </rpc-error>
</rpc-reply>
```

A.1.3. Deleting an "immutable" Interface Type

Should the server accept the configuration of deleting the type of interface named "eth0" from the running configuration?

```
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interface>
        <name>eth0</name>
        <type xc:operation="delete">ianaift:ethernetCsmacd</type>
      </interface>
    </config>
  </edit-config>
</rpc>
```

Even the type of interface named "eth0" is deleted from the running configuration, the client which performs a <get-data> towards <operational> will still get:

```
<interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
            xmlns:im="urn:ietf:params:xml:ns:yang:ietf-immutable"
            or:origin="or:intended">
  <interface>
    <name>eth0</name>
    <type im:immutable="true" or:origin="or:intended">
      ianaift:ethernetCsmacd
    </type>
    <mtu>1500</mtu>
  </interface>
</interfaces>
```

A.2. Built-in Access Control Example

In this example, the "ietf-netconf-acm" YANG module defined in [RFC8341] is used. Suppose when the device is powered on, the system may provide some built-in access control groups, for each access control group there exists at least one built-in user, which might be read-only and cannot be modified by the client. In addition, the system also can predefine some access control rules for a specific group. Some rules can be modified, while others are not (e.g., access control rule for a root account should be immutable). In addition, clients can also define their own groups and access control rules freely.

The built-in access control groups and rules are provided by the system, they are present in <operational>. The example below illustrates what the built-in groups and rules might look like for a server after it boots:

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"
      xmlns:im="urn:ietf:params:xml:ns:yang:ietf-immutable">
  <groups>
    <group>
      <name>admin</name>
      <user-name im:immutable="true">admin</user-name>
    </group>
    <group>
      <name>monitor</name>
      <user-name im:immutable="true">user</user-name>
    </group>
    <group>
      <name>visit</name>
      <user-name im:immutable="true">guest</user-name>
    </group>
  </groups>
  <rule-list im:immutable="true">
    <name>admin-acl</name>
    <group>admin</group>
    <rule>
      <name>permit-all</name>
      <module-name>*</module-name>
      <access-operations>*</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
  <rule-list>
    <name>monitor-acl</name>
    <group>visit</group>
    <rule>
      <name>permit-monitor</name>
      <module-name>ietf-netconf-monitoring</module-name>
      <access-operations>read</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
  <rule-list>
    <name>visit-acl</name>
    <group>visit</group>
    <rule>
      <name>deny-ncm</name>
      <module-name>ietf-netconf-monitoring</module-name>
      <access-operations>*</access-operations>
      <action>deny</action>
    </rule>
  </rule-list>
</nacm>
```

As indicated in the above XML snippet, the system predefines three access control groups and for each group there is one built-in immutable user, which means that the client can define new users for a particular access control group, but deletion of a built-in user-name is not allowed. The system also provides three access control list entries, one for each predefined group. Only the access control rule for the "admin" group is read-only to clients.

Authors' Addresses

Qiufang Ma (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: maqiufang1@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: bill.wu@huawei.com

Hongwei Li
HPE

Email: flycoolman@gmail.com

NETMOD
Internet-Draft
Updates: RFC8342, RFC6241, RFC8526, RFC8040 (if approved)
Intended status: Standards Track
Expires: 18 August 2022

Q. Ma, Ed.
Huawei
K. Watsen
Watsen Networks
Q. Wu
C. Feng
Huawei
J. Lindblad
Cisco Systems

14 February 2022

System-defined Configuration
draft-ma-netmod-with-system-02

Abstract

This document updates NMDA [RFC8342] to define a read-only conventional configuration datastore called "system" to hold system-defined configurations. To avoid clients' explicit copy/paste of referenced system-defined configuration, a "resolve-system" parameter has been defined to allow the server acting as a "system client" to populate referenced system-defined nodes automatically. The solution enables clients to reference nodes defined in <system>, overwrite values of configurations defined in <system>, and configure descendant nodes of system-defined nodes.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 18 August 2022.

Copyright Notice

Table of Contents

1. Introduction

NMDA [RFC8342] defines system configuration as the configuration that is supplied by the device itself and should be present in <operational> when it is in use.

However, there is a desire to enable a server to better document the system configuration. Clients can benefit from a standard mechanism to see what system configuration is available in a server.

In some cases, the client references a system configuration which isn't returned when the datastore is read. Having to copy the entire contents of the system configuration into <running> should be avoided or reduced when possible while ensuring that all referential integrity constraints are satisfied.

In some other cases, configuration of descendant nodes of system defined configuration needs to be supported. For example, the system configuration may contain an almost empty physical interface, while the client needs to be able to add, modify, remove a number of descendant nodes. Some descendant nodes may not be modifiable (e.g., "name" and "type" set by the system).

This document updates NMDA [RFC8342] to define a read-only conventional configuration datastore called "system" to hold system-defined configurations. To avoid clients' explicit copy/paste of referenced system-defined configuration, a "resolve-system" parameter has been defined to allow the server acting as a "system client" to populate referenced system-defined nodes automatically. The solution

enables clients to reference nodes defined in <system>, overwrite values of configurations defined in <system>, and configure descendant nodes of system-defined nodes.

Conformance to this document requires servers to implement the "ietf-system-datastore" YANG Module.

1.1. Terminology

This document assumes that the reader is familiar with the contents of [RFC6241], [RFC7950], [RFC8342], [RFC8407], and [RFC8525] and uses terminologies from those documents.

The following terms are defined in this document as follows:

System configuration: Configuration that is provided by the system itself [RFC8342].

System configuration datastore: A configuration datastore holding the complete configuration provided by the system itself. This datastore is referred to as "<system>".

This document redefines the term "conventional configuration datastore" from RFC 8342 to add "system" to the list conventional configuration datastores:

Conventional configuration datastore: One of the following set of configuration datastores: <running>, <startup>, <candidate>, <system>, and <intended>. These datastores share a common datastore schema, and protocol operations allow copying data between these datastores. The term "conventional" is chosen as a generic umbrella term for these datastores.

1.2. Requirements Language

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.

1.3. Updates to RFC 8342

This document updates RFC 8342 to define a configuration datastore called "system" to hold system configuration, it also redefines the term "conventional configuration datastore" from RFC 8342 to add "system" to the list conventional configuration datastores. The contents of <system> datastore are read-only to clients but may change dynamically. The <system> aware client may retrieve all three types of system configuration defined in Section 2, reference nodes defined in <system>, overwrite values of configurations defined in <system>, and configure descendant nodes of system-defined nodes.

The server will merge <running> and <system> to create <intended>. As always, system configuration will appear in <operational> with origin="system".

The <system> datastore makes system configuration visible to clients in order for being referenced or configurable prior to present in <operational>.

1.4. Updates to RFC 6241, RFC 8526

This document augments <edit-config> and <edit-data> RPC operations defined in [RFC6241] and [RFC8526] respectively, with a new additional input parameter "resolve-system".

The "resolve-system" parameter is optional and has no value. When it is provided and the server detects that there is a reference to a system-defined node during the validation, the server will automatically populate the referenced system configuration into the validated datastore to make the configuration valid without the client doing so explicitly. Legacy Clients interacting with servers that support this parameter don’t see any changes in <edit-config> and <edit-data> behaviors.

According to the NETCONF constraint enforcement model defined in the section 8.3 of [RFC7950], if the target datastore of the <edit-config> or <edit-data> is "running" or "startup", the server-populating of the target datastore MUST be enforced at the end of the <edit-config> or <edit-data> operations during the validation. If the target datastore of the <edit-config> or <edit-data> is "candidate", the server-populating of the target datastore is delayed until a <commit> or <validate> operation takes place.

1.5. Updates to RFC 8040

This document extends Section 4.8 of [RFC8040] to add a new query parameter "resolve-system".

The "resolve-system" parameter controls whether to allow a server populate any referenced system-defined configuration automatically without the client doing so explicitly. This parameter is only allowed with no values carried. If this parameter has any unexpected value, then a "400 Bad Request" status-line is returned.

Name	Methods	Description
resolve-system	POST, PUT	resolve any references not resolved by the client and populate referenced system configuration into <running> automatically. This parameter can be given in any order.

2. Kinds of System Configuration

There are three types of system configurations: immediately-active system configuration, conditionally-active system configuration and inactive-until-referenced system configuration.

2.1. Immediately-Active

Immediately-active system configurations are those applied and active immediately (e.g., a loop-back interface) , irrespective of physical resource present or not, a special functionality enabled or not.

2.2. Conditionally-Active

System configurations which are provided and activated based on specific conditions being met in a system, e.g., if a physical resource is present (e.g., insert interface card), the system will automatically detect it and load pre-provisioned configuration; when the physical resource is not present(remove interface card), the system configuration will be automatically cleared. Another example is when a special functionality is enabled, e.g., when QoS function is enabled, QoS policies are automatically created by the system.

2.3. Inactive-Until-Referenced

There are some predefined objects(e.g., application ids, anti-x signatures, trust anchor certs, etc.) as a convenience for the clients. The clients can also define their own data objects for their unique requirements. Inactive-until-referenced system configurations are not applied and active immediately but only after they are referenced by client-defined configuration.

3. Static Characteristics

3.1. Read-only to Clients

From the client's perspective, the contents of the <system> datastore are read-only. There is no way to delete system configuration from a server. Any deletable system-provided configuration must be defined in <factory-default> [RFC8808], which is used to initialize <running> when the device is first-time powered on or reset to its factory default condition.

3.2. May Change via Software Upgrades

System configuration MAY change dynamically, e.g., depending on factors like device upgrade or if system-controlled resources(e.g., HW available) change. In some implementations, when QoS function is enabled, QoS-related predefined policies are created by system. If the system configuration gets changed, YANG notification (e.g., "push-change-update" notification) [RFC8641][RFC8639][RFC6470] can be used to notify the client.

3.3. No Impact to <operational>

This work intends to have no impact to <operational>. As always, system configuration will appear in <operational> with "origin=system". This work enables a subset of those system generated nodes to be defined like configuration, i.e., made visible to clients in order for being referenced or configurable prior to present in <operational>. The referenced system configuration in <running> automatically copied from <system> by the server MUST have its origin set to "system" when present in <operational>. "Config false" nodes are completely out of scope, hence existing "config false" nodes are not impacted by this work.

4. Dynamic Behavior

4.1. Conceptual Model

This document introduces a mandatory datastore named "system" which is used to hold all three types of system configurations defined in Section 2.

When the device is powered on, immediately-active system configuration will be provided and activated immediately but inactive-until-referenced system configuration only becomes active if it is referenced by client-defined configuration. While conditionally-active system configuration will be created and immediately activated if the condition on system resources is met when the device is powered on or running.

All above three types of system configurations will go into <system>. The client may reference nodes defined in <system>, overwrite values of configurations defined in <system>, and configure descendant nodes of system-defined nodes in <running>. Then the server will merge <running> and <system> to create <intended>, in which process, <running> MAY overwrite and/or extend <system>. If a server implements <intended>, <system> MUST be merged into <intended>.

Servers MUST enforce that configuration references in <running> are resolved within the <running> datastore and ensure that <running> contains any referenced system objects. Clients MUST either explicitly configure system-defined nodes in <running> or use the "resolve-system" parameter. The server MUST enforce that the referenced system nodes injected into <running> by the client is consistent with <system>. Note that only <system> aware clients copy referenced system nodes from <system>. How clients unaware of the <system> datastore can find appropriate configurations are beyond the scope of this document.

No matter how the referenced system objects are populated, the nodes copied into <running> would always be returned in a read of <running>, regardless if the client is <system> aware.

4.2. Servers Auto-populating Referenced System Configuration

This document defines a new parameter "resolve-system" to the input for the <edit-config> and <edit-data> operations. Clients that are aware of the "resolve-system" parameter MAY use this parameter to avoid the requirement to provide a referentially complete configuration in <running>.

If the "resolve-system" is present, the server MUST populate relevant referenced system-defined nodes into the target datastore without the client doing the copy/paste explicitly, to resolve any references not resolved by the client. The server acting as a "system client" like any other remote clients populates the referenced system-defined nodes when triggered by the "resolve-system" parameter. If the "resolve-system" parameter is not given by the client, the server MUST NOT modify <running> in any way not specified by the client.

The server may automatically configure the list entries (with at least the keys) in the target datastore (e.g., <running>) for any system configuration list entries that are referenced elsewhere by the clients. Not all the contents of the list entry (i.e., the descendant nodes) are necessarily populated by the sever - only the parts that are required to make the <running> valid. A read back of <running> (i.e., <get>, <get-config> or <get-data> operation) returns those automatically populated nodes.

4.3. Explicit Declaration of System Configuration

It is also possible for a client to explicitly declare system configuration nodes in the target datastore (e.g., <running>) with the same values as in <system>. When a client configures a node (list entry, leaf, etc) in <running> that matches the same node and value in <system>, then that node becomes part of <running>. A read back of <running> returns those explicitly configured nodes.

This explicit configuration of system configuration in <running> can be useful, for example, when the client doesn't want a "system client" to have a role or hasn't implemented the "resolve-system" parameter. The client can explicitly declare (i.e. configure in <running>) the list entries (with at least the keys) for any system configuration list entries that are referenced elsewhere in <running>. Similarly, The client does not necessarily need to declare all the contents of the list entry (i.e. the descendant nodes) - only the parts that are required to make the <running> appear valid.

4.4. Modifying (overriding) System Configuration

In some cases, a server may allow some parts of system configuration to be modified. List keys in system configuration can't be changed by a client, but other descendant nodes in a list entry may be modifiable or non-modifiable. Leafs and leaf-lists outside of lists may also be modifiable or non-modifiable. Modification of system configuration is achieved by the client writing configuration to <running> that overrides the system configuration. Client configuration statements in <running> take precedence over system configuration nodes in <system> if the server allows the nodes to be modified. If a system configuration node is non-modifiable, then writing a different value for that node in <running> MUST return an error.

A server may also allow a client to add data nodes to a list entry in <system> by writing those additional nodes in <running>. Those additional data nodes may not exist in <system> (i.e. an *addition* rather than an override).

While modifying (overriding) system configuration nodes may be supported by a server, there is no mechanism for deleting a system configuration node. A "mandatory true" leaf, for example, may have a value in <system> which can be modified (overridden) by a client setting that leaf to a value in <running>. But the leaf could not be deleted.

Comment 1: What if <system> contains a set of values for a leaf-list, and a client configures another set of values for that leaf-list in <running>, will the set of values in <running> completely replace the set of values in <system>? Or the two sets of values are merged together?

Comment 2: how "ordered-by user" lists and leaf-lists are merged? Do the <running> values go before or after, or is this a case where a full-replace is needed.

4.5. Examples

This section shows the examples of server-populating of <running> automatically, declaring a system-defined node in <running> explicitly, modifying a system-instantiated leaf's value and configuring descendant nodes of a system-defined node. For each example, the corresponding XML snippets are provided.

4.5.1. Server Populating of <running> Automatically

In this subsection, the following fictional module is used:

```
module example-application {
  yang-version 1.1;
  namespace "urn:example:application";
  prefix "app";

  import ietf-inet-types {
    prefix "inet";
  }
  container applications {
    list application {
      key "name";
      leaf name {
        type string;
      }
      leaf protocol {
        type enumeration {
          enum tcp;
          enum udp;
        }
      }
      leaf destination-port {
        type inet:port-number;
      }
    }
  }
}
```

The server may predefine some applications as a convenience for the clients. These predefined objects are applied only after being referenced by other configurations, which fall into the "inactive-until-referenced" system configuration as defined in Section 2. The system-instantiated application entries may be present in <system> as follows:

```
<applications xmlns="urn:example:application">
  <application>
    <name>ftp</name>
    <protocol>tcp</protocol>
    <destination-port>21</destination-port>
  </application>
  <application>
    <name>tftp</name>
    <protocol>udp</protocol>
    <destination-port>69</destination-port>
  </application>
  <application>
    <name>smtp</name>
    <protocol>tcp</protocol>
    <destination-port>25</destination-port>
  </application>
  ...
</applications>
```

The client may also define its customized applications. Suppose the configuration of applications is present in <running> as follows:

```
<applications xmlns="urn:example:application">
  <application>
    <name>my-app-1</name>
    <protocol>tcp</protocol>
    <destination-port>2345</destination-port>
  </application>
  <application>
    <name>my-app-2</name>
    <protocol>udp</protocol>
    <destination-port>69</destination-port>
  </application>
</applications>
```

A fictional ACL YANG module is used as follows, which defines a leafref for the leaf-list "application" data node to refer to an existing application name.

```
module example-acl {
  yang-version 1.1;
  namespace "urn:example:acl";
  prefix "acl";

  import example-application {
    prefix "app";
  }
  import ietf-inet-types {
    prefix "inet";
  }

  container acl {
    list acl_rule {
      key "name";
      leaf name {
        type string;
      }
      container matches {
        choice l3 {
          container ipv4 {
            leaf source_address {
              type inet:ipv4-prefix;
            }
            leaf destination_address {
              type inet:ipv4-prefix;
            }
          }
        }
        choice applications {
          leaf-list application {
            type leafref {
            path "/app:applications/app:application/app:name";
            }
          }
        }
      }
      leaf packet_action {
        type enumeration {
          enum forward;
          enum drop;
          enum redirect;
        }
      }
    }
  }
}
```

If a client configures an ACL rule referencing system predefined nodes which are not present in <running>, the client MAY issue an <edit-config> operation with the parameter "resolve-system" as follows:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <acl xmlns="urn:example:acl">
        <acl_rule>
          <name>allow_access_to_ftp_tftp</name>
          <matches>
            <ipv4>
              <source_address>198.51.100.0/24</source_address>
              <destination_address>192.0.2.0/24</destination_address>
            </ipv4>
            <application>ftp</application>
            <application>tftp</application>
            <application>my-app-1</application>
          </matches>
          <packet_action>forward</packet_action>
        </acl_rule>
      </acl>
    </config>
    <resolve-system/>
  </edit-config>
</rpc>
```

Then following gives the configuration of applications in <running> which is returned in the response to a follow-up <get-config> operation:

```
          <applications xmlns="urn:example:application">
            <application>
              <name>my-app-1</name>
              <protocol>tcp</protocol>
              <destination-port>2345</destination-port>
            </application>
            <application>
              <name>my-app-2</name>
              <protocol>udp</protocol>
              <destination-port>69</destination-port>
            </application>
            <application>
              <name>ftp</name>
            </application>
            <application>
              <name>tftp</name>
            </application>
          </applications>
```

Then the configuration of applications is present in <operational> as follows:

```
     <applications xmlns="urn:example:application"
                   xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                   or:origin="or:intended">
       <application>
         <name>my-app-1</name>
         <protocol>tcp</protocol>
         <destination-port>2345</destination-port>
       </application>
       <application>
         <name>my-app-2</name>
         <protocol>udp</protocol>
         <destination-port>69</destination-port>
       </application>
       <application or:origin="or:system">
         <name>ftp</name>
         <protocol>tcp</protocol>
         <destination-port>21</destination-port>
       </application>
       <application or:origin="or:system">
         <name>tftp</name>
         <protocol>udp</protocol>
         <destination-port>69</destination-port>
       </application>
     </applications>
```

Since the configuration of application "smtp" is not referenced by the client, it does not appear in <operational> but only in <system>.

4.5.2. Declaring a System-defined Node in <running> Explicitly

It's also possible for a client to explicitly declare the system-defined configurations that are referenced. For instance, in the above example, the client MAY also explicitly configure the following system defined applications "ftp" and "tftp" only with the list key "name" before referencing:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
     <target>
       <running/>
     </target>
     <config>
       <applications xmlns="urn:example:application">
         <application>
           <name>ftp</name>
         </application>
         <application>
           <name>tftp</name>
         </application>
       </applications>
     </config>
   </edit-config>
 </rpc>
```

Then the client issues an <edit-config> operation to configure an ACL rule referencing applications "ftp" and "tftp" without the parameter "resolve-system" as follows:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <acl xmlns="urn:example:acl">
        <acl_rule>
          <name>allow_access_to_ftp_tftp</name>
          <matches>
            <ipv4>
              <source_address>198.51.100.0/24</source_address>
              <destination_address>192.0.2.0/24</destination_address>
            </ipv4>
            <application>ftp</application>
            <application>tftp</application>
            <application>my-app-1</application>
          </matches>
          <packet_action>forward</packet_action>
        </acl_rule>
      </acl>
    </config>
  </edit-config>
</rpc>
```

Then following gives the configuration of applications in <running> which is returned in the response to a follow-up <get-config> operation, all the configuration of applications are explicitly configured by the client:

```
          <applications xmlns="urn:example:application">
            <application>
              <name>my-app-1</name>
              <protocol>tcp</protocol>
              <destination-port>2345</destination-port>
            </application>
            <application>
              <name>my-app-2</name>
              <protocol>udp</protocol>
              <destination-port>69</destination-port>
            </application>
            <application>
              <name>ftp</name>
            </application>
            <application>
              <name>tftp</name>
            </application>
          </applications>
```

Then the configuration of applications is present in <operational> as follows:

```
     <applications xmlns="urn:example:application"
                   xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                   or:origin="or:intended">
       <application>
         <name>my-app-1</name>
         <protocol>tcp</protocol>
         <destination-port>2345</destination-port>
       </application>
       <application>
         <name>my-app-2</name>
         <protocol>udp</protocol>
         <destination-port>69</destination-port>
       </application>
       <application>
         <name>ftp</name>
         <protocol or:origin="or:system">tcp</protocol>
         <destination-port or:origin="or:system">21</destination-port>
       </application>
       <application>
         <name>tftp</name>
         <protocol or:origin="or:system">udp</protocol>
         <destination-port or:origin="or:system">69</destination-port>
       </application>
     </applications>
```

Since the application names "ftp" and "tftp" are explicitly configured by the client, they take precedence as the value in <system>, the "origin" attribute will be set to "intended".

4.5.3. Modifying a System-instantiated Leaf's Value

In this subsection, we will use this fictional QoS data model:

```
module example-qos-policy {
  yang-version 1.1;
  namespace "urn:example:qos";
  prefix "qos";

  container qos-policies {
     list policy {
       key "name";
       leaf name {
       type string;
     }
       list queue {
         key "queue-id";
           leaf queue-id {
             type int32 {
               range "1..32";
             }
           }
           leaf maximum-burst-size {
             type int32 {
               range "0..100";
             }
           }
       }
     }
   }
```

Suppose a client creates a qos policy "my-policy" with 4 system instantiated queues(1~4). The Configuration of qos-policies is present in <system> as follows:

```
<qos-policies xmlns="urn:example:qos">
  <name>my-policy</name>
  <queue>
    <queue-id>1</queue-id>
    <maximum-burst-size>50</maximum-burst-size>
  </queue>
  <queue>
    <queue-id>2</queue-id>
    <maximum-burst-size>60</maximum-burst-size>
  </queue>
  <queue>
    <queue-id>3</queue-id>
    <maximum-burst-size>70</maximum-burst-size>
  </queue>
  <queue>
    <queue-id>4</queue-id>
    <maximum-burst-size>80</maximum-burst-size>
  </queue>
</qos-policies>
```

A client modifies the value of maximum-burst-size to 55 in queue-id 1:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <qos-policies xmlns="urn:example:qos">
        <name>my-policy</name>
        <queue>
          <queue-id>1</queue-id>
          <maximum-burst-size>55</maximum-burst-size>
        </queue>
      </qos-policies>
    </config>
  </edit-config>
</rpc>
```

Then the configuration of qos-policies is present in <operational> as follows:

```
        <qos-policies  xmlns="urn:example:qos"
                       xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                       or:origin="or:intended">
          <name>my-policy</name>
          <queue>
            <queue-id>1</queue-id>
            <maximum-burst-size>55</maximum-burst-size>
          </queue>
          <queue or:origin="or:system">
            <queue-id>2</queue-id>
            <maximum-burst-size>60</maximum-burst-size>
          </queue>
           <queue or:origin="or:system">
            <queue-id>3</queue-id>
            <maximum-burst-size>70</maximum-burst-size>
          </queue>
           <queue or:origin="or:system">
            <queue-id>4</queue-id>
            <maximum-burst-size>80</maximum-burst-size>
          </queue>
        </qos-policies>
```

4.5.4. Configuring Descendant Nodes of a System-defined Node

This subsection also uses the fictional interface YANG module defined in Appendix C.3 of [RFC8342]. Suppose the system provides a loopback interface (named "lo0") with a default IPv4 address of "127.0.0.1" and a default IPv6 address of "::1".

The configuration of "lo0" interface is present in <system> as follows:

```
        <interfaces>
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>
```

The configuration of "lo0" interface is present in <operational> as follows:

```
<interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
            or:origin="or:system">
  <interface>
    <name>lo0</name>
    <ip-address>127.0.0.1</ip-address>
    <ip-address>::1</ip-address>
  </interface>
</interfaces>
```

Later on, the client further configures the description node of a "lo0" interface as follows:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interfaces>
        <interface>
          <name>lo0</name>
          <description>loopback</description>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

Then the configuration of interface "lo0" is present in <operational> as follows:

```
<interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
            or:origin="or:intended">
  <interface>
    <name>lo0</name>
    <description>loopback</description>
    <ip-address or:origin="or:system">127.0.0.1</ip-address>
    <ip-address or:origin="or:system">::1</ip-address>
  </interface>
</interfaces>
```

5. The <system> Configuration Datastore

NMDA servers claiming to support this document MUST implement a <system> configuration datastore, and they SHOULD also implement the <intended> datastore.

Following guidelines for defining datastores in the appendix A of [RFC8342], this document introduces a new datastore resource named 'system' that represents the system configuration. A device MAY implement the mechanism defined in this document without implementing the "system" datastore, which would only eliminate the ability to programmatically determine the system configuration.

* Name: "system"

* YANG modules: all

* YANG nodes: all "config true" data nodes up to the root of the tree, generated by the system

* Management operations: The content of the datastore is set by the server in an implementation dependent manner. The content can not be changed by management operations via NETCONF, RESTCONF, the CLI, etc, but may change itself by upgrades and/or when resource-conditions are met. The datastore can be read using the standard NETCONF/RESTCONF protocol operations.

* Origin: This document does not define any new origin identity when it interacts with <intended> datastore and finally flows into <operational>. The "system" origin Metadata Annotation [RFC7952] is used to indicate the origin of a data item is system.

* Protocols: YANG-driven management protocols, such as NETCONF and RESTCONF.

* Defining YANG module: "ietf-system-datastore".

The datastore's content is populated by the server and read-only to clients. Upon the content is created or changed, it will be merged into <intended> datastore. Unlike <factory-default>[RFC8808], it MAY change dynamically, e.g., depending on factors like during device upgrade or system-controlled resources(e.g., HW available) and the <system> datastore does not have to persist across reboots. <factory-reset> RPC operation defined in [RFC8808] can reset it to its factory default configuration without including configuration generated due to the system update or client-enabled functionality.

6. The "ietf-system-datastore" Module

6.1. Data Model Overview

This YANG module defines a new YANG identity named "system" that uses the "ds:datastore" identity defined in [RFC8342]. A client can discover the <system> datastore support on the server by reading the YANG library information from the operational state datastore. Note that no new origin identity is defined in this document, the "or:system" origin Metadata Annotation [RFC7952] is used to indicate the origin of a data item is system. Support for the "origin" annotation is identified with the feature "origin" defined in [RFC8526].

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-system-datastore" and "ietf-datastores" YANG modules:

```
Identities:
    +--- datastore
    |  +--- conventional
    |  |  +--- running
    |  |  +--- candidate
    |  |  +--- startup
    |  |  +--- system
    |  |  +--- intended
    |  +--- dynamic
    |  +--- operational
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

6.2. Example Usage

This section gives an example of data retrieval from <system>. The YANG module used are shown in Appendix C.2 of [RFC8342]. All the messages are presented in a protocol-independent manner. JSON is used only for its conciseness.

Suppose the following data is added to <running>:

```
{
    "bgp": {
        "local-as": "64501",
        "peer-as": "64502",
        "peer": {
            "name": "2001:db8::2:3"
        }
    }
}
```

REQUEST (a <get-data> or GET request sent from the NETCONF or RESTCONF client):

Datastore: <system>
Target:/bgp

An example of RESTCONF request:

```
GET /restconf/ds/system/bgp HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

RESPONSE ("local-port" leaf value is supplied by the system):

```
{
    "bgp": {
        "peer": {
            "name": "2001:db8::2:3",
            "local-port": "60794"
        }
    }
}
```

6.3. YANG Module

```
<CODE BEGINS>
 file="ietf-system-datastore@2021-05-14.yang"
 module ietf-system-datastore {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-system-datastore";
   prefix sysds;

   import ietf-datastores {
     prefix ds;
     reference
       "RFC 8342: Network Management Datastore Architecture(NMDA)";
    }

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

    contact
      "WG Web:   <http://tools.ietf.org/wg/netmod/>
       WG List:  <mailto:netmod@ietf.org>
       Author: Qiufang Ma
               <mailto:maqiufang1@huawei.com>
       Author: Chong Feng
               <mailto:frank.fengchong@huawei.com>
```

```
          Author: Qin Wu
                  <mailto:bill.wu@huawei.com>";

       description
        "This module defines a new YANG identity that uses the
         ds:datastore identity defined in [RFC8342].

         Copyright (c) 2021 IETF Trust and the persons identified
         as authors of the code. All rights reserved.

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

         This version of this YANG module is part of RFC HHHH
         (https://www.rfc-editor.org/info/rfcHHHH); 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 2021-05-14 {
         description

           "Initial version.";
         reference
          "RFC XXXX: System-defined Configuration";
       }

       identity system {
         base ds:conventional;
         description
           "This read-only datastore contains the complete configuration
            provided by the system itself.";
       }
   }
   <CODE ENDS>
```

7. The "ietf-netconf-resolve-system" Module

This YANG module is optional to implement.

7.1. Data Model Overview

This YANG module augments NETCONF <edit-config> and <edit-data> operations with a new parameter "resolve-system" in the input parameters. If the "resolve-system" parameter is present, the server will populate the referenced system configuration into target datastore automatically. A NETCONF client can discover the "resolve-system" parameter support on the server by checking the YANG library information with "ietf-netconf-resolve-system" included from the operational state datastore.

Comment: How does a RESTCONF client know if the RESTCONF server implements the "resolve-system" parameter?

The following tree diagram [RFC8340] illustrates the "ietf-netconf-resolve-system" module:

```
module: ietf-netconf-resolve-system
  augment /nc:edit-config/nc:input:
    +---w resolve-system?   empty
  augment /ncds:edit-data/ncds:input:
    +---w resolve-system?   empty
```

The following tree diagram [RFC8340] illustrates "edit-config" and "edit-data" rpcs defined in "ietf-netconf" and "ietf-netconf-nmda" respectively, augmented by "ietf-netconf-resolve-system" YANG module :

```
rpcs:
  +---x edit-config
  |  +---w input
  |     +---w target
  |     |  +---w (config-target)
  |     |     +--:(candidate)
  |     |     |  +---w candidate?   empty {candidate}?
  |     |     +--:(running)
  |     |        +---w running?     empty {writable-running}?
  |     +---w default-operation?   enumeration
  |     +---w test-option?         enumeration {validate}?
  |     +---w error-option?        enumeration
  |     +---w (edit-content)
  |     |   +--:(config)
  |     |   |  +---w config?        <anyxml>
  |     |   +--:(url)
  |     |      +---w url?           inet:uri {url}?
  |     +---w resolve-system?      empty
  +---x edit-data
     +---w input
        +---w datastore             ds:datastore-ref
        +---w default-operation?    enumeration
        +---w (edit-content)
        |  +--:(config)
        |  |  +---w config?         <anydata>
        |  +--:(url)
        |     +---w url?            inet:uri {nc:url}?
        +---w resolve-system?       empty
```

7.2. Example Usage

This section gives an example of an <edit-config> request to reference system-defined data nodes which are not present in <running> with a "resolve-system" parameter. A retrieval of <running> to show the auto-populated referenced system objects after the <edit-config> request is also given. The YANG module used is shown as follows, leafrefs refer to an existing name and address of an interface:

```
module example-interface-management {
  yang-version 1.1;
  namespace "urn:example:interfacemgmt";
  prefix "inm";

  container interfaces {
    list interface {
      key name;
      leaf name {
        type string;
      }
      leaf description {
        type string;
      }
      leaf mtu {
        type uint16;
      }
      leaf ip-address {
        type inet:ip-address;
      }
    }
  }
  container default-address {
    leaf ifname {
      type leafref {
        path "../../interfaces/interface/name";
      }
    }
    leaf address {
      type leafref {
        path "../../interfaces/interface[name = current()/../ifname]"
           + "/ip-address";
      }
    }
  }
}
```

Image that the system provides a loopback interface (named "lo0") with a predefined MTU value of "1500" and a predefined IP address of "127.0.0.1". The <system> datastore shows the following configuration of loopback interface:

```
   <interfaces xmlns="urn:example:interfacemgmt">
     <interface>
       <name>lo0</name>
       <mtu>1500</mtu>
       <ip-address>127.0.0.1</ip-address>
     </interface>
   </interfaces>
```

The client sends an <edit-config> operation to add the configuration of default-address with a "resolve-system" parameter:

```
  <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
    <edit-config>
      <target>
        <running/>
      </target>
      <config>
        <default-address xmlns="urn:example:interfacemgmt">
          <if-name>lo0</if-name>
          <address>127.0.0.1</address>
        </default-address>
      </config>
    </edit-config>
    <resolve-system/>
  </rpc>
```

Since the "resolve-system" parameter is provided, the server will resolve any leafrefs to system configurations and copy the referenced system-defined nodes into <running> automatically with the same value (i.e., the name and ip-address data nodes of lo0 interface) in <system> at the end of <edit-config> operation constraint enforcement. After the processing, a positive resonse is returned:

```
   <rpc-reply message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <ok/>
   </rpc-reply>
```

Then the client sends a <get-config> operation towards <running>:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <interfaces xmlns="urn:example:interfacemgmt"/>
    </filter>
  </get-config>
</rpc>
```

Given that the referenced interface "name" and "ip-address" of lo0 are populated by the server, the following response is returned:

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <interfaces xmlns="urn:example:interfacemgmt">
      <interface>
        <name>lo0</name>
        <ip-address>127.0.0.1</ip-address>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

7.3. YANG Module

```
<CODE BEGINS>
 file="ietf-netconf-resolve-system@2021-05-14.yang"
 module ietf-netconf-resolve-system {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-resolve-system";
    prefix ncrs;

    import ietf-netconf {
      prefix nc;
      reference
        "RFC 6241: Network Configuration Protocol (NETCONF)";
    }

    import ietf-netconf-nmda {
      prefix ncds;
      reference
        "RFC 8526: NETCONF Extensions to Support the Network
         Management Datastore Architecture";
    }
```

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

      contact
        "WG Web:   <http://tools.ietf.org/wg/netmod/>
         WG List:  <mailto:netmod@ietf.org>
         Author: Qiufang Ma
                 <mailto:maqiufang1@huawei.com>
         Author: Chong Feng
                 <mailto:frank.fengchong@huawei.com>
         Author: Qin Wu
                 <mailto:bill.wu@huawei.com>";

      description
        "This module defines an extension to the NETCONF protocol
         that allows the NETCONF client to control whether the server
         is allowed to populate referenced system configuration
         automatically without the client doing so explicitly.

          Copyright (c) 2021 IETF Trust and the persons identified
          as authors of the code. All rights reserved.

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

          This version of this YANG module is part of RFC HHHH
          (https://www.rfc-editor.org/info/rfcHHHH); 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 2021-05-14 {
        description
          "Initial version.";
        reference
          "RFC XXXX: System-defined Configuration";
      }

    augment /nc:edit-config/nc:input {
```

```
       description
         "Allows the server to automatically populate
          referenced system configuration to make configuration
          valid.";
        leaf resolve-system {
          type empty ;
          description
            "When present, the server is allowed to automatically
             populate referenced system configuration into <running>.";
           }
      }

     augment /ncds:edit-data/ncds:input {
       description
         "Allows the server to automatically populate
          referenced system configuration to make configuration
          valid.";
        leaf resolve-system {
          type empty ;
          description
            "When present, the server is allowed to automatically
             populate referenced system configuration into <running>.";
           }
      }
    }
   <CODE ENDS>
```

8. IANA Considerations

8.1. The "IETF XML" Registry

This document registers two XML namespace URNs in the 'IETF XML registry', following the format defined in [RFC3688].

```
   URI: urn:ietf:params:xml:ns:yang:ietf-system-datastore
   Registrant Contact: The IESG.
   XML: N/A, the requested URIs are XML namespaces.

   URI: urn:ietf:params:xml:ns:yang:ietf-netconf-resolve-system
   Registrant Contact: The IESG.
   XML: N/A, the requested URIs are XML namespaces.
```

8.2. The "YANG Module Names" Registry

This document registers two module names in the 'YANG Module Names' registry, defined in [RFC6020] .

```
name: ietf-system-datastore
prefix: sys
namespace: urn:ietf:params:xml:ns:yang:ietf-system-datatstore
RFC: XXXX // RFC Ed.: replace XXXX and remove this comment

name: ietf-netconf-resolve-system
prefix: ncrs
namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-resolve-system
RFC: XXXX // RFC Ed.: replace XXXX and remove this comment
```

9. Security Considerations

9.1. Regarding the "ietf-system-datastore" YANG Module

The YANG module defined in this document extends the base operations for NETCONF [RFC6241] and RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

9.2. Regarding the "ietf-netconf-resolve-system" YANG Module

The YANG module defined in this document extends the base operations for NETCONF [RFC6241] and [RFC8526]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

The security considerations for the base NETCONF protocol operations (see Section 9 of [RFC6241] apply to the new extended RPC operations defined in this document.

10. Contributors

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf@chinatelecom.cn

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Acknowledgements

Thanks to Robert Wilton, Balazs Lengyel, Andy Bierman, Juergen Schoenwaelder, Alex Clemm, Martin Bjorklund, Timothy Carey for reviewing, and providing important input to, this document.

References

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/info/rfc2119>.

Informative References

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

[RFC8808] Wu, Q., Lengyel, B., and Y. Niu, "A YANG Data Model for Factory Default Settings", RFC 8808, DOI 10.17487/RFC8808, August 2020, <https://www.rfc-editor.org/info/rfc8808>.

Appendix A. Key Use Cases

Following provides three use cases related to system-defined configuration lifecycle management. The simple interface data model defined in Appendix C.3 of [RFC8342] is used. For each use case, snippets of <running>, <system>, <intended> and <operational> are shown.

A.1. Device Powers On

<running>:

No configuration for lo0 appears in <running>;

<system>:

```
     <interfaces>
       <interface>
         <name>lo0</name>
         <ip-address>127.0.0.1</ip-address>
         <ip-address>::1</ip-address>
       </interface>
     </interfaces>
```

<intended>:

```
     <interfaces>
       <interface>
         <name>lo0</name>
         <ip-address>127.0.0.1</ip-address>
         <ip-address>::1</ip-address>
       </interface>
     </interfaces>
```

```
   <operational>:

        <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                    or:origin="or:system">
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>
```

A.2. Client Commits Configuration

If a client creates an interface "et-0/0/0" but the interface does not physically exist at this point:

```
   <running>:

        <interfaces>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
          </interface>
        </interfaces>

   <system>:

        <interfaces>
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>

   <intended>:

        <interfaces>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
          </interface>
          <interface>
        </interfaces>
```

```
   <operational>:

        <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                    or:origin="or:intended">
          <interface or:origin="or:system">
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>
```

A.3. Operator Installs Card into a Chassis

```
   <running>:

        <interfaces>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
          </interface>
        </interfaces>

   <system>:

        <interfaces>
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>et-0/0/0</name>
            <mtu>1500</mtu>
          </interface>
        </interfaces>

   <intended>:
```

```
        <interfaces>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
            <mtu>1500</mtu>
          </interface>
          <interface>
        </interfaces>
```

<operational>:

```
        <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                    or:origin="or:intended">
          <interface or:origin="or:system">
            <name or:origin>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
         <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
            <mtu or:origin="or:system">1500</mtu>
          </interface>
          <interface>
        </interfaces>
```

Appendix B. Changes between Revisions

v00 - v01

* Remove the "with-system" parameter to retrieve <running> with system configuration merged in.

* Add a new parameter named "resolve-system" to allow the server to populate referenced system configuration into <running> automatically in order to make <running> valid.

* Usage examples refinement.

v02 - v00

* Restructure the document content based on input in the system defined configuration interim meeting.

* Updates NMDA to define a read-only conventional configuration datastore called "system".

* Retrieval of implicit hidden system configuration via <get><get-config> with "with-system" parameter to support non-NMDA servers.

* Provide system defined configuration classification.

* Define Static Characteristics and dynamic behavior for system defined configuration.

* Separate "ietf-system-datastore" Module from "ietf-netconf-with-system" Module.

* Provide usage examples for dynamic behaviors.

* Provide usage examples for two YANG modules.

* Provide three use cases related to system-defined configuration lifecycle management.

* Classify the relation with <factory-default>.

Appendix C. Open Issues tracking

* Should the "with-origin" parameter be supported for <intended>?

Authors' Addresses

Qiufang Ma (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: maqiufang1@huawei.com

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Qin Wu
Huawei
101 Software Avenue, Yuhua District

Nanjing
Jiangsu, 210012
China

Email: bill.wu@huawei.com

Feng Chong
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: frank.fengchong@huawei.com

Jan Lindblad
Cisco Systems

Email: jlindbla@cisco.com