

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2019

C. Hopps
LabN Consulting, L.L.C.
March 2, 2019

YANG Geo Location
draft-chopps-netmod-geo-location-01

Abstract

This document defines a generic geographical location object YANG grouping. The geographical location grouping is intended to be used in YANG models for specifying a location on or in reference to the Earth or any other astronomical object.

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 September 3, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
2. The Geo Location Object	3
2.1. Frame of Reference	3
2.2. Location	4
2.3. Motion	4
2.4. Nested Locations	5
2.5. Non-location Attributes	5
2.6. Tree	5
3. YANG Module	6
4. ISO 6709:2008 Conformance	11
5. Usability	12
5.1. Portability	12
5.1.1. IETF URI Value	12
5.1.2. W3C	12
5.1.3. Geography Markup Language (GML)	14
5.1.4. KML	15
6. IANA Considerations	16
6.1. Geodetic System Value Registry	16
7. Security Considerations	17
8. References	17
8.1. Normative References	17
8.2. Informative References	18
Appendix A. Examples	19
Appendix B. Acknowledgements	22
Author's Address	22

1. Introduction

In many applications we would like to specify the location of something geographically. Some examples of locations in networking might be the location of data center, a rack in an internet exchange point, a router, a firewall, a port on some device, or it could be the endpoints of a fiber, or perhaps the failure point along a fiber.

Additionally, while this location is typically relative to The Earth, it does not need to be. Indeed it is easy to imagine a network or device located on The Moon, on Mars, on Enceladus (the moon of Saturn) or even a comet (e.g., 67p/churyumov-gerasimenko).

Finally, one can imagine defining locations using different frames of reference or even alternate systems (e.g., simulations or virtual realities).

This document defines a "geo-location" YANG grouping that allows for all of the above data to be captured.

This specification conforms to [ISO.6709.2008].

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

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 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. The Geo Location Object

2.1. Frame of Reference

The frame of reference ("reference-frame") defines what the location values refer to and their meaning. The referred to object can be any astronomical body. It could be a planet such as The Earth or Mars, a moon such as Enceladus, an asteroid such as Ceres, or even a comet such as 1P/Halley. This value is specified in "astronomical-body" and is defined by the International Astronomical Union (<<http://www.iau.org>>), The default "astronomical-body" value is "earth".

In addition to identifying the astronomical body we also need to define the meaning of the coordinates (e.g., latitude and longitude) and the definition of 0-height. This is done with a "geodetic-datum" value. The default value for "geodetic-datum" is "wgs-84" (i.e., the World Geodetic System, [WGS84]), which is used by the Global Positioning System (GPS) among many others. We define an IANA registry for specifying standard values for the "geodetic-datum".

In addition to the "geodetic-datum" value we allow refining the coordinate and height accuracy using "coord-accuracy" and "height-accuracy" respectively. When specified these values override the defaults implied by the "geodetic-datum" value.

Finally, we define an optional feature which allows for changing the system for which the above values are defined. This optional feature adds an "alternate-system" value to the reference frame. This value is normally not present which implies the natural universe is the system. The use of this value is intended to allow for creating virtual realities or perhaps alternate coordinate systems. The definition of alternate systems is outside the scope of this document.

2.2. Location

This is the location on or relative to the astronomical object. It is specified using 2 or 3 coordinates values. These values are given either as "latitude", "longitude", and an optional "height", or as Cartesian coordinates of "x", "y" and an optional "z". For the standard location choice "latitude" and "longitude" are specified as fractions of decimal degrees, and the "height" value is in fractions of meters. For the Cartesian choice "x", "y" and "z" are in fractions of meters. In both choices the exact meanings of all of the values are defined by the "geodetic-datum" value in the Section 2.1.

2.3. Motion

Support is added for objects in relatively stable motion. For objects in relatively stable motion the grouping provides a 3-dimensional vector value. The components of the vector are "v-north", "v-east" and "v-up" which are all given in fractional meters per second. The values "v-north" and "v-east" are relative to true-north as defined by the reference frame for the astronomical body, "v-up" is perpendicular to the plane defined by "v-north" and "v-east", and is pointed away from the center of mass.

To derive the 2-dimensional heading and speed one would use the following formulas:

$$\text{speed} = \sqrt{v_{\text{north}}^2 + v_{\text{east}}^2}$$

$$\text{heading} = \arctan(v_{\text{east}} / v_{\text{north}})$$

For some applications that demand high accuracy, and where the data is infrequently updated this velocity vector can track very slow movement such as continental drift.

Tracking more complex forms of motion is outside the scope of this work. The intent of the grouping being defined here is to identify where something is located, and generally this is expected to be somewhere on or relative to the Earth (or another astronomical body). At least two options are available to YANG models that wish to use this grouping with objects that are changing location frequently in non-simple ways, they can add additional motion data to their model directly, or if the application allows it can require more frequent queries to keep the location data current.

2.4. Nested Locations

When locations are nested (e.g., a building may have a location which houses routers that also have locations) the module using this grouping is free to indicate in its definition that the "reference-frame" is inherited from the containing object so that the "reference-frame" need not be repeated in every instance of location data.

2.5. Non-location Attributes

During the development of this module, the question of whether it would support data such as orientation arose. These types of attributes are outside the scope of this grouping because they do not deal with a location but rather describe something more about the object that is at the location. Module authors are free to add these non-location attributes along with their use of this location grouping.

2.6. Tree

The following is the YANG tree diagram [RFC8340] for the geo-location grouping.

```
module: geo-location
  +-- geo-location
    +-- reference-frame
      +-- alternate-system?    string {alternate-systems}?
      +-- astronomical-body?  string
      +-- geodetic-system
        +-- geodetic-datum?    string
        +-- coord-accuracy?    decimal64
        +-- height-accuracy?   decimal64
    +-- (location)
      +--:(ellipsoid)
        +-- latitude           degrees
        +-- longitude          degrees
        +-- height?            decimal64
      +--:(cartesian)
        +-- x                  decimal64
        +-- y                  decimal64
        +-- z?                 decimal64
    +-- velocity
      +-- v-north?             decimal64
      +-- v-east?              decimal64
      +-- v-up?                decimal64
    +-- timestamp?             types:date-and-time
```

3. YANG Module

```
<CODE BEGINS> file "ietf-geo-location@2019-02-17.yang"
module ietf-geo-location {
  namespace "urn:ietf:params:xml:ns:yang:ietf-geo-location";
  prefix geo;
  import ietf-yang-types { prefix types; }

  organization
    "IETF NETMOD Working Group (NETMOD)";
  contact
    "Christian Hopps <chopps@chopps.org>";

  // RFC Ed.: replace XXXX with actual RFC number and
  // remove this note.

  description
    "This module defines a grouping of a container object for
    specifying a location on or around an astronomical object (e.g.,
    The Earth).

    Copyright (c) 2018 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).

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

  // RFC Ed.: replace XXXX with actual RFC number and
  // remove this note.

  revision 2019-02-17 {
    description "Initial Revision";
    reference "RFC XXXX: YANG Geo Location";
  }
```

```
typedef degrees {
    type decimal64 {
        fraction-digits 16;
    }
    units "decimal degrees";
    description "Coordinate value.";
}

feature alternate-systems {
    description
        "This feature means the device supports specifying locations
        using alternate systems for reference frames.";
}

grouping geo-location {
    description
        "Grouping to identify a location on an astronomical object.";

    container geo-location {
        description
            "A location on an astronomical body (e.g., The Earth)
            somewhere in a universe.";

        container reference-frame {
            description
                "The Frame of Reference for the location values.";

            leaf alternate-system {
                if-feature alternate-systems;
                type string;
                description
                    "The system in which the astronomical body and
                    geodetic-datum is defined. Normally, this value is not
                    present and the system is the natural universe; however,
                    when present this value allows for specifying alternate
                    systems (e.g., virtual realities). An alternate-system
                    modifies the definition (but not the type) of the other
                    values in the reference frame.";
            }
        }
        leaf astronomical-body {
            type string {
                pattern
                    '[-0-9a-z #x22#x23#x5B#x5D' +
                    ' !$%&()*+,\./:;<=>?@\^_`{|}~]+';
            }
            default "earth";
            description
                "An astronomical body as named by the International
```

```

    Astronomical Union (IAU) or according to the alternate
    system if specified. Examples include 'sun' (our star),
    'earth' (our planet), 'moon' (our moon), 'enceladus' (a
    moon of Saturn), 'ceres' (an asteroid),
    '67p/churyumov-gerasimenko (a comet). The value should
    be comprised of all lower case ASCII characters not
    including control characters (i.e., values 32..64, and
    91..126)";
}
container geodetic-system {
  description
    "The geodetic system of the location data.";
  leaf geodetic-datum {
    type string {
      pattern
        '[-0-9a-z#x22#x23#x5B#x5D' +
        ' !$%&()*+,\./:;<=>?@\\^_`{|}~]+';
    }
    default "wgs-84";
    description
      "A geodetic-datum defining the meaning of latitude,
      longitude and height. The default is 'wgs-84' which is
      used by the Global Positioning System (GPS)";
  }
  leaf coord-accuracy {
    type decimal64 {
      fraction-digits 6;
    }
    description
      "The accuracy of the latitude longitude pair. When
      coord-accuracy is specified it overrides the
      geodetic-datum implied accuracy. If Cartesian
      coordinates are in use this accuracy corresponds to
      the X and Y components";
  }
  leaf height-accuracy {
    type decimal64 {
      fraction-digits 6;
    }
    units "meters";
    description
      "The accuracy of height value. When specified it
      overrides the geodetic-datum implied default. If
      Cartesian coordinates are in use this accuracy
      corresponds to the Z component.";
  }
}
// May wish to allow for height to be relative.
// If so need to decide if we have a boolean (to ground)

```



```
// or an enumeration (e.g., local ground, sea-floor,
// ground floor, containing object, ...) or even allow
// for a string for most generic but least portable
// comparable
// leaf height-relative {
// }
}
}
choice location {
  mandatory true;
  description
    "The location data either in lat/long or Cartesian values";
  case ellipsoid {
    leaf latitude {
      type degrees;
      mandatory true;
      description
        "The latitude value on the astronomical body. The
        definition and precision of this measurement is
        indicated by the reference-frame.";
    }
    leaf longitude {
      type degrees;
      mandatory true;
      description
        "The longitude value on the astronomical body. The
        definition and precision of this measurement is
        indicated by the reference-frame.";
    }
  }
  leaf height {
    type decimal64 {
      fraction-digits 6;
    }
    units "meters";
    description
      "Height from a reference 0 value. The precision and '0'
      value is defined by the reference-frame.";
  }
}
case cartesian {
  leaf x {
    type decimal64 {
      fraction-digits 6;
    }
    mandatory true;
    description
      "The X value as defined by the reference-frame.";
  }
}
```

```
    leaf y {
      type decimal64 {
        fraction-digits 6;
      }
      mandatory true;
      description
        "The Y value as defined by the reference-frame.";
    }
    leaf z {
      type decimal64 {
        fraction-digits 6;
      }
      units "meters";
      description
        "The Z value as defined by the reference-frame.";
    }
  }
}
container velocity {
  description
    "If the object is in motion the velocity vector describes
    this motion at the the time given by the timestamp.";

  leaf v-north {
    type decimal64 {
      fraction-digits 12;
    }
    units "meters per second";
    description
      "v-north is the rate of change (i.e., speed) towards
      truth north as defined by the ~geodetic-system~.";
  }

  leaf v-east {
    type decimal64 {
      fraction-digits 12;
    }
    units "meters per second";
    description
      "v-east is the rate of change (i.e., speed) perpendicular
      to truth-north as defined by the ~geodetic-system~.";
  }

  leaf v-up {
    type decimal64 {
      fraction-digits 12;
    }
    units "meters per second";
  }
}
```

```

        description
          "v-up is the rate of change (i.e., speed) away from the
           center of mass.";
      }
    }
    leaf timestamp {
      type types:date-and-time;
      description "Reference time when location was recorded.";
    }
  }
}
}
<CODE ENDS>

```

4. ISO 6709:2008 Conformance

[ISO.6709.2008] provides an appendix with a set of tests for conformance to the standard. The tests and results are given in the following table along with an explanation of non-applicable tests.

Test	Description	Pass Explanation
A.1.2.1	elements reqd. for a geo. point location	CRS is always indicated
A.1.2.2	Description of a CRS from a register	CRS register is defined
A.1.2.3	definition of CRS	N/A - Don't define CRS
A.1.2.4	representation of horizontal position	lat/long values conform
A.1.2.5	representation of vertical position	height value conforms
A.1.2.6	text string representation	N/A - No string format

Conformance Test Results

For test "A.1.2.1" the YANG geo location object either includes a CRS ("reference-frame") or has a default defined ([WGS84]).

For "A.1.2.3" we do not define our own CRS, and doing so is not required for conformance.

For "A.1.2.6" we do not define a text string representation, which is also not required for conformance.

5. Usability

The geo-location object defined in this document and YANG module have been designed to be usable in a very broad set of applications. This includes the ability to locate things on astronomical bodies other than The Earth, and to utilize entirely different coordinate systems and realities.

Many systems make use of geo-location data, and so it's important to be able describe this data using this geo-location object defined in this document.

5.1. Portability

In order to verify portability while developing this module the following standards and standard APIs and were considered.

5.1.1. IETF URI Value

[RFC5870] defines a standard URI value for geographic location data. It includes the ability to specify the "geodetic-value" (it calls this "crs") with the default being "wgs-84" [WGS84]. For the location data it allows 2 to 3 coordinates defined by the "crs" value. For accuracy it has a single "u" parameter for specifying uncertainty. The "u" value is in fractions of meters and applies to all the location values. As the URI is a string, all values are specified as strings and so are capable of as much precision as required.

URI values can be mapped to and from the YANG grouping, with the caveat that some loss of precision (in the extremes) may occur due to the YANG grouping using decimal64 values rather than strings.

5.1.2. W3C

See <<https://w3c.github.io/geolocation-api/#dom-geolocationposition>>.

W3C Defines a geo-location API in [W3CGEO]. We show a snippet of code below which defines the geo-location data for this API. This is used by many application (e.g., Google Maps API).

```

interface GeolocationPosition {
  readonly attribute GeolocationCoordinates coords;
  readonly attribute DOMTimeStamp timestamp;
};

interface GeolocationCoordinates {
  readonly attribute double latitude;
  readonly attribute double longitude;
  readonly attribute double? altitude;
  readonly attribute double accuracy;
  readonly attribute double? altitudeAccuracy;

  readonly attribute double? speed;
};

```

5.1.2.1. Compare with YANG Model

Field	Type	YANG	Type
accuracy	double	coord-accuracy	dec64 fr 6
altitude	double	height	dec64 fr 6
altitudeAccuracy	double	height-accuracy	dec64 fr 6
heading	double	heading	dec64 fr 16
latitude	double	latitude	dec64 fr 16
longitude	double	longitude	dec64 fr 16
speed	double	speed	dec64 fr 12
timestamp	DOMTimeStamp	timestamp	string

accuracy (double): Accuracy of "latitude" and "longitude" values in meters.

altitude (double): Optional height in meters above the [WGS84] ellipsoid.

altitudeAccuracy (double): Optional accuracy of "altitude" value in meters.

heading (double): Optional Direction in decimal deg from true north increasing clock-wise.

latitude, longitude (double): Standard lat/long values in decimal degrees.

speed (double): Speed along heading in meters per second.

timestamp (DOMTimeStamp): Specifies milliseconds since the Unix EPOCH in 64 bit unsigned integer. The YANG model defines the timestamp with arbitrarily large precision by using a string which encompasses all representable values of this timestamp value.

W3C API values can be mapped to the YANG grouping, with the caveat that some loss of precision (in the extremes) may occur due to the YANG grouping using decimal64 values rather than doubles.

Conversely, only YANG values for The Earth using the default "wgs-84" [WGS84] as the "geodetic-datum", can be directly mapped to the W3C values, as W3C does not provide the extra features necessary to map the broader set of values supported by the YANG grouping.

5.1.3. Geography Markup Language (GML)

ISO adopted the Geography Markup Language (GML) defined by OGC 07-036 as [ISO.19136.2007]. GML defines, among many other things, a position type "gml:pos" which is a sequence of "double" values. This sequence of values represent coordinates in a given CRS. The CRS is either inherited from containing elements or directly specified as attributes "srsName" and optionally "srsDimension" on the "gml:pos".

GML defines an Abstract CRS type which Concrete CRS types derive from. This allows for many types of CRS definitions. We are concerned with the Geodetic CRS type which can have either ellipsoidal or Cartesian coordinates. We believe that other non-Earth based CRS as well as virtual CRS should also be representable by the GML CRS types as well.

Thus GML "gml:pos" values can be mapped directly to the YANG grouping, with the caveat that some loss of precision (in the extremes) may occur due to the YANG grouping using decimal64 values rather than doubles.

Conversely, YANG grouping values can be mapped to GML as directly as the GML CRS available definitions allow with a minimum of Earth-based geodetic systems fully supported.

GML also defines an observation value in "gml:Observation" which includes a timestamp value "gml:validTime" in addition to other components such as "gml:using" "gml:target" and "gml:resultOf". Only the timestamp is mappable to and from the YANG grouping. Furthermore

"gml:validTime" can either be an Instantaneous measure ("gml:TimeInstant") or a time period ("gml:TimePeriod"). Only the instantaneous "gml:TimeInstant" is mappable to and from the YANG grouping.

5.1.4. KML

KML 2.2 [KML22] (formerly Keyhole Markup Language) was submitted by Google to Open Geospatial Consortium (OGC) <<https://www.opengeospatial.org/>> and was adopted. The latest version as of this writing is KML 2.3 [KML23]. This schema includes geographic location data in some of its objects (e.g., <kml:Point or <kml:Camera> objects). This data is provided in string format and corresponds to the [W3CGEO] values. The timestamp value is also specified as a string as in our YANG grouping.

KML has some special handling for the height value useful for visualization software, "kml:altitudeMode". These values for "kml:altitudeMode" include indicating the height is ignored ("clampToGround"), in relation to the locations ground level ("relativeToGround"), or in relation to the geodetic datum ("absolute"). The YANG grouping can directly map the ignored and absolute cases, but not the relative to ground case.

In addition to the "kml:altitudeMode" KML also defines two seafloor height values using "kml:seaFloorAltitudeMode". One value is to ignore the height value ("clampToSeaFloor") and the other is relative ("relativeToSeaFloor"). As with the "kml:altitudeMode" value, the YANG grouping supports the ignore case but not the relative case.

The KML location values use a geodetic datum defined in Annex A by the GML Coordinate Reference System (CRS) [ISO.19136.2007] with identifier "LonLat84_5773". The altitude value for KML absolute height mode is measured from the vertical datum specified by [WGS84].

Thus the YANG grouping and KML values can be directly mapped in both directions (when using a supported altitude mode) with the caveat that some loss of precision (in the extremes) may occur due to the YANG grouping using decimal64 values rather than strings. For the relative height cases the application doing the transformation is expected to have the data available to transform the relative height into an absolute height which can then be expressed using the YANG grouping.

6. IANA Considerations

6.1. Geodetic System Value Registry

This registry allocates names for standard geodetic systems. Often these values are referred to using multiple names (e.g., full names or multiple acronyms values). The intent of this registry is to provide a single standard value for any given geodetic system.

The values SHOULD use an acronym when available, they MUST be converted to lower case, and spaces MUST be changed to dashes "-".

Each entry should be sufficient to define the 3 coordinate values (2 if height is not required). So for example the "wgs-84" is defined as WGS-84 with the geoid updated by at least [EGM96] for height values. Specific entries for [EGM96] and [EGM08] are present if a more precise definition of the data is required.

It should be noted that [RFC5870] also creates a registry for Geodetic Systems (it calls CRS); however, this registry has a very strict modification policy. The authors of [RFC5870] have the stated goal of making CRS registration hard to avoid proliferation of CRS values. As our module defines alternate systems and has a broader (beyond earth) scope, the registry defined below is meant to be more easily modified.

TODO: Open question, should we create a new registry here or attempt to modify the one created by [RFC5870]. It's worth noting that we include the ability to specify any geodetic system including ones designed for astronomical bodies other than the earth, as well as ones based on alternate systems. These requirements may be too broad for adapting the existing [RFC5870] registry.

TODO: Open question, is FCFS too easy, perhaps expert review would strike a good balance. If expert review is acceptable, would it also be acceptable to update the policy on [RFC5870] and use it instead?

The allocation policy for this registry is First Come First Served, [RFC8126] as the intent is simply to avoid duplicate values.

The initial values for this registry are as follows.

Name	Description
me	Mean Earth/Polar Axis (Moon)
mola-vik-1	MOLA Height, IAU Viking-1 PM (Mars)
wgs-84-96	World Geodetic System 1984 [WGS84] w/ EGM96
wgs-84-08	World Geodetic System 1984 [WGS84] w/ [EGM08]
wgs-84	World Geodetic System 1984 [WGS84] (EGM96 or better)

7. Security Considerations

This document defines a common geo location grouping using the YANG data modeling language. The grouping itself has no security or privacy impact on the Internet, but the usage of the grouping in concrete YANG modules might have. The security considerations spelled out in the YANG 1.1 specification [RFC7950] apply for this document as well.

8. References

8.1. Normative References

- [EGM08] Pavlis, N., Holmes, S., Kenyon, S., and J. Factor, "An Earth Gravitational Model to Degree 2160: EGM08.", 2008, <http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm2008/egm08_wgs84.html>.
- [EGM96] Lemoine, F., Kenyon, S., Factor, J., Trimmer, R., Pavlis, N., Chinn, D., Cox, C., Klosko, S., Luthcke, S., Torrence, M., Wang, Y., Williamson, R., Pavlis, E., Rapp, R., and T. Olson, "The Development of the Joint NASA GSFC and the National Imagery and Mapping Agency (NIMA) Geopotential Model EGM96.", 1998, <<https://cddis.nasa.gov/926/egm96/egm96.html>>.
- [ISO.6709.2008] International Organization for Standardization, "ISO 6709:2008 Standard representation of geographic point location by coordinates.", 2008.

- [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>>.
- [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>>.
- [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>>.
- [WGS84] National Imagery and Mapping Agency., "National Imagery and Mapping Agency Technical Report 8350.2, Third Edition.", 1 2000, <<http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf>>.

8.2. Informative References

- [ISO.19136.2007] International Organization for Standardization, "ISO 19136:2007 Geographic information -- Geography Markup Language (GML)".
- [KML22] Wilson, T., Ed., "OGC KML (Version 2.2)", 4 2008, <http://portal.opengeospatial.org/files/?artifact_id=27810>.
- [KML23] Burggraf, D., Ed., "OGC KML 2.3", 8 2015, <<http://docs.opengeospatial.org/is/12-007r2/12-007r2.html>>.
- [RFC5870] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Locations ('geo' URI)", RFC 5870, DOI 10.17487/RFC5870, June 2010, <<https://www.rfc-editor.org/info/rfc5870>>.
- [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>>.

- [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>>.
- [W3CGEO] Popescu, A., "Geolocation API Specification", 11 2016, <<https://www.w3.org/TR/2016/REC-geolocation-API-20161108/>>.

Appendix A. Examples

Below is a fictitious module that uses the geo-location grouping.

```
<CODE BEGINS> file "ietf-uses-geo-location@2019-02-02.yang"
module iETF-uses-geo-location {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-uses-geo-location";
  prefix ugeo;
  import geo-location { prefix geo; }
  organization "Empty Org";
  contact "Example Author <eauthor@example.com>";
  description "Example use of geo-location";
  revision 2019-02-02 { reference "None"; }
  container locatable-items {
    description "container of locatable items";
    list locatable-item {
      key name;
      description "A of locatable item";
      leaf name {
        type string;
        description "name of locatable item";
      }
      uses geo:geo-location;
    }
  }
}
<CODE ENDS>
```

Below is a the YANG tree for the fictitious module that uses the geo-location grouping.

```
module: ietf-uses-geo-location
  +--rw locatable-items
    +--rw locatable-item* [name]
      +--rw name                string
      +--rw geo-location
        +--rw reference-frame
          +--rw alternate-system?  string {alternate-systems}?
          +--rw astronomical-body? string
          +--rw geodetic-system
            +--rw geodetic-datum?  string
            +--rw coord-accuracy?  decimal64
            +--rw height-accuracy? decimal64
        +--rw (location)
          +--:(ellipsoid)
            +--rw latitude    degrees
            +--rw longitude   degrees
            +--rw height?     decimal64
          +--:(cartesian)
            +--rw x           decimal64
            +--rw y           decimal64
            +--rw z?         decimal64
        +--rw velocity
          +--rw v-north?      decimal64
          +--rw v-east?       decimal64
          +--rw v-up?        decimal64
        +--rw timestamp?     types:date-and-time
```

Below is some example YANG XML data for the fictitious module that uses the geo-location grouping.

```
<ns0:config xmlns:ns0="urn:ietf:params:xml:ns:netconf:base:1.0">
  <locatable-items
    xmlns="urn:ietf:params:xml:ns:yang:ietf-uses-geo-location">
    <locatable-item>
      <name>Gaetana's</name>
      <geo-location>
        <latitude>40.73297</latitude>
        <longitude>-74.007696</longitude>
      </geo-location>
    </locatable-item>
    <locatable-item>
      <name>Pont des Arts</name>
      <geo-location>
        <timestamp>2012-03-31T16:00:00Z</timestamp>
        <latitude>48.8583424</latitude>
        <longitude>2.3375084</longitude>
        <height>35</height>
      </geo-location>
    </locatable-item>
    <locatable-item>
      <name>Saint Louis Cathedral</name>
      <geo-location>
        <timestamp>2013-10-12T15:00:00-06:00</timestamp>
        <latitude>29.9579735</latitude>
        <longitude>-90.0637281</longitude>
      </geo-location>
    </locatable-item>
    <locatable-item>
      <name>Apollo 11 Landing Site</name>
      <geo-location>
        <timestamp>1969-07-21T02:56:15Z</timestamp>
        <reference-frame>
          <astronomical-body>moon</astronomical-body>
          <geodetic-system>
            <geodetic-datum>me</geodetic-datum>
          </geodetic-system>
        </reference-frame>
        <latitude>0.67409</latitude>
        <longitude>23.47298</longitude>
      </geo-location>
    </locatable-item>
  </locatable-items>
</ns0:config>
```

Appendix B. Acknowledgements

We would like to thank Peter Lothberg for the motivation as well as help in defining a more broadly useful geographic location object.

We would also like to thank Acee Lindem and Qin Wu for their work on a geographic location object that led to this documents creation.

Author's Address

Christian Hopps
LabN Consulting, L.L.C.

Email: chopps@chopps.org

NETMOD Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: September 11, 2019

K. Watsen
Watsen Networks
Q. Wu
Huawei Technologies
A. Farrel
Old Dog Consulting
B. Claise
Cisco Systems, Inc.
March 10, 2019

Handling Long Lines in Inclusions in Internet-Drafts and RFCs
draft-ietf-netmod-artwork-folding-01

Abstract

This document introduces a simple and yet time-proven strategy for handling long lines in inclusions in drafts using a backslash ('\\') character where line-folding has occurred. The strategy works on any text-based content, but is primarily intended for a structured sequence of lines, such as would be referenced by the <sourcecode> element defined in Section 2.48 of RFC 7991, rather than for two-dimensional imagery, such as would be referenced by the <artwork> element defined in Section 2.5 of RFC 7991. The approach produces consistent results, regardless of the content, that is both self-documenting and enables automated reconstitution of the original content.

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 September 11, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Applicability Statement	3
3. Requirements Language	4
4. Goals	4
4.1. Automated Folding of Long Lines in Text Content	4
4.2. Automated Reconstitution of the Original Text Content	4
5. Limitations	4
5.1. Not Recommended for Graphical Artwork	5
5.2. Doesn't Work as Well as Format-Specific Options	5
6. Folded Structure	5
6.1. Header	6
6.2. Body	6
7. Algorithm	6
7.1. Automated Folding	6
7.1.1. Manual Folding	7
7.2. Automated Unfolding	8
8. Examples	8
8.1. Simple Example Showing Boundary Conditions	9
8.2. Example Showing Multiple Wraps of a Single Line	9
8.3. Example With Native Backslash	10
8.4. Example With Native Whitespace	10
8.5. Example of Manual Wrapping	10
9. Security Considerations	13
10. IANA Considerations	13
11. References	13
11.1. Normative References	13
11.2. Informative References	13
Appendix A. POSIX Shell Script	15
Acknowledgements	19
Authors' Addresses	20

1. Introduction

[RFC7994] sets out the requirements for plain-text RFCs and states that each line of an RFC (and hence of an Internet-Draft) must be limited to 72 characters followed by the character sequence that denotes an end-of-line (EOL).

Internet-Drafts and RFCs often include example text or code fragments. In order to render the formatting of such text it is usually presented as a figure using the "<sourcecode>" element in the source XML. Many times the example text or code exceeds the 72 character line-length limit and the 'xml2rfc' utility does not attempt to wrap the content of such inclusions, simply issuing a warning whenever lines exceed 69 characters. According to the RFC Editor, there is currently no convention in place for how to handle long lines, other than advising authors to clearly indicate what manipulation has occurred.

This document introduces a simple and yet time-proven strategy for handling long lines in inclusions in drafts using a backslash ('\') character where line-folding has occurred. The strategy works on any text based inclusion, but is primarily intended for a structured sequence of lines, such as would be referenced by the <sourcecode> element defined in Section 2.48 of [RFC7991], rather than for two-dimensional imagery, such as would be referenced by the <artwork> element defined in Section 2.5 of [RFC7991]. The approach produces consistent results, regardless of the content, that is both self-documenting and enables automated reconstitution of the original content.

Note that text files are represent as lines having their first character in column 1, and a line length of N where the last character is in the Nth column and is immediately followed by an end of line character sequence.

2. Applicability Statement

The format and algorithm defined in this document may be used in any context, whether for IETF documents or in other situations where structured folding is desired.

Within the IETF, this work is primarily targeted to xml2rfc v3 <sourcecode> element (Section 2.48 of [RFC7991]) and xml2rfc v2 <artwork> element (Section 2.5 of [RFC7749]) that, for lack of a better option, is currently used for both source code and artwork. This work may be also be used for the xml2rfc v3 <artwork> element (Section 2.5 of [RFC7991]) but, as described in Section 5.1, it is generally not recommended.

3. 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.

4. Goals

4.1. Automated Folding of Long Lines in Text Content

Automated folding of long lines is needed in order to support draft compilations that entail a) validation of source input files (e.g., XML, JSON, ABNF, ASN.1) and/or b) dynamic generation of output, using a tool that doesn't observe line lengths, that is stitched into the final document to be submitted.

Generally, in order for tooling to be able to process input files, the files must be in their original/natural state, which may include having some long lines. Thus, these source files need to be modified before inclusion in the document in order to satisfy the line length limits. This modification SHOULD be automated to reduce effort and errors resulting from manual effort.

Similarly, dynamically generated output (e.g., tree diagrams) must also be modified, if necessary, in order for the resulting document to satisfy the line length limits. When needed, this effort again SHOULD be automated to reduce effort and errors resulting from manual effort.

4.2. Automated Reconstitution of the Original Text Content

Automated reconstitution of the original content is needed to support validation of artwork extracted from documents. YANG [RFC7950] modules are already extracted from Internet-Drafts and validated as part of the draft-submission process. Additionally, there has been some discussion regarding needing to do the same for instance examples (i.e., XML/JSON documents) contained within Internet-Drafts ([yang-doctors-thread]). Thus, it SHOULD be possible to mechanically reconstitute the original text content in order to satisfy tooling input parsers.

5. Limitations

5.1. Not Recommended for Graphical Artwork

While the solution presented in this document will work on any kind of text-based content, it is most useful on content that represents source code (XML, JSON, etc.) or, more generally, on content that has not been laid out in two dimensions (e.g., diagrams).

Fundamentally, the issue is whether the text content remains readable once folded. Text content that is unpredictable is especially susceptible to looking bad when folded; falling into this category are most UML diagrams, YANG tree diagrams, and ASCII art in general.

It is NOT RECOMMENDED to use the solution presented in this document on graphical artwork.

5.2. Doesn't Work as Well as Format-Specific Options

The solution presented in this document works generically for all text-based content, as it only views content as plain text. However, various formats sometimes have built-in mechanisms that are better suited to prevent long lines.

For instance, both the 'pyang' and 'yanglint' utilities have the command line option "--tree-line-length" that can be used to indicate a desired maximum line length for when generating tree diagrams [RFC8340].

In another example, some source formats (e.g., YANG [RFC7950]) allow any quoted string to be broken up into substrings separated by a concatenation character (e.g., '+'), any of which can be on a different line.

In yet another example, some languages allow factoring blocks of code into call outs, such as functions. Using such call outs is especially helpful when in some deeply-nested code, as they typically reset the indentation back to the first column.

It is RECOMMENDED that authors do as much as possible within the selected format to avoid long lines.

6. Folded Structure

Text content that has been folded as specified by this document MUST contain the following structure.

6.1. Header

The header is two lines long.

The first line is the following 46-character string that MAY be surrounded by any number of printable characters. This first line cannot itself be folded.

NOTE: '\\\' line wrapping per BCP XX (RFC XXXX)

[Note to RFC Editor: Please replace XX and XXXX with the numbers assigned to this document and delete this note. Please make this change in multiple places in this document.]

The second line is a blank line. This line provides visual separation for readability.

6.2. Body

The character encoding is the same as described in Section 2 of [RFC7994], except that, per [RFC7991], tab characters are prohibited.

Lines that have a backslash ('\\') occurring as the last character in a line immediately followed by the end of line character sequence, when the subsequent line starts with a backslash ('\\') as the first non-space (' ') character, are considered "folded".

Really long lines may be folded multiple times.

7. Algorithm

This section describes the processes for folding and unfolding long lines when they are encountered in a single instance of text content. It is assumed that another process inserts/extracts the individual text content instances to/from an Internet-Draft or RFC. For example, the 'xiaz' utility [xiaz] does just this.

7.1. Automated Folding

Determine the desired maximum line length from input to the automated line-wrapping process, such as from a command line parameter. If no value is explicitly specified, the value "69" SHOULD be used.

Ensure that the desired maximum line length is not less than the minimum header, which is 46 characters. If the desired maximum line length is less than this minimum, exit (this text-based content cannot be folded).

Scan the text content for horizontal tab characters. If any horizontal tab characters appear, either resolve them to space characters or exit, forcing the input provider to convert them to space characters themselves first.

Scan the text content to see if any line exceeds the desired maximum. If no line exceeds the desired maximum, exit (this text content does not need to be folded).

Scan the text content to ensure no existing lines already end with a backslash ('\\') character when the subsequent line starts with a backslash ('\\') character as the first non-space (' ') character, as this would lead to an ambiguous result. If such a line is found, exit (this text content cannot be folded).

If this text content needs to and can be folded, insert the header as described in Section 6.1.

For each line in the text content, from top-to-bottom, if the line exceeds the desired maximum, then fold the line at the desired maximum column by 1) inserting the character backslash ('\\') character at the maximum column, 2) inserting the end of line character sequence, inserting any number of space (' ') characters, and 4) inserting a further backslash ('\\') character.

The result of this previous operation is that the next line starts with an arbitrary number of space (' ') characters, followed by a backslash ('\\') character, immediately followed by the character that was previously in the maximum column.

Continue in this manner until reaching the end of the text content. Note that this algorithm naturally addresses the case where the remainder of a folded line is still longer than the desired maximum, and hence needs to be folded again, ad infinitum.

The process described in this section is illustrated by the "fold_it()" function in Appendix A.

7.1.1. Manual Folding

Authors may choose to fold text examples and source code by hand to produce a text content that is more pleasant for a human reader but which can still be automatically unfolded (as described in Section 7.2) to produce single lines that are longer than the maximum document line length.

For example, an author may choose to make the fold at convenient gaps between words such that the backslash is placed in a lower column number than the text content's maximum column value.

Additionally, an author may choose to indent the start of a continuation line by inserting space characters before the line continuation marker backslash character.

Manual folding may also help handle the cases that cannot be automatically folded as described in Section 7.

Authors MUST produce a result that adheres to the structure described in Section 6.

7.2. Automated Unfolding

All unfolding is assumed to be automated although a reader will mentally perform the act of unfolding the text to understand the true nature of the original text content.

Scan the beginning of the text content for the header described in Section 6.1. If the header is not present, starting on the first line of the text content, exit (this artwork does not need to be unfolded).

Remove the 2-line header from the text content.

For each line in the text content, from top-to-bottom, if the line has a backslash ('\\') character immediately followed by the end of line character sequence, and if the next line has a backslash ('\\') character as the first non-space (' ') character, then the lines can be unfolded. Remove the first backslash ('\\') character, the end of line character sequence, any leading space (' ') characters, and the second backslash ('\\') character, which will bring up the next line. Then continue to scan each line in the text content starting with the current line (in case it was multiply folded).

Continue in this manner until reaching the end of the text content.

The process described in this section is illustrated by the "unfold_it()" function in Appendix A.

8. Examples

The following self-documenting examples illustrate folded text-based content.

The source text content cannot be presented here, as it would again need to be folded. Alas, only the result can be provided.

The examples in Sections 8.1 through 8.4 were automatically folded on column 69, the default value. Section 8.5 shows an example of manual folding.

8.1. Simple Example Showing Boundary Conditions

This example illustrates a boundary condition test using numbers for counting purposes. The input contains 5 lines, each line one character longer than the previous.

Any printable character (including ' ' and '\') can be used as a substitute for any number, except for on the 4th row, the trailing '9' is not allowed to be a '\' character if the first non-space character of the next line is a '\' character, as that would lead to an ambiguous result.

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```
12345678901234567890123456789012345678901234567890123456
123456789012345678901234567890123456789012345678901234567
1234567890123456789012345678901234567890123456789012345678
12345678901234567890123456789012345678901234567890123456789
1234567890123456789012345678901234567890123456789012345678\
\90
1234567890123456789012345678901234567890123456789012345678\
\901
1234567890123456789012345678901234567890123456789012345678\
\9012
```

8.2. Example Showing Multiple Wraps of a Single Line

This example illustrates one very long line (280 characters).

Any printable character (including ' ' and '\') can be used as a substitute for any number.

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```
1234567890123456789012345678901234567890123456789012345678\
\901234567890123456789012345678901234567890123456789012345\
\678901234567890123456789012345678901234567890123456789012\
\345678901234567890123456789012345678901234567890123456789\
\01234567890
```

8.3. Example With Native Backslash

This example has a '\' character in the wrapping column. The native text includes the sequence "fish\fowl" with the '\' character occurring on the 69th column.

```
string1="The quick brown dog jumps over the lazy dog which is a fish\
\\fowl as appropriate"
```

8.4. Example With Native Whitespace

This example has whitespace spanning the wrapping column. The native input contains 15 space (' ') characters between "like" and "white".

===== NOTE: '\ ' line wrapping per BCP XX (RFC XXXX) =====

```
Sometimes our strings include multiple spaces such as "We like      \
\           white space."
```

8.5. Example of Manual Wrapping

This example was manually wrapped to cause the folding to occur after each term, putting each term on its own line. Indentation is used to additionally improve readability. Also note that the mandatory header is surrounded by different printable characters than shown in the other examples.

===== NOTE: '\ ' line wrapping per BCP XX (RFC XXXX) =====

```
<yang-library
  xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"
  xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">

  <module-set>
    <name>config-modules</name>
    <module>
      <name>ietf-interfaces</name>
      <revision>2018-02-20</revision>
      <namespace>\
        \urn:ietf:params:xml:ns:yang:ietf-interfaces\
      </namespace>
    </module>
    <module>
      <name>ietf-ip</name>
      <revision>2018-02-22</revision>
      <namespace>\
        \urn:ietf:params:xml:ns:yang:ietf-ip\
```



```

    \</namespace>
</module>
<import-only-module>
  <name>ietf-yang-types</name>
  <revision>2013-07-15</revision>
  <namespace>\
    \urn:ietf:params:xml:ns:yang:ietf-yang-types\
  </namespace>
</import-only-module>
<import-only-module>
  <name>ietf-inet-types</name>
  <revision>2013-07-15</revision>
  <namespace>\
    \urn:ietf:params:xml:ns:yang:ietf-inet-types\
  </namespace>
</import-only-module>
</module-set>

<schema>
  <name>config-schema</name>
  <module-set>config-modules</module-set>
</schema>
<schema>
  <name>state-schema</name>
  <module-set>config-modules</module-set>
  <module-set>state-modules</module-set>
</schema>

<datastore>
  <name>ds:startup</name>
  <schema>config-schema</schema>
</datastore>
<datastore>
  <name>ds:running</name>
  <schema>config-schema</schema>
</datastore>
<datastore>
  <name>ds:operational</name>
  <schema>state-schema</schema>
</datastore>

  <content-id>75a43df9bd56b92aacc156a2958fbe12312fb285</content-id>
</yang-library>

```

The manual folding produces a more readable result than the following equivalent folding that contains no indentation.

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```
<yang-library
  xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"
  xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">

  <module-set>
    <name>config-modules</name>
    <module>
      <name>ietf-interfaces</name>
      <revision>2018-02-20</revision>
      <namespace>urn:ietf:params:xml:ns:yang:ietf-interfaces</namesp\
\ace>
    </module>
    <module>
      <name>ietf-ip</name>
      <revision>2018-02-22</revision>
      <namespace>urn:ietf:params:xml:ns:yang:ietf-ip</namespace>
    </module>
    <import-only-module>
      <name>ietf-yang-types</name>
      <revision>2013-07-15</revision>
      <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-types</namesp\
\ace>
    </import-only-module>
    <import-only-module>
      <name>ietf-inet-types</name>
      <revision>2013-07-15</revision>
      <namespace>urn:ietf:params:xml:ns:yang:ietf-inet-types</namesp\
\ace>
    </import-only-module>
  </module-set>

  <schema>
    <name>config-schema</name>
    <module-set>config-modules</module-set>
  </schema>
  <schema>
    <name>state-schema</name>
    <module-set>config-modules</module-set>
    <module-set>state-modules</module-set>
  </schema>

  <datastore>
    <name>ds:startup</name>
    <schema>config-schema</schema>
  </datastore>
  <datastore>
    <name>ds:running</name>
    <schema>config-schema</schema>
```

```
</datastore>
<datastore>
  <name>ds:operational</name>
  <schema>state-schema</schema>
</datastore>

<content-id>75a43df9bd56b92aacc156a2958fbe12312fb285</content-id>
</yang-library>
```

9. Security Considerations

This BCP has no Security Considerations.

10. IANA Considerations

This BCP has no IANA Considerations.

11. References

11.1. 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>>.
- [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>>.

11.2. Informative References

- [RFC7749] Reschke, J., "The "xml2rfc" Version 2 Vocabulary", RFC 7749, DOI 10.17487/RFC7749, February 2016, <<https://www.rfc-editor.org/info/rfc7749>>.
- [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>>.
- [RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <<https://www.rfc-editor.org/info/rfc7991>>.
- [RFC7994] Flanagan, H., "Requirements for Plain-Text RFCs", RFC 7994, DOI 10.17487/RFC7994, December 2016, <<https://www.rfc-editor.org/info/rfc7994>>.

[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>>.

[xiaz] "The 'xiaz' Python Package",
<<https://pypi.org/project/xiaz/>>.

[yang-doctors-thread]
"[yang-doctors] automating yang doctor reviews",
<<https://mailarchive.ietf.org/arch/msg/yang-doctors/DCfBqgfZPAD7afzeDF1Q1Xm2X3g>>.

Appendix A. POSIX Shell Script

This non-normative appendix section includes a shell script that can both fold and unfold text content. Note that this script is applied only to single text content instances.

```
#!/bin/bash

print_usage() {
    echo
    echo "Folds the text file, only if needed, at the specified"
    echo "column, according to BCP XX."
    echo
    echo "Usage: $0 [-c <col>] [-r] -i <infile> -o <outfile>"
    echo
    echo "  -c: column to fold on (default: 69)"
    echo "  -r: reverses the operation"
    echo "  -i: the input filename"
    echo "  -o: the output filename"
    echo "  -d: show debug messages"
    echo "  -h: show this message"
    echo
    echo "Exit status code: zero on success, non-zero otherwise."
    echo
}

# global vars, do not edit
debug=0
reversed=0
infile=""
outfile=""
maxcol=69 # default, may be overridden by param
hdr_txt="NOTE: '\\\\' line wrapping per BCP XX (RFC XXXX)"
equal_chars="====="
space_chars=" "

fold_it() {
    # since upcomming tests are >= (not >)
    testcol=`expr "$maxcol" + 1`

    # check if file needs folding
    grep ".\{$testcol\}" $infile >> /dev/null 2>&1
    if [ $? -ne 0 ]; then
        if [[ $debug -eq 1 ]]; then
            echo "nothing to do"
        fi
        cp $infile $outfile
    fi
}
```

```

    return -1
fi

foldcol=`expr "$maxcol" - 1` # for the inserted '\ ' char

# ensure input file doesn't contain a TAB
grep $'\t' $infile >> /dev/null 2>&1
if [ $? -eq 0 ]; then
    echo
    echo "Error: infile contains a TAB character, which is not"
    echo "allowed."
    echo
    return 1
fi

# ensure input file doesn't contain the fold-sequence already
pcgrep -M "\\n[\ ]*\\n" $infile >> /dev/null 2>&1
if [ $? -eq 0 ]; then
    echo
    echo "Error: infile has a line ending with a '\ ' character"
    echo "    followed by a '\ ' character as the first non-space"
    echo "    character on the next line. This file cannot be"
    echo "    folded."
    echo
    return 1
fi

# center header text
length=`expr ${#hdr_txt} + 2`
left_sp=`expr \( "$maxcol" - "$length" \) / 2`
right_sp=`expr "$maxcol" - "$length" - "$left_sp"`
header=`printf "%.s %s %.s" "$left_sp" "$equal_chars"\
               "$hdr_txt" "$right_sp" "$equal_chars"`

# fold using recursive passes ('g' didn't work)
if [ -z "$1" ]; then
    # init recursive env
    cp $infile /tmp/wip
fi
gsed "/.{${testcol}}/s/\(.{${foldcol}}\)/\1\\n\\n/" < /tmp/wip\
    >> /tmp/wip2
diff /tmp/wip /tmp/wip2 > /dev/null 2>&1
if [ $? -eq 1 ]; then
    mv /tmp/wip2 /tmp/wip
    fold_it "recursing"
else
    echo "$header" > $outfile
    echo "" >> $outfile

```

```

    cat /tmp/wip2 >> $outfile
    rm /tmp/wip*
fi

## following two lines represent a non-functional variant to the
## recursive logic presented in the block above. It used to work
## before the '\' on the next line was added to the format (i.e.,
## the trailing '\\\\' in the substitution below), but now there
## is an off-by-one error. Leaving here in case anyone can fix it.
#echo "$header" > $outfile
#echo "" >> $outfile
#gsed "/.\{$testcol\}/s/\(.\{$foldcol\}\)/\1\\\\\n\\\\/g"\
    < $infile >> $outfile

return 0
}

unfold_it() {
    # check if file needs unfolding
    line='head -n 1 $infile | fgrep "$hdr_txt"'
    if [ $? -ne 0 ]; then
        if [[ $debug -eq 1 ]]; then
            echo "nothing to do"
        fi
        cp $infile $outfile
        return -1
    fi
    # output all but the first two lines (the header) to wip (work
    # in progress) file
    awk "NR>2" $infile > /tmp/wip

    # unfold wip file
    gsed ":x; /.*\n$/N; s/\\n[ ]*\\n//; tx; s/\t/g" /tmp/wip\
        > $outfile

    # clean up and return
    rm /tmp/wip
    return 0
}

process_input() {
    while [ "$1" != "" ]; do
        if [ "$1" == "-h" -o "$1" == "--help" ]; then
            print_usage
            exit 1
        fi
    done
}

```

```
fi
if [ "$1" == "-d" ]; then
    debug=1
fi
if [ "$1" == "-c" ]; then
    maxcol="$2"
    shift
fi
if [ "$1" == "-r" ]; then
    reversed=1
fi
if [ "$1" == "-i" ]; then
    infile="$2"
    shift
fi
if [ "$1" == "-o" ]; then
    outfile="$2"
    shift
fi
shift
done

if [ -z "$infile" ]; then
    echo
    echo "Error: infile parameter missing (use -h for help)"
    echo
    exit 1
fi

if [ -z "$outfile" ]; then
    echo
    echo "Error: outfile parameter missing (use -h for help)"
    echo
    exit 1
fi

if [ ! -f "$infile" ]; then
    echo
    echo "Error: specified file \"$infile\" is does not exist."
    echo
    exit 1
fi

min_supported=`expr ${#hdr_txt} + 8`
if [ $maxcol -lt $min_supported ]; then
    echo
    echo "Error: the folding column cannot be less than"
    echo "$min_supported"
```



```
        echo
        exit 1
    fi

    max_supported=`expr ${#equal_chars} + 1 + ${#hdr_txt} + 1\
        + ${#equal_chars}`
    if [ $maxcol -gt $max_supported ]; then
        echo
        echo "Error: the folding column cannot be more than"
        echo "$max_supported"
        echo
        exit 1
    fi
}

main() {
    if [ "$#" == "0" ]; then
        print_usage
        exit 1
    fi

    process_input $@

    if [[ $reversed -eq 0 ]]; then
        fold_it
        code=$?
    else
        unfold_it
        code=$?
    fi
    exit $code
}

main "$@"
```

Acknowledgements

The authors thank the following folks for their various contributions (sorted by first name): Gianmarco Bruno, Italo Busi, Jonathan Hansford, Joel Jaeggli, Lou Berger, Martin Bjorklund, Italo Busi, and Rob Wilton.

The authors additionally thank the RFC Editor for confirming that there is no set convention today for handling long lines in artwork/sourcecode inclusions.

Authors' Addresses

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net

Qin Wu
Huawei Technologies

EMail: bill.wu@huawei.com

Adrian Farrel
Old Dog Consulting

EMail: adrian@olddog.co.uk

Benoit Claise
Cisco Systems, Inc.

EMail: bclaise@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2019

A. Bierman
YumaWorks
M. Bjorklund
Cisco
K. Watsen
Watsen Networks
March 8, 2019

YANG Data Structure Extensions
draft-ietf-netmod-yang-data-ext-02

Abstract

This document describes YANG mechanisms for defining abstract data structures with YANG. It is intended to replace and extend the "yang-data" extension statement defined in RFC 8040.

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 <http://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 September 9, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
1.1.1. NMDA	3
1.1.2. YANG	3
2. Definitions	4
3. YANG Data Structure Extensions Module	4
4. IANA Considerations	9
4.1. YANG Module Registry	9
5. Security Considerations	9
6. References	9
6.1. Normative References	9
6.2. Informative References	10
Appendix A. Examples	10
A.1. "structure" Example	10
A.2. "augment-structure" Example	11
A.3. XML Encoding Example	12
A.4. JSON Encoding Example	12
Appendix B. Change Log	13
B.1. v01 to v02	13
B.2. v00 to v01	13
Appendix C. Open Issues	13
Authors' Addresses	13

1. Introduction

There is a need for standard mechanisms to allow the definition of abstract data that is not intended to be implemented as configuration or operational state. The "yang-data" extension statement from RFC 8040 [RFC8040] was defined for this purpose but it is limited in its functionality.

The intended use of the "yang-data" extension was to model all or part of a protocol message, such as the "errors" definition in the YANG module "ietf-restconf" [RFC8040], or the contents of a file. However, protocols are often layered such that the header or payload portions of the message can be extended by external documents. The YANG statements that model a protocol need to support this extensibility that is already found in that protocol.

The "yang-data" extension from [RFC8040] has been copied here, renamed to "structure", and updated to be more flexible. There is no assumption that a YANG data structure can only be used as a top-level abstraction, instead of nested within some other data structure.

This document also defines a new YANG extension statement called "augment-structure", which allows abstract data structures to be augmented from external modules, similar to the existing YANG "augment" statement. Note that "augment" cannot be used to augment a YANG data structure since a YANG compiler or other tool is not required to understand the "structure" extension.

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 used within this document:

- o YANG data structure: A data structure defined with the "structure" statement.

1.1.1. NMDA

The following terms are defined in the Network Management Datastore Architecture (NMDA) [RFC8342]. and are not redefined here:

- o configuration
- o operational state

1.1.2. YANG

The following terms are defined in [RFC7950]:

- o absolute-schema-nodeid
- o container
- o data definition statement
- o data node
- o leaf
- o leaf-list
- o list

2. Definitions

A YANG Data Structure is defined with the "structure" extension statement, defined in the YANG module "ietf-yang-structure-ext". The argument to the "structure" extension statement is the name of the data structure. The data structures are considered to be in the same identifier namespace as defined in section 6.2.1 of [RFC7950]. In particular, bullet 7:

All leafs, leaf-lists, lists, containers, choices, rpcs, actions, notifications, anydatas, and anyxmls defined (directly or through a "uses" statement) within a parent node or at the top level of the module or its submodules share the same identifier namespace.

This means that data structures defined with the "structure" statement cannot have the same name as sibling nodes from regular YANG data definition statements or other "structure" statements in the same YANG module.

This does not mean a YANG data structure has to be used as a top-level protocol message or other top-level data structure.

3. YANG Data Structure Extensions Module

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-yang-structure-ext@2019-03-07.yang"

```
module ietf-yang-structure-ext {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-structure-ext";
  prefix sx;

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Author:   Andy Bierman
               <mailto:andy@yumaworks.com>

     Author:   Martin Bjorklund
               <mailto:mbj@tail-f.com>

     Author:   Kent Watsen
               <mailto:kent+ietf@watsen.net>";
```

description

"This module contains conceptual YANG specifications for defining abstract data structures.

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

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

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.

```
revision 2019-03-07 {  
  description  
    "Initial revision.";  
  // RFC Ed.: replace XXXX with RFC number and remove this note  
  reference  
    "RFC XXXX: YANG Structure Extensions.";  
}
```

```
extension structure {  
  argument name {  
    yin-element true;  
  }  
  description  
    "This extension is used to specify a YANG data structure that  
    represents conceptual data defined in YANG. It is intended to  
    describe hierarchical data independent of protocol context or  
    specific message encoding format. Data definition statements  
    within a 'structure' extension statement specify the generic  
    syntax for the specific YANG data structure, whose name is the  
    argument of the 'structure' extension statement.  
  
    Note that this extension does not define a media-type. A  
    specification using this extension MUST specify the message  
    encoding rules, including the content media type, if  
    applicable.
```

The mandatory 'name' parameter value identifies the YANG data structure that is being defined.

This extension is only valid as a top-level statement, i.e., given as a sub-statement to 'module' or 'submodule'.

The sub-statements of this extension MUST follow the ABNF rules below, where the rules are defined in RFC 7950:

```
*must-stmt
[status-stmt]
[description-stmt]
[reference-stmt]
*(typedef-stmt / grouping-stmt)
*data-def-stmt
```

A YANG data structure defined with this extension statement is encoded in the same way as an 'anydata' statement. This means that the name of the structure is encoded as a 'container', with the instantiated child statements encoded as child nodes to this node.

The module name and namespace value for the YANG module using the extension statement is assigned to each of the data definition statements resulting from the YANG data structure.

The XPath document element is the extension statement itself, such that the child nodes of the document element are represented by the data-def-stmt sub-statements within this extension. This conceptual document is the context for the following YANG statements:

- must-stmt
- when-stmt
- path-stmt
- min-elements-stmt
- max-elements-stmt
- mandatory-stmt
- unique-stmt
- ordered-by
- instance-identifier data type

The following data-def-stmt sub-statements are constrained when used within a 'structure' extension statement.

- The list-stmt is not required to have a key-stmt defined.
- The config-stmt is ignored if present.

";


```
}  
  
extension augment-structure {  
  argument path {  
    yin-element true;  
  }  
  description  
    "This extension is used to specify an augmentation to YANG data  
    structure defined with the 'structure' statement. It is  
    intended to describe hierarchical data independent of protocol  
    context or specific message encoding format.
```

This statement has almost the same structure as the 'augment-stmt'. Data definition statements within this statement specify the semantics and generic syntax for the additional data to be added to the specific YANG data structure, identified by the 'path' argument.

The mandatory 'path' parameter value identifies the YANG conceptual data node that is being augmented, represented as an absolute-schema-nodeid string, where the first node in the absolute-schema-nodeid string identifies the YANG data structure to augment, and the rest of the nodes in the string identifies the node within the YANG structure to augment.

This extension is only valid as a top-level statement, i.e., given as a sub-statement to 'module' or 'submodule'.

The sub-statements of this extension MUST follow the ABNF rules below, where the rules are defined in RFC 7950:

```
[status-stmt]  
[description-stmt]  
[reference-stmt]  
1*(data-def-stmt / case-stmt)
```

The module name and namespace value for the YANG module using the extension statement is assigned to instance document data conforming to the data definition statements within this extension.

The XPath document element is the augmented extension statement itself, such that the child nodes of the document element are represented by the data-def-stmt sub-statements within the augmented 'structure' statement.

The context node of the 'augment-structure' statement is derived in the same way as the 'augment' statement, as defined

in section 6.4.1 of [RFC7950]. This conceptual node is considered the context node for the following YANG statements:

- must-stmt
- when-stmt
- path-stmt
- min-elements-stmt
- max-elements-stmt
- mandatory-stmt
- unique-stmt
- ordered-by
- instance-identifier data type

The following data-def-stmt sub-statements are constrained when used within an 'augment-structure' extension statement.

- The list-stmt is not required to have a key-stmt defined.
- The config-stmt is ignored if present.

Example:

```
module foo {
  import ietf-yang-structure-ext { prefix sx; }

  sx:structure foo-data {
    container foo-con { }
  }
}

module bar {
  import ietf-yang-structure-ext { prefix sx; }
  import foo { prefix foo; }

  sx:augment-structure /foo:foo-data/foo:foo-con {
    leaf add-leaf1 { type int32; }
    leaf add-leaf2 { type string; }
  }
}
";
}
}
<CODE ENDS>
```

4. IANA Considerations

4.1. YANG Module Registry

This document registers one URI as a namespace in the "IETF XML Registry" [RFC3688]:

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

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]:

```
name:          ietf-yang-structure-ext
namespace:     urn:ietf:params:xml:ns:yang:ietf-yang-structure-ext
prefix:        sx
// RFC Ed.: replace XXXX with RFC number and remove this note
reference:     RFC XXXX
```

5. Security Considerations

This document defines YANG extensions that are used to define conceptual YANG data structures. It does not introduce any new vulnerabilities beyond those specified in YANG 1.1 [RFC7950].

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>>.
- [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>>.

6.2. Informative References

- [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>>.

Appendix A. Examples

A.1. "structure" Example

This example shows a simple address book that could be stored as an artifact.

```
module example-module {
  yang-version 1.1;
  namespace "urn:example:example-module";
  prefix exm;

  import ietf-yang-structure-ext {
    prefix sx;
  }

  sx:structure address-book {
    list address {
      key "last first";
      leaf last {
        type string;
        description "Last name";
      }
      leaf first {
        type string;
        description "First name";
      }
      leaf street {
        type string;
        description "Street name";
      }
      leaf city {
        type string;
        description "City name";
      }
      leaf state {
        type string;
        description "State name";
      }
    }
  }
}
```

A.2. "augment-structure" Example

This example adds "county" and "zipcode" leafs to the address book:

```
module example-module-aug {  
  yang-version 1.1;  
  namespace "urn:example:example-module-aug";  
  prefix exma;  
  
  import ietf-yang-structure-ext {  
    prefix sx;  
  }  
  import example-module {  
    prefix exm;  
  }  
  
  sx:augment-structure "/exm:address-book/exm:address" {  
    leaf county {  
      type string;  
      description "County name";  
    }  
    leaf zipcode {  
      type string;  
      description "Postal zipcode";  
    }  
  }  
}
```

A.3. XML Encoding Example

This example shows how an address book can be encoded in XML:

```
<address-book xmlns="urn:example:example-module">  
  <address>  
    <last>Flintstone</last>  
    <first>Fred</first>  
    <street>301 Cobblestone Way</street>  
    <city>Bedrock</city>  
    <zipcode xmlns="urn:example:example-module-aug">70777</zipcode>  
  </address>  
</address-book>
```

A.4. JSON Encoding Example

This example shows how an address book can be encoded in JSON:

```
"example-module:address-book": {  
  "address": [  
    {  
      "city": "Bedrock",  
      "example-module-aug:zipcode": "70777",  
      "first": "Fred",  
      "last": "Flintstone",  
      "street": "301 Cobblestone Way"  
    }  
  ]  
}
```

Appendix B. Change Log

RFC Ed.: remove this section before publication.

B.1. v01 to v02

- o terminology fixes (use the term "structure" instead of "template")
- o renamed the statement to "structure" from "yang-data"
- o removed limitations on if-feature and identities in YANG structures

B.2. v00 to v01

- o moved open issues to github
- o added examples section
- o filled in IANA considerations

Appendix C. Open Issues

RFC Ed.: remove this section before publication.

The YANG Data Structure Extensions issues are tracked on github.com:

<https://github.com/netmod-wg/yang-data-ext/issues>

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Cisco

Email: mbj@tail-f.com

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Netmod
Internet-Draft
Intended status: Standards Track
Expires: August 30, 2019

B. Lengyel
Ericsson
B. Claise
Cisco Systems, Inc.
February 26, 2019

YANG Instance Data File Format
draft-ietf-netmod-yang-instance-file-format-02

Abstract

There is a need to document data defined in YANG models when a live YANG server is not available. Data is often needed already at design or implementation time or needed by groups that do not have a live running YANG server available. This document specifies a standard file format for YANG instance data (which follows the syntax and semantic from existing YANG models, re-using the same format as the reply to a <get> operation/request) and decorates it with metadata.

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 August 30, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology	2
2. Introduction	3
2.1. High Level Principles	4
3. Instance Data File Format	4
3.1. Specifying the Target YANG Modules: target-ptr	6
3.1.1. INLINE Method	7
3.1.2. URI Method	8
3.2. Examples	8
4. Data Life cycle	12
5. Delivery of Instance Data	13
6. Backwards Compatibility	13
7. YANG Model	13
8. Security Considerations	17
9. IANA Considerations	17
9.1. URI Registration	17
9.2. YANG Module Name Registration	17
10. Acknowledgments	17
11. References	18
11.1. Normative References	18
11.2. Informative References	19
Appendix A. Open Issues	19
Appendix B. Changes between revisions	19
Appendix C. Detailed Use Cases - Non-Normative	21
C.1. Use Cases	21
C.1.1. Use Case 1: Early Documentation of Server Capabilities	22
C.1.2. Use Case 2: Preloading Data	23
C.1.3. Use Case 3: Documenting Factory Default Settings	23
Authors' Addresses	23

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 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

Instance Data Set: A named set of data items decorated with metadata that can be used as instance data in a YANG data tree.

Instance Data File: A file containing an instance data set formatted according to the rules described in this document.

Target YANG Module: A YANG module for which the instance data set contains instance data, like `ietf-yang-library` in the examples.

YANG Instance Data, or just instance data for short, is data that could be stored in a datastore and whose syntax and semantics is defined by YANG models.

2. Introduction

There is a need to document data defined in YANG models when a live YANG server is not available. Data is often needed already at design or implementation time or needed by groups that do not have a live running YANG server available. To facilitate this off-line delivery of data this document specifies a standard format for YANG instance data sets and YANG instance data files.

The following is a list of already implemented and potential use cases.

- UC1 Documentation of server capabilities
- UC2 Preloading default configuration data
- UC3 Documenting Factory Default Settings
- UC4 Instance data used as backup
- UC5 Storing the configuration of a device, e.g. for archive or audit purposes
- UC6 Storing diagnostics data
- UC7 Allowing YANG instance data to potentially be carried within other IPC message formats
- UC8 Default instance data used as part of a templating solution
- UC9 Providing data examples in RFCs or internet drafts

In Appendix C we describe the first three use cases in detail.

There are already many and varied use cases where YANG instance data could be used. We do not want to limit future uses of instance data sets, so specifying how and when to use Yang instance data is out of scope for this document. It is anticipated that other documents

outside the instance data set itself will define specific use cases. Use cases are listed here only to indicate the need for this work.

2.1. High Level Principles

The following is a list of the basic principles of the instance data format:

- P1 Two standard formats are based on the XML and the JSON encoding
- P2 Re-use existing formats similar to the <get> operation/request
- P3 Add metadata about the instance data set
- P4 A YANG instance data file shall contain only a single YANG instance data set
- P5 A YANG instance data set may contain data for many target YANG modules
- P6 Instance data may include configuration data, state data or a mix of the two
- P7 Partial data sets are allowed
- P8 YANG instance data format may be used for any data for which target YANG module(s) are defined and available to the reader, independent of whether the module is actually implemented by a YANG server

3. Instance Data File Format

A YANG instance data file MUST contain a single instance data set and no additional data.

The instance data set is placed in a top level auxiliary container named "instance-data-set". An instance data set is made up of a header part and content-data. The initial header part carries metadata for the instance data set. It is defined by the ietf-yang-instance-data YANG module. The content-data is all data inside the anydata datanode, this carries the "real data" that we want to document/provide. The syntax and semantics of content-data is defined by the target YANG modules.

Two formats are specified that can be used to represent YANG instance data based on the XML and JSON encoding. Later as other YANG encodings (e.g. CBOR) are defined further instance data formats may be specified.

The content-data part of the XML format SHALL follow the encoding rules defined in [RFC7950] for XML and [RFC7951] for JSON and MUST use UTF-8 character encoding.

It MAY include metadata as defined by [RFC7952].

It MAY include entity-tags and timestamps as defined in [RFC8040]

It MAY include an explicit tag for default values as defined in [RFC6243] and [RFC8040]

It MAY include the origin metadata as specified in [I-D.ietf-netconf-nmda-netconf] and [I-D.ietf-netconf-nmda-restconf]

It MAY include implementation specific metadata. Unknown metadata MUST be ignored by users of YANG instance data, allowing it to be used later for other purposes.

It MAY include implementation specific XML attributes. Unknown attributes MUST be ignored by users of YANG instance data, allowing them to be used later for other purposes.

The content-data part will be very similar to the result returned for a NETCONF <get-data> or for a RESTCONF get operation.

The content-data part MUST conform to the corresponding target YANG Modules. A single instance data set MAY contain data for any number of target YANG modules; if needed it MAY carry the complete configuration and state data set for a YANG server. Default values SHOULD NOT be included.

Config=true and config=false data MAY be mixed in the instance data file.

Instance data files MAY contain partial data sets. This means mandatory, min-elements or require-instance=true constrains MAY be violated.

The name of the file SHALL be of the form:

instance-data-set-name ['@' revision-date] '.filetype'

E.g. acme-router-modules@2018-01-25.xml

The revision date is optional. ".filetype" SHALL be ".json" or ".xml" according to the format used.

Metadata, information about the data set itself SHALL be included in the instance data set. This data will be children of the top level instance-data-set container as defined in the ietf-instance-data YANG module. Metadata MUST include:

- o name of the instance data set

Metadata SHOULD include:

- o target-ptr: A pointer to the list of target YANG modules their revision, supported features and deviations.
- o An inline definition of target-modules, when the INLINE method is used for the target-ptr
- o Description of the instance data set. The description SHOULD contain information whether and how the data can change during the lifetime of the YANG server.

Metadata MAY include:

- o Organization responsible for the instance data set
- o Contact information
- o Information about the datastore associated with the instance data set e.g. the datastore from where the data was read or the datastore where the data could be loaded or the datastore which is being documented. This information is optional, as often a single datastore can not be specified.
- o Revision date of the instance data set. If both this date and the date in the instance data file name are present they MUST have the same value.
- o Timestamp: The date and time when the instance data set was last modified.
- o It is anticipated that different organizations will have the need to augment the metadata with various other data nodes.

3.1. Specifying the Target YANG Modules: target-ptr

To properly understand and use an instance data set the user needs to know the list of target YANG modules their revision, supported features and deviations. The metadata "target-ptr" is used to specify the YANG target module list. One of the following options SHOULD be used:

INLINE method: Include the needed information as part of instance data set as defined by e.g. ietf-yang-library

URI method: Include a URI that points to the target module set. (if you don't want to repeat the info again and again)

EXTERNAL Method: Do not include the target-ptr as the target YANG module set is already known, or the information is available through external documents.

Additional methods e.g. a YANG-package based solution may be added later.

Note, the specified target YANG modules only indicate the set of modules that were used to define this YANG instance data set. Sometimes instance data may be used for a YANG server supporting a different YANG module set e.g. for "UC2 Preloading Data" the instance data set may not be updated every time the YANG modules on the YANG server are updated, an unchanged instance data set may still be usable. Whether the instance data set is usable for a possibly different real-life target YANG module set depends on many factors including the compatibility between the specified target and the real-life target YANG module set (considering modules, revisions, features, deviations), the scope of the instance data, etc.

3.1.1. INLINE Method

One or more inline-target-spec elements SHALL be specified. The first one specifies ietf-yang-library or a similar YANG module listing target YANG modules with their name, revision-date, supported-features and deviations. Deviations or unsupported features MUST NOT remove any of the above data from the module. Using ietf-yang-library MUST be supported.

E.g. ietf-yang-library@2016-06-21.yang

As some versions of ietf-yang-library MAY contain different module-sets for different datastores, if multiple module-sets are defined, the instance data set's meta-data MUST contain the datastore information and instance data for the ietf-yang library MUST also contain information specifying the module-set for the relevant datastore.

Subsequent inline-target-spec elements MAY specify YANG modules augmenting the first module with useful data (e.g. a semantic version).

When using the inline method a 'target-modules' element MUST be present. This SHALL contain instance data corresponding to the YANG modules specified in the inline-target-spec elements specifying the set of target YANG modules for this instance-data-set.

3.1.2. URI Method

A target-uri element SHALL contain a URI that references another YANG instance data file. The current instance data file will use the same set of target YANG modules, revisions, supported features and deviations as the referenced YANG instance data file.

The referenced instance data file will usually contain data only for ietf-yang-library to specify the target YANG modules for the original instance data file.

The URI method is advantageous when the user wants to avoid the overhead of specifying the target YANG modules in the instance data file: E.g. In Use Case 6, when the system creates a diagnostic file every 10 minutes to document the state of the YANG server.

The referenced YANG instance data file might use the in-line method or might use the URI method to reference further instance data file(s). However at the end of this reference chain there MUST be an instance data file using the in-line method.

If a referenced instance data file is not available the revision data, supported features and deviations for the target YANG modules are unknown.

3.2. Examples

The following example is based on "UC1, Documenting Server Capabilities". It provides (a shortened) list of supported YANG modules and Netconf capabilities for a YANG server. It uses the inline method for the target-ptr.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
  "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-modules</name>
  <inline-target-spec>
    ietf-yang-library@2016-06-21.yang
  </inline-target-spec>
  <target-modules>
    <module-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module>
        <name>ietf-yang-library</name>
```



```
    <revision>2016-06-21</revision>
  </module>
  <module>
    <name>ietf-netconf-monitoring</name>
    <revision>2010-10-04</revision>
  </module>
</module-state>
</target-modules>
<revision>
  <date>2108-01-25</date>
  <description>Initial version</description>
</revision>
<description>Defines the minimal set of modules that any acme-router
  will contain.</description>
<contact>info@acme.com</contact>
<content-data>
  <!-- The example lists only 4 modules, but it could list the
    full set of supported modules for a YANG server, potentially many
    dozens of modules -->
  <module-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
    <module>
      <name>ietf-yang-library</name>
      <revision>2016-06-21</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-yang-library
      </namespace>
      <conformance-type>implement</conformance-type>
    </module>
    <module>
      <name>ietf-system</name>
      <revision>2014-08-06</revision>
      <namespace>urn:ietf:params:xml:ns:yang:ietf-system</namespace>
      <feature>sys:authentication</feature>
      <feature>sys:local-users</feature>
      <deviation>
        <name>acme-system-ext</name>
        <revision>2018-08-06</revision>
      </deviation>
      <conformance-type>implement</conformance-type>
    </module>
    <module>
      <name>ietf-yang-types</name>
      <revision>2013-07-15</revision>
      <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-types
      </namespace>
      <conformance-type>import</conformance-type>
    </module>
  </module>
```

```
<name>acme-system-ext</name>
<revision>2018-08-06</revision>
<namespace>urn:rdns:acme.com:oammodel:acme-system-ext
  </namespace>
<conformance-type>implement</conformance-type>
</module>
</module-state>
<netconf-state>
  <capabilities>
    <capability>
      urn:ietf:params:netconf:capability:validate:1.1
    </capability>
  </capabilities>
</netconf-state>
</content-data>
</instance-data-set>
```

Figure 1: XML Instance Data Set - Use case 1, Documenting server capabilities

The following example is based on "UC2, Preloading Default Configuration". It provides a (shortened) default rule set for a read-only operator role. It uses the inline method for the target-ptr.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
  "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>read-only-acm-rules</name>
  <inline-target-spec>ietf-yang-library@2016-06-21.yang
  </inline-target-spec>
  <target-modules>
    <module-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module>
        <name>ietf-netconf-acm</name>
        <revision>2012-02-22</revision>
      </module>
    </module-state>
  </target-modules>
  <revision>
    <date>2018-01-25</date>
    <description>Initial version</description>
  </revision>
  <description>Access control rules for a read-only role.</description>
  <contact>info@acme.com</contact>
  <content-data>
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
      <enable-nacm>true</enable-nacm>
      <read-default>deny</read-default>
      <exec-default>deny</exec-default>
      <rule-list>
        <name>read-only-role</name>
        <group>read-only-group</group>
        <rule>
          <name>read-all</name>
          <module-name>*</module-name>
          <access-operation>read</access-operation>
          <action>permit</action>
        </rule>
      </rule-list>
    </nacm>
  </content-data>
</instance-data-set>
```

Figure 2: XML Instance Data Set - Use case 2, Preloading access control data

The following example is based on UC6 Storing diagnostics data. An instance data set is produced by the YANG server every 15 minutes that contains statistics about NETCONF. As a new set is produced periodically multiple times a day a revision-date would be useless; instead a timestamp is included.

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "acme-router-netconf-diagnostics",
    "target-uri": "file:///acme-netconf-diagnostics-yanglib.json",
    "timestamp": "2018-01-25T17:00:38Z",
    "description":
      "Netconf statistics",
    "content-data": {
      "ietf-netconf-monitoring:netconf-state": {
        "statistics": {
          "netconf-start-time ": "2018-12-05T17:45:00Z",
          "in-bad-hellos ": "32",
          "in-sessions ": "397",
          "dropped-sessions ": "87",
          "in-rpcs ": "8711",
          "in-bad-rpcs ": "408",
          "out-rpc-errors ": "408",
          "out-notifications": "39007"
        }
      }
    }
  }
}
```

Figure 3: JSON Instance Data File example - UC6 Storing diagnostics data

4. Data Life cycle

Data defined or documented in YANG instance data sets may be used for preloading a YANG server with this data, but the server may populate the data without using the actual file in which case the instance data file is only used as documentation.

While such data will usually not change, data documented by instance data sets MAY be changed by the YANG server itself or by management operations. It is out of scope for this document to specify a method to prevent this. Whether such data changes and if so, when and how, SHOULD be described either in the instance data set's description statement or in some other implementation specific manner.

YANG instance data is a snap-shot of information at a specific point of time. If the data changes afterwards this is not represented in the instance data set anymore, the valid values can be retrieved in run-time via NETCONF/RESTCONF.

Notifications about the change of data documented by instance data sets may be supplied by e.g. the Yang-Push mechanism, but it is out of scope for this document.

5. Delivery of Instance Data

Instance data sets that are produced as a result of some sort of specification or design effort SHOULD be available without the need for a live YANG server e.g. via download from the vendor's website, or in any other way product documentation is distributed.

Other instance data sets may be read from or produced by the YANG server itself e.g. UC6 documenting diagnostic data.

6. Backwards Compatibility

The concept of backwards compatibility and what changes are backwards compatible are not defined for instance data sets as it is highly dependent on the specific use case and the target YANG model. However as instance data does use the concept of managed entities identified by key values the following guidelines are provided:

- o For list entries representing the same managed entity as previously key values SHOULD NOT be changed.
- o The meaning of list entries, representing the same managed entity as previously, SHOULD NOT be changed e.g. redefining an alarm-type but not changing its alarm-type-id should be avoided.
- o Keys for previously removed list entries SHOULD NOT be reused if they represent a different meaning.

7. YANG Model

```
<CODE BEGINS> file "ietf-yang-instance-data.yang"
module ietf-yang-instance-data {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data";
  prefix yid ;

  import ietf-yang-data-ext { prefix yd; }
  import ietf-datastores { prefix ds; }
  import ietf-inet-types { prefix inet; }
  import ietf-yang-types { prefix yang; }

  organization "IETF NETMOD Working Group";
  contact
```

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

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

description "The module defines the structure and content of YANG
instance data sets.";

```
revision 2019-02-20 {  
  description "Initial revision.";   
  reference "RFC XXXX: YANG Instance Data Format";  
}
```

```
yd:yang-data instance-data-format {  
  container instance-data-set {  
    description "Auxiliary container to carry meta-data for  
      the complete instance data set.";
```

```
    leaf name {  
      type string;  
      mandatory true;  
      description "Name of the YANG instance data set.";   
    }  
  }
```

```
  choice target-ptr {  
    description "A pointer to the list of target YANG modules  
      their revisions, supported features and deviations.";
```

```
    case inline {  
      leaf-list inline-target-spec {  
        type string {  
          pattern '.*@\\d{4}-\\d{2}-\\d{2}\\..yang';  
        }  
        min-elements 1;  
        ordered-by user;  
        description  
          "Indicates that target modules are specified inline.  
          Each value MUST be a YANG Module name including the  
          revision-date as defined for YANG file names in RFC7950.
```

E.g. ietf-yang-library@2016-06-21.yang

The first item is either ietf-yang-library or some other
YANG module that contains a list of YANG modules with
their name, revision-date, supported-features and
deviations.
As some versions of ietf-yang-library MAY contain

different module-sets for different datastores, if multiple module-sets are defined, the instance data set's meta-data MUST contain the datastore information and instance data for the ietf-yang-library MUST also contain information specifying the module-set for the relevant datastore.

Subsequent items MAY specify YANG modules augmenting the first module with useful data (e.g. a semantic version).";

```
}
anydata target-modules {
  mandatory true;
  description "Instance data corresponding to the YANG modules
    specified in the inline-target-spec nodes defining the set
    of target YANG modules for this instance-data-set.";
}

case uri {
  leaf target-uri {
    type inet:uri;
    description
      "A reference to another YANG instance data file.
      This instance data file will use the same set of target
      YANG modules, revisions, supported features and deviations
      as the referenced YANG instance data file.";
  }
}

leaf description { type string; }

leaf contact {
  type string;
  description "Contact information for the person or
    organization to whom queries concerning this
    instance data set should be sent.";
}

leaf organization {
  type string;
  description "Organization responsible for the instance
    data set.";
}

leaf datastore {
  type ds:datastore-ref;
  description "The identity of the datastore with which the
```

```
instance data set is associated. If a single specific
datastore can not be specified, the leaf MUST be absent.

If this leaf is absent, then the datastore to which the
instance data belongs is undefined.";
}

list revision {
  key date;
  description "Instance data sets that are produced as
    a result of some sort of specification or design effort
    SHOULD have at least one revision entry. For every
    published editorial change, a new one SHOULD be added
    in front of the revisions sequence so that all
    revisions are in reverse chronological order.

    For instance data sets that are read from
    or produced by the YANG server or otherwise
    subject to frequent updates or changes, revision
    SHOULD NOT be present";

  leaf date {
    type string {
      pattern '\d{4}-\d{2}-\d{2}';
    }
    description "Specifies the date the instance data set
      was last modified. Formatted as YYYY-MM-DD";
  }

  leaf description { type string; }
}

leaf timestamp {
  type yang:date-and-time;
  description "The date and time when the instance data set
    was last modified.

    For instance data sets that are read from or produced
    by the YANG server or otherwise subject to frequent
    updates or changes, timestamp SHOULD be present";
}

anydata content-data {
  mandatory true;
  description "Contains the real instance data.
    The data MUST conform to the relevant YANG Modules.";
}
}
```



```
    }  
  }  
<CODE ENDS>
```

8. Security Considerations

Depending on the nature of the instance data, instance data files MAY need to be handled in a secure way. The same type of handling should be applied, that would be needed for the result of a <get> operation returning the same data.

9. IANA Considerations

This document registers one URI and one YANG module.

9.1. URI Registration

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-yang-instance-data

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

9.2. YANG Module Name Registration

This document registers one YANG module in the YANG Module Names registry [RFC6020].

name: ietf-yang-instance-data
namespace: urn:ietf:params:xml:ns:yang:ietf-yang-instance-data
prefix: yid
reference: RFC XXXX

10. Acknowledgments

For their valuable comments, discussions, and feedback, we wish to acknowledge Andy Bierman, Juergen Schoenwaelder, Rob Wilton, Joe Clark, Martin Bjorklund, Ladislav Lhotka, Qin Wu and other members of the Netmod WG.

11. References

11.1. Normative References

- [I-D.ietf-netconf-nmda-netconf]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "NETCONF Extensions to Support the Network Management Datastore Architecture", draft-ietf-netconf-nmda-netconf-08 (work in progress), October 2018.
- [I-D.ietf-netconf-nmda-restconf]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "RESTCONF Extensions to Support the Network Management Datastore Architecture", draft-ietf-netconf-nmda-restconf-05 (work in progress), October 2018.
- [I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Extensions", draft-ietf-netmod-yang-data-ext-01 (work in progress), March 2018.
- [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>>.
- [RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <<https://www.rfc-editor.org/info/rfc6243>>.
- [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>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/info/rfc7951>>.
- [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>>.

11.2. Informative References

- [I-D.ietf-ccamp-alarm-module]
Vallin, S. and M. Bjorklund, "YANG Alarm Module", draft-ietf-ccamp-alarm-module-07 (work in progress), January 2019.
- [I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.
- [I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-22 (work in progress), February 2019.
- [I-D.wu-netconf-restconf-factory-restore]
Wu, Q., Lengyel, B., and Y. Niu, "Factory default Setting", draft-wu-netconf-restconf-factory-restore-03 (work in progress), October 2018.
- [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>>.
- [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. Open Issues

- o Augmenting metadata must be possible. As of now it looks like yang-data-ext will solve that. If not, define instance data as regular YANG instead of yd:yang-data.

Appendix B. Changes between revisions

v01 - v02

- o Removed design time from terminology

- o Defined the format of the content-data part by referencing various RFCs and drafts instead of the result of the get-data and get operations.
- o Changed target-ptr to a choice
- o Inline target-ptr may include augmenting modules and alternatives to ietf-yang-library
- o Moved list of target modules into a separate <target-modules> element.
- o Added backwards compatibility considerations

v00 - v01

- o Added the target-ptr metadata with 3 methods
- o Added timestamp metadata
- o Removed usage of dedicated .yid file extension
- o Added list of use cases
- o Added list of principles
- o Updated examples
- o Moved detailed use case descriptions to appendix

v05 - v00-netmod

- o New name for the draft following Netmod workgroup adoption. No other changes

v04 - v05

- o Changed title and introduction to clarify that this draft is only about the file format and documenting server capabilities is just a use case.
- o Added reference to draft-wu-netconf-restconf-factory-restore
- o Added new open issues.

v03 - v04

- o Updated changelog for v02-v03

v02 - v03

- o Updated the document according to comments received at IETF102
- o Added parameter to specify datastore
- o Rearranged chapters
- o Added new use case: Documenting Factory Default Settings
- o Added "Target YANG Module" to terminology
- o Clarified that instance data is a snapshot valid at the time of creation, so it does not contain any later changes.
- o Removed topics from Open Issues according to comments received at IETF102

v01 - v02

- o The recommendation to document server capabilities was changed to be just the primary use-case. (Merged chapter 4 into the use case chapter.)
- o Stated that RFC7950/7951 encoding must be followed which also defines (dis)allowed whitespace rules.
- o Added UTF-8 encoding as it is not specified in t950 for instance data
- o added XML declaration

v00 - v01

- o Redefined using yang-data-ext
- o Moved metadata into ordinary leafs/leaf-lists

Appendix C. Detailed Use Cases - Non-Normative

C.1. Use Cases

We present a number of use cases where YANG instance data is needed.

C.1.1.1. Use Case 1: Early Documentation of Server Capabilities

A YANG server has a number of server-capabilities that are defined in YANG modules and can be retrieved from the server using protocols like NETCONF or RESTCONF. YANG server capabilities include

- o data defined in ietf-yang-library: YANG modules, submodules, features, deviations, schema-mounts, datastores supported ([I-D.ietf-netconf-rfc7895bis])
- o alarms supported ([I-D.ietf-ccamp-alarm-module])
- o data nodes, subtrees that support or do not support on-change notifications ([I-D.ietf-netconf-yang-push])
- o netconf-capabilities in ietf-netconf-monitoring

While it is good practice to allow a client to query these capabilities from the live YANG server, that is often not possible.

Often when a network node is released an associated NMS (network management system) is also released with it. The NMS depends on the capabilities of the YANG server. During NMS implementation information about server capabilities is needed. If the information is not available early in some off-line document, but only as instance data from the live network node, the NMS implementation will be delayed, because it has to wait for the network node to be ready. Also assuming that all NMS implementors will have a correctly configured network node available to retrieve data from, is a very expensive proposition. (An NMS may handle dozens of node types.)

Network operators often build their own home-grown NMS systems that needs to be integrated with a vendor's network node. The operator needs to know the network node's server capabilities in order to do this. Moreover the network operator's decision to buy a vendor's product may even be influenced by the network node's OAM feature set documented as the Yang server's capabilities.

Beside NMS implementors, system integrators and many others also need the same information early. Examples could be model driven testing, generating documentation, etc.

Most server-capabilities are relatively stable and change only during upgrade or due to licensing or addition or removal of HW. They are usually defined by a vendor at design time, before the product is released. It feasible and advantageous to define/document them early e.g. in a YANG instance data File.

It is anticipated that a separate IETF document will define in detail how and which set of server capabilities should be documented.

C.1.2. Use Case 2: Preloading Data

There are parts of the configuration that must be fully configurable by the operator, however for which often a simple default configuration will be sufficient.

One example is access control groups/roles and related rules. While a sophisticated operator may define dozens of different groups often a basic (read-only operator, read-write system administrator, security-administrator) triplet will be enough. Vendors will often provide such default configuration data to make device configuration easier for an operator.

Defining Access control data is a complex task. To help the device vendor pre-defines a set of default groups (/nacm:nacm/groups) and rules for these groups to access specific parts of common models (/nacm:nacm/rule-list/rule).

YANG instance data files are used to document and/or preload the default configuration.

C.1.3. Use Case 3: Documenting Factory Default Settings

Nearly every YANG server has a factory default configuration. If the system is really badly misconfigured or if the current configuration is to be abandoned the system can be reset to this default.

In Netconf the <delete-config> operation can already be used to reset the startup datastore. There are ongoing efforts to introduce a new, more generic reset-datastore operation for the same purpose [I-D.wu-netconf-restconf-factory-restore]

The operator currently has no way to know what the default configuration actually contains. YANG instance data can be used to document the factory default configuration.

Authors' Addresses

Balazs Lengyel
Ericsson
Magyar Tudosok korutja 11
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2019

R. Wilton
Cisco Systems, Inc.
March 10, 2019

YANG Packages
draft-rwilton-netmod-yang-packages-01

Abstract

This document defines YANG packages, an organizational structure holding a set of related YANG modules, that can be used to simplify the conformance and sharing of YANG schema. It describes how YANG instance data documents are used to define YANG packages, and how the YANG library information published by a server can be augmented with additional packaging related information.

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 September 11, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology and Conventions	2
2. Introduction	3
3. Background on YANG packaging	4
4. Possible alternative solutions	4
4.1. Using module tags	4
4.2. Using YANG library	5
5. Objectives	6
6. Package description	7
6.1. Package definition rules	7
6.2. Package versioning	8
6.3. Client server package conformance	10
6.4. Schema referential completeness	10
6.5. Submodules packaging considerations	11
6.6. Revision history	11
6.7. Uniqueness of packages and global registry	11
7. YANG Packaging instance data	12
8. YANG Packaging additions to YANG library	13
8.1. Package List	14
8.2. Binding from schema to package	14
8.3. Tree diagram	15
9. YANG Packaging groupings	15
10. YANG Modules	17
11. Security Considerations	29
12. IANA Considerations	30
13. Open Questions/Issues	31
14. Acknowledgements	31
15. References	31
15.1. Normative References	31
15.2. Informative References	32
Appendix A. Tree output for ietf-yang-library with package augmentations	32
Appendix B. Examples	34
B.1. Example IETF Network Device YANG package	35
B.2. Example IETF Basic Routing YANG package	37
B.3. Package import conflict resolution example	40
Author's Address	43

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.verdt-netmod-yang-versioning-reqs].

This document also makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

- o datastore schema

In addition, this document makes use of the following terminology:

- o bc: Used as an abbreviation for a backwards-compatible change.
- o nbc: Used as an abbreviation for a non-backwards-compatible change.
- o editorial change: A backwards-compatible change that does not change the YANG module semantics in any way.

Note - the bc/nbc/editorial terminology should probably be defined and referenced from the YANG module versioning solution draft.

2. Introduction

This document defines and describes the YANG [RFC7950] constructs that are used to define and use YANG packages.

A YANG package is an organizational structure that groups a set of related YANG modules together into a consistent versioned definition. YANG packages can themselves refer to and reuse other package definitions.

The draft consists of the following significant sections:

A background section that describes some of the prior work in this area, both within IETF and the wider industry.

An overview of the objectives for a YANG packaging solution, and also what work is out of scope for this document.

The definition of YANG packages, how package definitions are constructed, and how they are used.

How YANG instance data documents [I-D.ietf-netmod-yang-instance-file-format] are used to define particular YANG package instances.

Augmentations to the YANG library [I-D.ietf-netconf-rfc7895bis] content published by servers to include YANG packaging related information.

YANG modules the provide the definitions for YANG packages.

Non-normative examples of YANG package instances are provided in the appendicies.

3. Background on YANG packaging

It has long been acknowledged within the IETF NETMOD 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 particular package instances. This document did not progress in the working group, although this may have been due to other higher priority concerns or resource constraints within the working group rather than due to consideration of the technical merits of the proposed approach.

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 to the YANG versioning requirements, [I-D.verdt-netmod-yang-versioning-reqs]. Solutions to the versioning requirements can be split into several distinct areas. One draft, TBD (draft-verdt-netmod-yang-semver), has a primary focus on YANG versioning scoped to individual modules. But an overall solution should also consider YANG versioning and conformance scoped to a server's datastore schema. YANG packages may help form part of the solution for versioning at the datastore schema level.

4. Possible alternative solutions

4.1. 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 centralized YANG Catalog, or in instance data documents 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 hierarchical scheme to construct YANG schema that is proposed in this draft.

4.2. 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 document. The use of YANG packages is intended to offer 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 can 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 YANG schema for each device, instead they can rely on the named packages. YANG libraries use of checksums are unique only to the device that generated them.
3. Packages are versioned using a semantic versioning scheme, YANG library does not have a schema level semantic version number, although this could potentially be added if required.
4. For a YANG library instance data document to contain the necessary information, it needs both YANG library, and probably also various augmentations (e.g. to include each module's semantic version number), unless a new version of YANG library is defined containing this information. A module definition for a YANG package could be defined to contain all of the necessary information to solve the problem.

5. YANG library is designed to publish information about the modules, datastores, and datastore schema used by a server. It is unclear whether exactly the same information is required for an offline schema definition, or whether these definitions might deviate from each other over time. E.g., some thought needs to be given concerning the relationship between datastores and schema.

5. Objectives

The main goals of YANG package definitions include, but are not restricted to:

- o To act as a simplified YANG conformance mechanism. Rather than conformance being performed against a set of individual YANG module revisions, conformance could also be more simply stated in terms of YANG packages, with a set modifications (e.g. additional modules, deviations, or features).
- o To allow YANG datastore schema to be specified in a more concise way rather than having to list all modules and revisions. YANG package definitions can be defined in documents that can be referenced by 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.
- o To provide generic packaging related YANG grouping definitions for use in other YANG modules, as required.
- o 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 were 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, then finding a consistent lowest common denominator set of YANG module versions may be difficult, or impossible.

Protocol mechanisms of how clients could negotiate which packages or package versions are be used for client server communications are outside the scope of this document. However, the design of the YANG library augmentations for YANG packages are intended to keep open the possibility of such extensions in future work.

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 be used to extend the functionality supported by YANG packages.

6. Package description

This document specifies an approach to defining YANG packages that is different to either of the approaches described in the background.

The approach defined here is for a YANG package definition structure to be defined using existing YANG language statements without requiring extensions or new YANG statements. By making use of this structure, particular YANG package instances can be defined as YANG instance data documents [I-D.ietf-netmod-yang-instance-file-format] with well defined names and locations.

The YANG semantic versioning scheme, described in draft-verdt-netmod-yang-semver (TBD), is used to version YANG packages using an equivalent scheme to how individual YANG modules version numbers are changed.

YANG library is augmented to allow servers to report the packages that they implement and to associate those packages back to particular datastore schema.

TODO - It would be helpful if the YANG instance data file format [I-D.ietf-netmod-yang-instance-file-format] could also reference a YANG packages to specify the schema associated with an instance data document. This could either be defined in instance-file-format draft, or as a YANG augmentation as part of this draft.

Each version of a YANG package defines: a set of YANG modules that are implemented at particular versions or revisions; a set of YANG modules that are import-only with particular versions or revisions; and a set of mandatory module features that implementations of the package MUST implement or otherwise deviate.

6.1. Package definition rules

The following rules define how packages are defined:

Every YANG package definition MUST be referentially complete. I.e. all import and include statements for all YANG modules included in a package MUST resolve to a module specified in the package itself, or an imported package.

For a given package, each separate instance of the package MUST have a unique version number that follows the semantic versioning rules described in Section 6.2.

A package MAY have a revision-date. Any package revision-dates MUST be unique for different package versions.

For each module implemented by a package, only a single revision/version MUST be implemented.

The version/revision of a module listed in the package module list supercedes any version/revision of the module listed in a imported package module list. This allows a package to resolve any conflicting implemented module versions/revisions in imported packages.

The replaces-revision leaf-list in the import-only-module list can be used to exclude duplicate revisions of import-only modules from imported packages. Otherwise, the import-only-modules for a package are the import-only-modules from all imported packages combined with any modules listed in the packages import-only-module list.

Modules referenced by a package SHOULD specify the version of the module, both in the package definition and within the module definition itself.

Modules referenced by a package MUST specify the revision date of the module, both in the package definition and within the module definition itself.

6.2. Package versioning

Every YANG package must specify a YANG semantic version field that defines the particular version of the package.

The rules for incrementing the YANG package version number are equivalent to the semantic versioning rules used to version individual YANG modules, defined in TBD (draft-verdt-netmod-yang-semver).

The semantic versioning rules, as they apply to YANG packages, are defined using the following two step process:

The first step is to determined whether the change to the YANG package is classified as a major, minor, or editorial based on the content that has changed in the package relative to the previous version. Where available, the semantic version number of the

referenced elements in the package (imported packages or modules) can be used to help determine what type of change is being made. The formal rules are:

If any of the referenced elements of the package (imported packages or modules) are changed in an nbc way, or if any imported package, module, or mandatory-feature is removed from the package definition, then the package has been updated in an nbc way.

If none of the referenced elements of the package (imported packages, modules) are removed or changed in a nbc way, but some referenced elements are changed in a bc way, or new referenced elements or mandatory-features added, then the package is deemed to be updated in a bc way.

If none of the referenced elements of the package (imported packages, modules) are added, removed, or changed in a nbc or bc way, but some referenced elements have editorial changes then the package is deemed to be updated in an editorial way.

The second step, after it has been determined what type of version change is being made to the YANG package, is for the YANG semantic versioning rules to be applied to update the YANG package semantic version number. The formal rules are:

If the package is being updated in a nbc way, then the package version "X.Y.Z[m|M]" SHOULD be updated to "X+1.0.0" unless that package version has already been defined with different content, in which case the package version "X.Y.Z+1M MUST be used instead.

If the package is being updated in a bc way, then the package version "X.Y.Z[m|M]" SHOULD be updated to "X.Y+1.0" unless that package version has already been defined with different content, in which case if the current package version is "X.Y.ZM" then it MUST be updated to "X.Y.Z+1M", or otherwise "X.Y.Z+1m".

If the package is being updated in an editorial way, then the package version "X.Y.Z[m|M]" MUST be updated to "X.Y.Z+1[m|M]", retaining the 'm|M' character if it is already present in the previous version."

Package YANG semantic version numbers beginning with 0, i.e "0.X.Y" are regarded as beta definitions and need not follow the nbc rules, and the minor version number can be incremented instead.

In all cases, the 3 number fields that comprise a YANG semantic version number associated with a YANG package MUST uniquely identify the contents of that YANG package.

6.3. Client server package conformance

The YANG semantic versioning scheme used for YANG packages means that a client can determine the nature of changes between two package revisions.

This means that a client is not restricted to working only with servers that advertise exactly the same version of package in YANG library. Instead, reasonable clients should be able to interoperate with a server that supports a package version that is backwards compatible to what the client is designed for.

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 necessarily 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 NBC patch level changes (i.e. 'M' labelled versions).

6.4. Schema referential completeness

A YANG package may represent a schema that is 'referentially complete', or 'referentially incomplete'.

If all import statements in all YANG modules included in the package (either directly, or through imported 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 to group sets of logically related modules together, but without requiring a fixed dependency on all imported 'types' modules, instead leaving the choice of specific revisions of 'types' modules to be resolved when the package definition is used.

6.5. Submodules packaging considerations

As defined in [RFC7950] and draft-verdt-netmod-yang-semver (TBD), 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 YANG schema to be fully constructed from a YANG package instance-data definition.

Restructuring how a module uses, or does not use, submodules is treated as an editorial level change in YANG semantic versioning, on the condition that there is no change in the modules semantic behavior due to the restructuring.

To ensure that a module and any constituent submodule are tightly related, all 'include' statements in a YANG module SHOULD specify revision-dates of the included submodules. If 'include' statement revision-dates are included in the YANG module then they MUST match the 'revision' field specified for the submodule in the packages's submodules lists.

6.6. Revision history

YANG packages do not contain a revision history, because a linear revision history does not work well for a versioning object that supports branching. In addition, some packages could have frequent revisions, and a long revision history would bloat the package definition.

To mitigate this, the package definition includes a 'previous-version' leaf that indicates the specific version this package definition is based on. By recursively examining the 'previous-version' leaf of a package definition, a full revision history can be dynamically constructed if required.

6.7. Uniqueness of packages and global registry

The name given to a package SHOULD be globally unique, and it SHOULD include an appropriate organization prefix in the name, equivalent to YANG module naming conventions.

Ideally a YANG instance data document defining a particular package version would be publically available at one or more URLs.

7. YANG Packaging instance data

YANG packages are expected to be defined as YANG instance data documents [I-D.ietf-netmod-yang-instance-file-format] using the YANG schema below to define the package data itself.

The instance data document 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
  +--ro yang-package
    +--ro name                yang:yang-identifier
    +--ro version              yang-sem-ver
    +--ro revision-date?      yanglib:revision-identifier
    +--ro location*            inet:uri
    +--ro description?         string
    +--ro reference?           string
    +--ro previous-version?    yang-sem-ver
    +--ro tag*                  tags:tag
    +--ro referentially-complete? boolean
    +--ro mandatory-feature*   string
    +--ro imported-packages* [name version]
      +--ro name                yang:yang-identifier
      +--ro version              yang-sem-ver
      +--ro deviated?           boolean
      +--ro location*            inet:uri
    +--ro module* [name]
      +--ro name                yang:yang-identifier
      +--ro revision?           revision-identifier
      +--ro version?            yang-sem-ver
      +--ro namespace           inet:uri
      +--ro location*            inet:uri
      +--ro submodule* [name]
        +--ro name                yang:yang-identifier
        +--ro revision            yanglib:revision-identifier
        +--ro location*            inet:uri
    +--ro import-only-module* [name revision]
      +--ro name                yang:yang-identifier
      +--ro revision              union
      +--ro version?            yang-sem-ver
      +--ro namespace           inet:uri
      +--ro location*            inet:uri
      +--ro submodule* [name]
        +--ro name                yang:yang-identifier
        +--ro revision            yanglib:revision-identifier
        +--ro location*            inet:uri
    +--ro replaces-revision*   yanglib:revision-identifier

```

8. YANG Packaging additions to YANG library

8.1. Package List

The main addition is a top level 'yang-library/package' list that lists all package of all versions known to the server. Each package itself is defined using imported packages and module-sets to define the specific set of modules implemented and imported by the package. The use of module-sets allows the module definitions to be shared with the existing YANG library schema definitions. The existing rule of RFC 7995bis related to combining modules-sets also applies here, i.e. The combined set of modules defined by the module-sets MUST NOT contain modules implemented at different revisions. I.e. the module-sets leaf-list is directly equivalent to the explicit module and import-only-module lists in the instance data YANG package definition.

The 'yang-library/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.

8.2. Binding from schema to package

The second augmentation is to allow a server to optionally indicate that a schema definition directly relates to a package. Since YANG packages are available offline, it may be sufficient for a client to only check that a compatible version of the YANG package is being implemented by the server without fetching and comparing the full module list.

If a server indicates that its schema maps to a particular package then it MUST support all mandatory-features defined as part of that package, and it MUST NOT have any deviations to the modules defined by the package. A server MAY implement features not specified in the package's mandatory-features list.

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 imported package, and the new package can define additional modules containing deviations to the original package, allowing the new package to accurately describe the server behavior. There is no specific mechanism provided to indicate that a mandatory-feature is not supported on a server, but deviations MAY be used to disable functionality predicated by a mandatory-feature.

8.3. Tree diagram

The "ietf-yang-library-packages" YANG module has the following structure:

```

module: ietf-yang-library-packages
  augment /yanglib:yang-library:
    +--ro package* [name version]
      +--ro name          yang:yang-identifier
      +--ro version       yang-sem-ver
      +--ro revision-date? yanglib:revision-identifier
      +--ro location*     inet:uri
      +--ro description?  string
      +--ro reference?    string
      +--ro previous-version? yang-sem-ver
      +--ro tag*          tags:tag
      +--ro referentially-complete? boolean
      +--ro mandatory-feature* string
      +--ro imported-packages* [name version]
        +--ro name          yang:yang-identifier
        +--ro version       yang-sem-ver
        +--ro deviated?    boolean
      +--ro module-set*
        -> /yanglib:yang-library/module-set/name
  augment /yanglib:yang-library/yanglib:schema:
    +--ro package
      +--ro name?      -> /yanglib:yang-library/package/name
      +--ro version?   leafref
  augment /yanglib:yang-library/yanglib:module-set/
    yanglib:import-only-module:
    +--ro replaces-revision* yanglib:revision-identifier

```

9. YANG Packaging groupings

Groupings for YANG packaging related constructs are provided in a 'types' module for use by the instance-data and YANG library constructs described previously. They are also available to be used by other modules that have a need for packaging information.

The "ietf-yang-package-types" YANG module has the following structure:

```

module: ietf-yang-package-types

  grouping yang-pkg-identification-leafs

```

```

+----- name          yang:yang-identifier
+----- version       yang-sem-ver
grouping yang-pkg-common-leafs
+----- revision-date? yanglib:revision-identifier
+----- location*      inet:uri
+----- description?   string
+----- reference?     string
+----- previous-version? yang-sem-ver
+----- tag*           tags:tag
+----- referentially-complete? boolean
+----- mandatory-feature* string
+----- imported-packages* [name version]
|   +----- name          yang:yang-identifier
|   +----- version       yang-sem-ver
|   +----- deviated?     boolean
grouping yang-pkg-library-definition
+----- name          yang:yang-identifier
+----- version       yang-sem-ver
+----- revision-date? yanglib:revision-identifier
+----- location*      inet:uri
+----- description?   string
+----- reference?     string
+----- previous-version? yang-sem-ver
+----- tag*           tags:tag
+----- referentially-complete? boolean
+----- mandatory-feature* string
+----- imported-packages* [name version]
|   +----- name          yang:yang-identifier
|   +----- version       yang-sem-ver
|   +----- deviated?     boolean
+----- module-set*
|   -> /yanglib:yang-library/module-set/name
grouping yang-pkg-file-definition
+----- name          yang:yang-identifier
+----- version       yang-sem-ver
+----- revision-date? yanglib:revision-identifier
+----- location*      inet:uri
+----- description?   string
+----- reference?     string
+----- previous-version? yang-sem-ver
+----- tag*           tags:tag
+----- referentially-complete? boolean
+----- mandatory-feature* string
+----- imported-packages* [name version]
|   +----- name          yang:yang-identifier
|   +----- version       yang-sem-ver
|   +----- deviated?     boolean
|   +----- location*      inet:uri

```



```

+---- module* [name]
|   +---- name          yang:yang-identifier
|   +---- revision?     revision-identifier
|   +---- version?      yang-sem-ver
|   +---- namespace     inet:uri
|   +---- location*     inet:uri
|   +---- submodule* [name]
|       +---- name?      yang:yang-identifier
|       +---- revision   yanglib:revision-identifier
|       +---- location*  inet:uri
+---- import-only-module* [name revision]
    +---- name?          yang:yang-identifier
    +---- revision?      union
    +---- version?       yang-sem-ver
    +---- namespace      inet:uri
    +---- location*      inet:uri
    +---- submodule* [name]
        +---- name?      yang:yang-identifier
        +---- revision   yanglib:revision-identifier
        +---- location*  inet:uri
+---- replaces-revision* yanglib:revision-identifier

```

10. YANG Modules

The YANG module definitions for the modules described in the previous sections.

```

<CODE BEGINS> file "ietf-yang-package-types@2018-11-26.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-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types.";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types.";
  }
  import ietf-yang-library {
    prefix yanglib;
    reference "RFC 7895bis: YANG Library";
  }
  import ietf-module-tags {

```

```
    prefix tags;
    reference "XXX, (draft-ietf-netmod-module-tags-03): 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) 2018 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.
revision 2018-11-26 {
  description
    "Initial revision";
  reference
    "RFC XXXX: YANG Schema Versioning.";
}

/*
 * Typedefs
 */

typedef yang-sem-ver {
  type string {
```

```
    pattern '\d+[.]\d+[.]\d+[mM]?';
  }
  description
    "Represents a YANG semantic version number.";
  reference
    "TODO - Should be defined by YANG versioning types module";
}

/*
 * Groupings
 */

grouping yang-pkg-identification-leafs {
  description
    "Parameters for identifying a specific version of a YANG
    package";

  leaf name {
    type yang:yang-identifier;
    mandatory true;
    description
      "The YANG package name.";
  }

  leaf version {
    type yang-sem-ver;
    mandatory true;
    description
      "YANG package version. Follows YANG semantic versions rules
      defined in XXX";
  }
}

grouping yang-pkg-common-leafs {
  description
    "Defines definitions common to all YANG package definitions.";

  leaf revision-date {
    type yanglib:revision-identifier;

    description
      "An optional revision identifier of when this package version
      was created. This does not need to be unique across all
      versions of a package.";
  }

  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";
}

leaf description {
  type string;

  description "Provides a description of the package";
}

leaf reference {
  type string;

  description "Allows for a reference for the package";
}

leaf previous-version {
  type yang-sem-ver;
  description
    "The previous package version that this version has been
     derived from. This leaf allows a full version history graph
     to be constructed if required.";
}

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 referentially-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 imported packages.";
}

leaf-list mandatory-feature {
    type string;
    // TODO - Is there a better type for this?
    description
        "List all features from modules included in the package that
        MUST be supported by any server implementing the package.
        All other features defined in included packages are OPTIONAL
        to implement.

        Features are identified using <module-name>:<feature>";
}

list imported-packages {
    key "name version";
    description
        "An entry in this list represents a package that is imported
        as part of the package definition.

        If packages implement different revisions or versions of the
        same module, then an explicit entry in the module list MUST
        be provided to select the specific module version
        'implemented' by this package definition.

        For import-only modules, the replaces-revision leaf-list can
        be used to select the specific module versions imported by
        this package.";
    reference
        "XXX";

    uses yang-pkg-identification-leafs;

    leaf deviated {
        type boolean;
        default true;
        description
            "Set to true if any data nodes in this package are modified
            in a non backwards compatible way, either through the use
            of deviations, or because one of the modules has been
            replaced by an earlier module version.";
    }
}
}
```

```
grouping yang-pkg-file-definition {
  description
    "The set of parameters that describe a particular YANG package.";

  uses yang-pkg-identification-leafs;

  uses yang-pkg-common-leafs {
    augment "imported-packages" {
      description "Add the package location path";

      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 version
     'implemented' by an imported package";
  reference
    "RFC 7950: The YANG 1.1 Data Modeling Language.";

  uses yanglib:module-identification-leafs;

  leaf version {
    type yang-sem-ver;
    description
      "The YANG module or submodule version.  If no version
       statement is present in the YANG module or submodule, this
       leaf is not instantiated.";
  }
}
```

```
leaf namespace {
  type inet:uri;
  mandatory true;
  description
    "The XML namespace identifier for this module.";
}
uses yanglib:location-leaf-list;

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 yanglib: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.";
  }

  uses yanglib:location-leaf-list;
}

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 union {
    type yanglib:revision-identifier;
    type string {
      length 0;
    }
  }
  description
    "The YANG module revision date. A zero-length string is
    used if no revision statement is present in the YANG
    module.";
}
leaf version {
  type yang-sem-ver;
  description
    "The YANG module or submodule version. If no version
    statement is present in the YANG module or submodule, this
    leaf is not instantiated.";
}
leaf namespace {
  type inet:uri;
  mandatory true;
  description
    "The XML namespace identifier for this module.";
}

uses yanglib:location-leaf-list;

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 yanglib: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.";
  }
}
```



```
    uses yanglib:location-leaf-list;
  }

  leaf-list replaces-revision {
    type yanglib:revision-identifier;
    description
      "Gives the revision of an import-only-module defined in
       an imported package that is replaced by this
       import-only-module revision.";
  }
}

grouping yang-pkg-library-definition {
  description
    "The set of parameters that describe a particular YANG package.";

  uses yang-pkg-identification-leafs;
  uses yang-pkg-common-leafs;

  leaf-list module-set {
    type leafref {
      path "/yanglib:yang-library/yanglib:module-set/yanglib:name";
    }
    description
      "Describes any modules in addition to, and replacing, and
       modules defined in the imported packages.

       If a non import-only module appears in multiple module sets,
       then the module revision and the associated features and
       deviations must be identical.";
  }
}
}
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-yang-package2018-11-26.yang"
module ietf-yang-package {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package";
  prefix pkg;

  import ietf-yang-package-types {
    prefix pkg-types;
    reference "RFC XXX: YANG Schema Versioning.";
  }
}
```

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 schema for an YANG instance data document specifying a YANG package.

Copyright (c) 2018 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.

```
revision 2018-11-26 {  
  description  
    "Initial revision";  
  reference  
    "RFC XXXX: YANG Schema Versioning."  
}
```

```
/*  
 * Top-level container  
 */
```

```
container yang-package {  
  config false;  
  description  
    "Defines a YANG package.
```

Intended to be used to specify a YANG package as an instance data document.";

```
    uses pkg-types:yang-pkg-file-definition;
  }
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-yang-library-packages@2018-11-26.yang"
module ietf-yang-library-packages {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-library-packages";
  prefix pkg;

  import ietf-yang-package-types {
    prefix pkg-types;
    reference "RFC XXX: YANG Packages.";
  }
  import ietf-yang-library {
    prefix yanglib;
    reference "RFC 7895bis: 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) 2018 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.
revision 2018-11-26 {
  description
    "Initial revision";
  reference
    "RFC XXXX: YANG Schema Versioning.";
}

/*
 * Add in the list of packaged into YANG library.
 */
augment "/yanglib:yang-library" {
  description "Add YANG package definitions into YANG library";

  list package {
    config "false";
    key "name version";

    description "Defines the packages available on this server.";

    uses "pkg-types:yang-pkg-library-definition";
  }
}

/*
 * Allow schema to be related to a YANG package.
 */
augment "/yanglib:yang-library/yanglib:schema" {
  description
    "Allow datastore schema to be related to a YANG package";

  container package {
    leaf name {
      type leafref {
        path "/yanglib:yang-library/package/name";
      }
      description
        "The name of the package this schema relates to.";
    }
  }
}
```

```
    leaf version {
      type leafref {
        path '/yanglib:yang-library/'
          + 'package[name = current()/../name]/version';
      }

      description
        "The version of the package this schema relates to.";
    }

    description
      "Describes which package the schema directly relates to, if
        any.";
  }
}

/*
 * Allow import-only modules to list the versions that they are
 * replacing.
 */

augment
  "/yanglib:yang-library/yanglib:module-set/" +
  "yanglib:import-only-module" {

    description
      "Add replaces-revision to import-only-module definitions";

    leaf-list replaces-revision {
      type yanglib:revision-identifier;
      description
        "Gives the revision of an import-only-module defined in an
          imported package that is replaced by this import-only-module
          revision.

          Only used for YANG package definitions";
    }
  }
}
<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 documents, 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 documents to restrict access to sensitive information.

As per the YANG library security considerations, the module, revision and version 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 packaging 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 packaging 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.

TODO - Add IANA registrations for YANG modules defined in this draft.

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, Ladislav Lhotka, and Jason Sterne.

15. References

15.1. Normative References

- [I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K.,
and R. Wilton, "YANG Library", draft-ietf-netconf-
rfc7895bis-07 (work in progress), October 2018.
- [I-D.ietf-netmod-module-tags]
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module
Tags", draft-ietf-netmod-module-tags-07 (work in
progress), March 2019.
- [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-02
(work in progress), February 2019.
- [I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-
verdt-netmod-yang-versioning-reqs-02 (work in progress),
November 2018.
- [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>>.
- [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>>.
- [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>>.

15.2. Informative References

- [I-D.bierman-netmod-yang-package] Bierman, A., "The YANG Package Statement", draft-bierman-netmod-yang-package-00 (work in progress), July 2015.
- [I-D.ietf-netmod-artwork-folding] Watsen, K., Wu, Q., Farrel, A., and B. Claise, "Handling Long Lines in Artwork in Internet-Drafts and RFCs", draft-ietf-netmod-artwork-folding-00 (work in progress), November 2018.
- [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. Tree output for ietf-yang-library with package augmentations

Complete tree output for ietf-yang-library with package augmentations.

```

module: ietf-yang-library
  +--ro yang-library
    |
    | +--ro module-set* [name]
    | |
    | | +--ro name string
    | | +--ro module* [name]
    | | |
    | | | +--ro name yang:yang-identifier
    | | | +--ro revision? revision-identifier
    | | | +--ro namespace inet:uri
    | | | +--ro location* inet:uri
    | | | +--ro submodule* [name]
    | | | |
    | | | | +--ro name yang:yang-identifier
    | | | | +--ro revision? revision-identifier
    | | | | +--ro location* inet:uri
    | | | +--ro feature* yang:yang-identifier
    | | | +--ro deviation* -> ../../module/name
    | | +--ro import-only-module* [name revision]
    | | |
    | | | +--ro name yang:yang-identifier
    | | | +--ro revision union
    | | | +--ro namespace inet:uri
    | | | +--ro location* inet:uri
    | | | +--ro submodule* [name]
    | | | |
    | | | | +--ro name yang:yang-identifier
    | | | | +--ro revision? revision-identifier
    | | | | +--ro location* inet:uri
    | | | +--ro pkg:replaces-revision*
    | | | | yanglib:revision-identifier
    | +--ro schema* [name]
    | |
    | | +--ro name string
    | | +--ro module-set* -> ../../module-set/name
    | | +--ro pkg:package
    | | | +--ro pkg:name?
    | | | | -> /yanglib:yang-library/package/name
    | | | +--ro pkg:version? leafref
    | +--ro datastore* [name]
    | |
    | | +--ro name ds:datastore-ref
    | | +--ro schema -> ../../schema/name
    | +--ro content-id string
    | +--ro pkg:package* [name version]
    | |
    | | +--ro pkg:name yang:yang-identifier
    | | +--ro pkg:version yang-sem-ver
    | | +--ro pkg:revision-date?
    | | | yanglib:revision-identifier
    | | +--ro pkg:location* inet:uri
    | | +--ro pkg:description? string

```

```

|      +--ro pkg:reference?                string
|      +--ro pkg:previous-version?         yang-sem-ver
|      +--ro pkg:tag*                      tags:tag
|      +--ro pkg:referentially-complete?   boolean
|      +--ro pkg:mandatory-feature*        string
|      +--ro pkg:imported-packages* [name version]
|      |      +--ro pkg:name                yang:yang-identifier
|      |      +--ro pkg:version             yang-sem-ver
|      |      +--ro pkg:deviated?          boolean
|      +--ro pkg:module-set*
|          -> /yanglib:yang-library/module-set/name
x--ro modules-state
  x--ro module-set-id    string
  x--ro module* [name revision]
    x--ro name                yang:yang-identifier
    x--ro revision            union
    +--ro schema?             inet:uri
    x--ro namespace           inet:uri
    x--ro feature*            yang:yang-identifier
    x--ro deviation* [name revision]
      |      x--ro name                yang:yang-identifier
      |      x--ro revision            union
    x--ro conformance-type    enumeration
    x--ro submodule* [name revision]
      x--ro name                yang:yang-identifier
      x--ro revision            union
      +--ro schema?             inet:uri

notifications:
+---n yang-library-update
|   +--ro content-id    -> /yang-library/content-id
x---n yang-library-change
  x--ro module-set-id    -> /modules-state/module-set-id

```

Appendix B. 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.

B.1. Example IETF Network Device YANG package

This section provides an instance data document 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, they modules are referenced by revision date rather than version 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",
    "target-ptr": "TBD",
    "timestamp": "2018-12-13T17:00:00Z",
    "description": "Example IETF network device YANG package definiti\
\ion",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-ietf-network-device",
        "version": "1.1.2",
        "namespace": "urn:ietf:params:xml:ns:yang-pkg:ietf-network-d\
\evice",
        "location": "file://example.org/yang/packages/ietf-network-d\
\evice@v1.1.2.json",
        "description": "This package defines a small sample set of Y\
\ANG modules that could represent the basic set of modules that a st\
\andard network device might be expected to support.",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-11-26",
        "module": [
          {
            "name": "iana-crypt-hash",
            "revision": "2014-08-06",
            "namespace": "urn:ietf:params:xml:ns:yang:iana-crypt-has\
\h",
            "location": "https://raw.githubusercontent.com/YangModel\
\s/yang/master/standard/ietf/RFC/iana-crypt-hash%402014-08-06.yang"
```

```

    },
    {
      "name": "ietf-system",
      "revision": "2014-08-06",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-system",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-system%402014-08-06.yang"
    },
    {
      "name": "ietf-interfaces",
      "revision": "2018-02-20",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-interface\
\s",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-interfaces%402018-02-20.yang"
    },
    {
      "name": "ietf-netconf-acm",
      "revision": "2018-02-14",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-netconf-a\
\cm",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-netconf-acm%402018-02-14.yang"
    },
    {
      "name": "ietf-key-chain",
      "revision": "2017-06-15",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-key-chain\
\s",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-key-chain%402017-06-15.yang"
    },
    {
      "name": "ietf-ip",
      "revision": "2018-02-22",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-ip",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-ip%402018-02-22.yang"
    }
  ],
  "import-only-module": [
    {
      "name": "ietf-yang-types",
      "revision": "2013-07-15",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-type\
\s",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-yang-types%402013-07-15.yang"
    }
  ]
}

```

```

    },
    {
      "name": "ietf-inet-types",
      "revision": "2013-07-15",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-type\
\s",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/yang/master/standard/ietf/RFC/ietf-inet-types%402013-07-15.yang"
    }
  ]
}
}
}
}
}
<CODE ENDS>

```

B.2. Example IETF Basic Routing YANG package

This section provides an instance data document 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 version 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: '\n' line wrapping per BCP XX (RFC XXXX) =====

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ietf-routing-pkg",
    "target-ptr": "TBD",
    "timestamp": "2018-12-13T17:00:00Z",
    "description": "Example IETF routing YANG package definition",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-ietf-routing",

```

```

        "version": "1.3.1",
        "namespace": "urn:ietf:params:xml:ns:yang-pkg:ietf-routing",
        "location": "file://example.org/yang/packages/ietf-routing@v\
1.3.1.json",
        "description": "This package defines a small sample set of I\
ETF routing YANG modules that could represent the set of IETF routi\
ng functionality that a basic IP network device might be expected t\
o support.",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-11-26",
        "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",
                "namespace": "urn:ietf:params:xml:ns:yang:ietf-routing",
                "location": [
                    "https://raw.githubusercontent.com/YangModels/yang/mas\
ter/standard/ietf/RFC/ietf-routing@2018-03-13.yang"
                ]
            },
            {
                "name": "ietf-ipv4-unicast-routing",
                "revision": "2018-03-13",
                "namespace": "urn:ietf:params:xml:ns:yang:ietf-ipv4-unca\
st-routing",
                "location": [
                    "https://raw.githubusercontent.com/YangModels/yang/mas\
ter/standard/ietf/RFC/ietf-ipv4-unicast-routing@2018-03-13.yang"
                ]
            },
            {
                "name": "ietf-ipv6-unicast-routing",
                "revision": "2018-03-13",
                "namespace": "urn:ietf:params:xml:ns:yang:ietf-ipv6-unca\
st-routing",
                "location": [
                    "https://raw.githubusercontent.com/YangModels/yang/mas\
ter/standard/ietf/RFC/ietf-ipv6-unicast-routing@2018-03-13.yang"
                ]
            }
        ]
    }

```

```

    ]
  },
  {
    "name": "ietf-isis",
    "revision": "2018-12-11",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-isis"
  },
  {
    "name": "ietf-interfaces-common",
    "revision": "2018-07-02",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-interface\
\s-common"
  },
  {
    "name": "ietf-if-l3-vlan",
    "revision": "2017-10-30",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-if-l3-vla\
\n"
  },
  {
    "name": "ietf-routing-policy",
    "revision": "2018-10-19",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-routing-p\
\olicy"
  },
  {
    "name": "ietf-bgp",
    "revision": "2018-05-09",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-bgp"
  },
  {
    "name": "ietf-access-control-list",
    "revision": "2018-11-06",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-access-co\
\ntrol-list"
  }
],
"import-only-module": [
  {
    "name": "ietf-routing-types",
    "revision": "2017-12-04",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-routing-t\
\ypes",
    "location": [
      "https://raw.githubusercontent.com/YangModels/yang/master/standard/ietf/RFC/ietf-routing-types@2017-12-04.yang"
    ]
  }
],

```

```

        {
            "name": "iana-routing-types",
            "revision": "2017-12-04",
            "namespace": "urn:ietf:params:xml:ns:yang:iana-routing-t\
\ypes",
            "location": [
                "https://raw.githubusercontent.com/YangModels/yang/mas\
\ter/standard/ietf/RFC/iana-routing-types@2017-12-04.yang"
            ]
        },
        {
            "name": "ietf-bgp-types",
            "revision": "2018-05-09",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-bgp-types"
        },
        {
            "name": "ietf-packet-fields",
            "revision": "2018-11-06",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-packet-fi\
\elds"
        },
        {
            "name": "ietf-ethertypes",
            "revision": "2018-11-06",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-ethertype\
\s"
        }
    ]
}
}
}
}
<CODE ENDS>

```

B.3. Package import conflict resolution example

This section provides an example of how a package can resolve conflicting module versions 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 versions 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 versions 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",
    "description": "First imported example package",
    "content-data": {
      "ietf-yang-package: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",
            "version": "1.0.0"
          },
          {
            "name": "example-module-B",
            "version": "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",
    "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",
```

```
        "version": "1.2.3"
      },
      {
        "name": "example-module-E",
        "version": "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",
    "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",
        "imported-packages": [
          {
            "name": "example-import-1",
            "version": "1.0.0"
          },
          {
            "name": "example-import-2",
            "version": "2.0.0"
          }
        ],
        "module": [
          {
            "name": "example-module-A",
            "version": "1.2.3"
          }
        ]
      }
    }
  }
}
```

```
    "import-only-module": [  
      {  
        "name": "example-types-module-C",  
        "revision": "2018-11-26",  
        "replaces-revision": [ "2018-01-01 " ]  
      }  
    ]  
  }  
}  
}
```

Author's Address

Robert Wilton
Cisco Systems, Inc.

Email: rwilton@cisco.com

Network Working Group
Internet-Draft
Updates: 7950 (if approved)
Intended status: Standards Track
Expires: September 12, 2019

B. Claise
J. Clarke
R. Rahman
R. Wilton, Ed.
Cisco Systems, Inc.
B. Lengyel
Ericsson
J. Sterne
Nokia
K. D'Souza
AT&T
March 11, 2019

YANG Semantic Versioning for Modules
draft-verdt-netmod-yang-semver-00

Abstract

This document specifies a new YANG module update procedure using semantic version numbers, to allow for limited non-backwards-compatible changes, as an alternative proposal to module update rules in the YANG 1.1 specifications. This document updates RFC 7950, RFC 8407 and RFC 8525.

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 September 12, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Updates to YANG RFCs	4
1.1.1. Updates to RFC7950	4
1.1.2. Updates to RFC8525	4
1.1.3. Updates to RFC8407	5
1.2. Complementary solutions for the other requirements	5
2. YANG Semantic Versioning	6
2.1. Classification of changes between module revisions	6
2.2. YANG Semantic Versioning Scheme for Modules	7
2.2.1. Examples for YANG semantic version numbers	8
2.3. YANG Semantic Version Update Rules	10
2.4. YANG Module Semver Extension	11
3. Import by Semantic Version	13
3.1. Module import examples	15
4. Classifying changes in YANG modules	16
4.1. Editorial changes	16
4.2. Backwards-compatible changes	16
4.3. Non-backwards-compatible changes	17
5. Updates to ietf-yang-library	17
5.1. Advertising module version number	17
5.2. Resolving ambiguous module imports	17
5.3. Reporting how deprecated and obsolete nodes are handled	18
6. YANG status description extension	19
7. Semantic versioning of YANG instance data	19
8. Guidelines	20
8.1. Guidelines to YANG model authors	20
8.1.1. Use of YANG semantic versioning	20
8.1.2. Making non-backwards-compatible changes to a YANG module	21
8.1.2.1. Removing a data node	22
8.1.2.2. Changing the type of a leaf node	23
8.1.2.3. Reducing the range of a leaf node	23
8.1.2.4. Changing the key of a list	23
8.1.2.5. Renaming a node	23
8.1.2.6. Changing a default value	23
8.2. Guidelines to YANG model clients	24

9. Semantic Version Extension YANG Modules	24
10. Contributors	30
11. Security Considerations	31
12. IANA Considerations	31
12.1. YANG Module Registrations	31
13. References	32
13.1. Normative References	32
13.2. Informative References	32
Appendix A. Appendix	34
A.1. Open Issues	34
A.2. Derived Semantic Version	35
A.2.1. The Derived Semantic Version	35
A.2.2. Implementation Experience	35
Authors' Addresses	36

1. Introduction

This document defines a solution to the YANG module lifecycle problems described in [I-D.verdt-netmod-yang-versioning-reqs], covering all of the specified requirements except for requirements: 2.2, 3.1, and 3.2.

Specifically, this document recognises a need to sometimes allow YANG modules to evolve with non-backwards-compatible changes, which might end up breaking clients. The solution makes use of semantic version numbers to help manage the lifecycle of YANG modules.

The solution is comprised of the following seven parts:

- A definition for the YANG semantic versioning scheme for modules, and an explanation of how the semver extension can be used to annotate modules with their semantic version number.

- A YANG extension to allow YANG module imports to be restricted to modules with particular semantic versions, allowing inter-module version dependencies to be captured within YANG module definitions.

- Updates to the YANG 1.1 module update rules to accommodate the semantic versioning scheme.

- Updates and augmentations to ietf-yang-library to include the YANG semantic version number in the module descriptions, to report how 'deprecated' and 'obsolete' nodes are handled by a server, and to clarify how module imports are resolved when multiple versions could otherwise be chosen.

A YANG extension to add a 'description' statement to the YANG 'status' statement to allow additional documentation as to why a node is being deprecated, and what alternatives may be available.

A description of how YANG semantic versioning applies to YANG instance data.

Guidelines to YANG module authors on how the YANG semantic versioning rules should be used, along with examples.

Open issues are listed at Appendix A.1, and tracked at <https://github.com/netmod-wg/yang-ver-dt/issues>.

1.1. Updates to YANG RFCs

1.1.1. Updates to RFC7950

This document proposes updates to [RFC7950] to address some of the requirements. It should be noted that there is also active WG discussion on the next steps towards an updated version of YANG, and potentially some of the functionality described here could be folded into an updated revision of [RFC7950], although that might adversely impact when (parts of) a standards based YANG module versioning solution is available.

The sections listed below provide updates to [RFC7950]. The design team does not believe any of the changes require a new version of the YANG language. It is believed that the extensions as they are defined can coexist with existing YANG 1.1 clients.

- o Section 4 describes modification to the [RFC7950] Section 11 module update text to advise the use of semantic versioning as described in this document.
- o Section 3 describes an extension to do import by semantic version.
- o Section 6 defines an extension that adds a description child element to the YANG "status" statement.

1.1.2. Updates to RFC8525

This document updates [RFC8525]. Section 5 defines how a reader of a YANG library datastore schema chooses which version of an import-only module is used to resolve a module import when the definition is otherwise ambiguous.

1.1.3. Updates to RFC8407

Section 8 updates [RFC8407] to provide guidelines on how the YANG module semantic versioning can be used to manage the lifecycle of YANG modules when using strict RFC 7950 chapter 11 backwards compatibility rules are not pragmatic.

1.2. Complementary solutions for the other requirements

This section is to aid the WG understand how the full set of YANG versioning requirements are intended to be holistically addressed and is intended to be removed if this draft is adopted by the WG.

As stated previously, this draft does not address requirements 2.2, 3.1 and 3.2 of the requirements specified in [I-D.verdt-netmod-yang-versioning-reqs]. Instead, additional work is needed to address those requirements, which the design team believes would be best addressed in separate drafts. It is hoped that the WG agrees that viable solutions to the other requirements exist that complement the solution proposed in this draft, and thus this work can usefully progress in parallel. In particular, there is value to the industry to achieve standardization of a partial solution that addresses the majority, but not all, of the stated requirements, on the agreement that a full solution will follow.

The two additional drafts are:

A tooling based solution is proposed for requirement 2.2, that allows two YANG schema versions to be algorithmically compared, with the algorithm reporting the list of differences between the two YANG schema and whether each change is regarded as being editorial, backwards-compatible, or non-backwards-compatible. Annotations to the YANG modules, via the use of extension statements, may help improve the accuracy of the comparison algorithm, particularly for statements that are very hard for an algorithm to correctly classify the scope of any differences (e.g., a change in the semantic behaviour of a data node defined via modifications to the associated YANG description statement). Given that requirement 2.2 is a soft requirement (SHOULD rather than MUST), and practical experience with the tooling is required, it is proposed that this work is deferred at this time.

A proposed solution for requirements 3.1 and 3.2 is via the use of YANG packages [I-D.rwilton-netmod-yang-packages] and a protocol based version selection scheme that can be used by clients to choose a particular YANG datastore schema from the set of datastore schema that are supported by the server.

2. YANG Semantic Versioning

The chapter defines YANG Semantic Versioning, explains how it is used with YANG modules, and the rules associated with changing a module's semantic version number when the module definitions are updated.

The YANG semantic versioning scheme applies only to YANG modules. YANG submodules are not independently versioned by the YANG semantic versioning scheme. Instead, if a versioned module includes one or more submodules then those submodules are implicitly versioned as part of the module's 'semver:version' statements, and all the module's 'include' statements MUST specify the revision-date for each of the included submodules.

2.1. Classification of changes between module revisions

The principle aim of YANG semantic versioning is to allow a user of a YANG module to understand the overall significance of any changes between two module revisions solely based on the semantic version number.

The semantic version change between any two arbitrary revisions of a YANG module can be classified into one of four categories: 'unchanged', 'editorial', 'backwards-compatible' or 'non-backwards-compatible'. A summary of the classification is given below, with the specific rules as they apply to YANG statements provided in Section 4.

The semantic version change between two module revisions is defined as 'unchanged' if, after excluding 'revision' and 'semver:version' statements and their substatements, the only remaining changes are insignificant white space changes.

An 'editorial' module semantic version change is where there are changes in the module's statements, between the two module revisions, but those changes do not affect the syntax or semantic meaning of the module in any way. An example of an editorial change would be a fix to a spelling mistake in a description statement.

A 'backwards-compatible' module semantic version change is where some syntax or semantic changes exists between the two module revisions, but all changes follow the rules specified in Section 4.2.

A 'non-backwards-compatible' module semantic version change is where some syntax or semantic changes exists between the two

module revisions, and those changes do not follow the rules for a 'backwards-compatible' version change.

2.2. YANG Semantic Versioning Scheme for Modules

This document defines the YANG semantic versioning scheme that is used for YANG modules. 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 module lifecycles that cannot be addressed using the semver.org 2.0.0 versioning scheme alone.

Unlike the semver.org 2.0.0 versioning scheme, the YANG semantic versioning scheme supports limited updates to older versions of YANG modules, to allow for bug fixes and enhancements to module versions that are not the latest.

Module definitions that follow the semver.org 2.0.0 versioning scheme are fully compatible with implementations that understand the YANG semantic versioning scheme.

If module updates are always restricted to the latest version of the module only, then the version numbers used by the YANG semantic versioning scheme are exactly the same as those defined by the semver.org 2.0.0 versioning scheme.

Every YANG module versioned using the YANG semantic versioning scheme specifies the module's semantic version number by including the 'semver:module-version' statement according to the following rules:

The module **MUST** include at least one revision statement.

The most recent module revision statement **MUST** include a 'semver:module-version' sub-statement, that defines the module's YANG semantic version.

The preceding module revision statement **SHOULD** also include a 'semver:module-version' sub-statement, to allow the module's semantic version history to be derived.

All other revision statements **MAY** include a 'semver:module-version' sub-statement if they have an associated YANG semantic version.

"The YANG semver version number is expressed as a string of the form: 'X.Y.Zv'; where X, Y, and Z each represent non-negative integers smaller than 32768, and v represents an optional single character suffix: 'm' or 'M'.

- o '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.
- o '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 'm' or 'M' modifier.
- o 'Zv' 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 'v' takes:
 - * If the modifier letter is absent, the change represents an editorial change
 - * 'm' - the change represents a backwards-compatible change
 - * 'M' - the change represents a non-backwards-compatible change

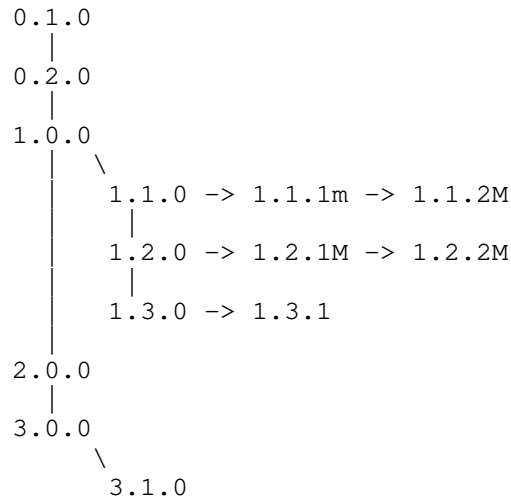
The YANG module name and YANG semantic version number uniquely identifies a revision of a module, with an associated revision date. There MUST NOT be multiple instances of a YANG module definition with the same module name and YANG semantic version number but different content or revision date.

There MUST NOT be multiple versions of a YANG module that have the same MAJOR, MINOR and PATCH version numbers, but different patch modifier letter. E.g., module version "1.2.3M" MUST NOT be defined if module version "1.2.3" has already been defined.

2.2.1. Examples for YANG semantic version numbers

The following diagram and explanation illustrates how YANG semantic version numbers work.

Example YANG semantic version numbers for an example module:



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

0.1.0 – first beta module version

0.2.0 – second beta 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)

1.3.0 – improve existing functionality, added leaf "foo-64" (BC)

1.3.1 – improve description wording for "foo-64" (Editorial)

1.1.1m – backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0 (BC)

2.0.0 – change existing model for performance reasons, e.g. re-key list (NBC)

1.1.2M – NBC point bug fix, not required in 2.0.0 due to model changes (NBC)

3.0.0 - NBC bugfix, rename "baz" to "bar"; also add new BC leaf "wibble"; (NBC)

1.2.1M - backport NBC fix, changing "baz" to "bar"

1.2.2M - backport "wibble". This is a BC change but "M" modifier is sticky.

3.1.0 - introduce new leaf "wobble" (BC)

The partial ordering relationships based on the semantic versioning numbers can be defined as follows:

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 2.0.0 < 3.0.0 < 3.1.0

1.0.0 < 1.1.0 < 1.1.1m < 1.1.2M

1.0.0 < 1.1.0 < 1.2.0 < 1.2.1M < 1.2.2M

There is no ordering relationship between 1.1.1M and either 1.2.0 or 1.2.1M, except that they share the common ancestor of 1.1.0.

Looking at the version number alone, the module definition in 2.0.0 does not necessarily contain the contents of 1.3.0. However, the module revision history in 2.0.0 would likely indicate that it was edited from module version 1.3.0.

2.3. YANG Semantic Version Update Rules

When a new revision of a module is produced, then the following rules define how the YANG semantic version number for the new module revision is calculated, based on the changes between the two module revisions, and the YANG semantic version number of the base module revision that the changes are derived from. A two step process is used:

The first step is to classify the module change as 'editorial', 'backwards-compatible', or 'non-backwards-compatible version' using the rules defined in Section 2.1 and Section 4.

The second step is to calculate the value of the 'semver:version' field for the new module revision, based on the value of the 'semver:version' field in the base module, any how the module changes have been classified.

The following rules define how the value for the 'semver:version' argument in the new module revision is calculated:

1. If a module is being updated in a non-backwards-compatible way, then the module version "X.Y.Z[m|M]" MUST be updated to "X+1.0.0" unless that module version has already been defined with different content, in which case the module version "X.Y.Z+1M" MUST be used instead.
2. If a module 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 module version MUST be updated to "X.Y+1.0", unless that module version has already been defined with different content, when the module version MUST be updated to "X.Y.Z+1m instead".
 - ii "X.Y.Zm" - the module version MUST be updated to "X.Y.Z+1m".
 - iii "X.Y.ZM" - the module version MUST be updated to "X.Y.Z+1M".
3. If a module 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 module version MUST be updated to "X.Y.Z+1"
 - ii "X.Y.Zm" - the module version MUST be updated to "X.Y.Z+1m".
 - iii "X.Y.ZM" - the module version MUST be updated to "X.Y.Z+1M".
4. YANG module semantic version numbers beginning with 0, i.e "0.X.Y" are regarded as beta 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.

2.4. YANG Module Semver Extension

This document defines a YANG extension to add the YANG module semantic version to a Module. The complete definition of this YANG module is in Section 9.

```
extension module-version {  
    argument semver;  
}
```

The extension would typically be used this way:

```
module yang-module-name {  
  
    namespace "name-space";  
    prefix "prefix-name";  
  
    import ietf-semver { prefix "semver"; }  
  
    description  
        "to be completed";  
  
    revision 2018-02-28 {  
        description "Added leaf 'wobble'";  
        semver:module-version "3.1.0";  
    }  
  
    revision 2017-12-31 {  
        description "Rename 'baz' to 'bar', added leaf 'wibble'";  
        semver:module-version "3.0.0";  
    }  
  
    revision 2017-10-30 {  
        description "Change the module structure";  
        semver:module-version "2.0.0";  
    }  
  
    revision 2017-08-30 {  
        description "Clarified description of 'foo-64' leaf";  
        semver:module-version "1.3.1";  
    }  
  
    revision 2017-07-30 {  
        description "Added leaf foo-64";  
        semver:module-version "1.3.0";  
    }  
  
    revision 2017-04-20 {  
        description "Add new functionality, leaf 'baz'";  
        semver:module-version "1.2.0";  
    }  
  
    revision 2017-04-03 {  
        description "Add new functionality, leaf 'foo'";  
        semver:module-version "1.1.0";  
    }  
  
    revision 2017-04-03 {
```

```
    description "First release version.";
    semver:module-version "1.0.0";
  }

  revision 2017-01-30 {
    description "NBC changes to initial revision";
    semver:module-version "0.2.0";
  }

  revision 2017-01-26 {
    description "Initial module version";
    semver:module-version "0.1.0";
  }

  //YANG module definition starts here
```

See also "Semantic Versioning and Structure for IETF Specifications" [I-D.claise-semver] for a mechanism to combine the semantic versioning, the GitHub tools, and a potential change to the IETF process.

3. Import by Semantic Version

RFC 7950 allows 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 overly burdensome 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 a revision date is also not ideal because the importing module may not work with all possible revisions of the imported module.

With semantic versioning, it is desirable for a importing module to specify the set of module versions of the imported module that are anticipated to be compatible.

This document specifies a YANG extension for selecting which versions of a module may be imported. It is designed around the assumption that most changes to a YANG module do not break importing modules, even if the changes themselves are not 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 it is augmented by another module.

The ietf-semver module defines the 'version' extension, a substatement to the YANG 'import' statement.

An 'import' statement MAY contain 'version' statements or a 'revision-date' statement, but not both.

The 'version' statement MAY be specified multiple times, requiring that the imported module version conforms to at least one of the 'version' statements.

The argument to the 'version' statement takes one of three valid forms:

1. "A.B.C" - import the exact module version that matches "A.B.C".
2. "A.B.C+" - import any module version that matches, or is greater than, "A.B.C".
3. "A.B.C-X.Y.Z" - import any module version that matches, or is greater than, "A.B.C"; and also matches, or is less than, "X.Y.Z". The word "MAX" can be used for 'Y' or 'Z' to represent the numerical value 32,767.

The rules for comparing module version numbers are as follows:

1. Version "R.S.T" matches version "A.B.C", only if
$$R = A, S = B, \text{ and } T = C$$
2. Version "R.S.T" is greater than version "A.B.C", only if
$$R = A, S = B, \text{ and } T > C; \text{ or}$$
$$R = A \text{ and } S > B; \text{ or}$$
$$R > A$$
3. Version "R.S.T" is less than version "X.Y.Z", only if
$$R = X, S = Y, \text{ and } T < Z; \text{ or}$$
$$R = X \text{ and } S < Y; \text{ or}$$
$$R < X$$

The patch modifier letter is not included as part of the 'semver:version' argument, and is entirely ignored for import statement module version number comparisons.

3.1. Module import examples

Consider an example module "example-module" that is hypothetically available in the following versions: 0.1.0, 0.2.0, 1.0.0, 1.1.0, 1.1.1m, 1.1.2M, 1.2.0, 1.2.1M, 1.2.2M, 1.3.0, 1.3.1, 2.0.0, 3.0.0, and 3.1.0. E.g. matching the versions illustrated in Section 2.2.1.

The first example selects the specific version 1.1.2M. A specific version import might be used if 1.1.2M contained changes that are incompatible with other versions.

```
import example-module {  
    semver:version 1.1.2;  
}
```

The next example selects module versions that match, or are greater than, version 1.2.0. This form may be used if there is a dependency on a data node introduced in version 1.2.0. This is expected to be the most commonly used form of 'import by version'.

Includes versions: 1.2.0, 1.2.1M, 1.2.2M, 1.3.0, 1.3.1, 2.0.0, 3.0.0 and 3.1.0.

```
import example-module {  
    semver:version 1.2.0+;  
}
```

The next example selects module versions that match, or are greater than 1.1.0, but excluding all 1.1.x and 1.2.x 'M' versions. This form may be needed if structural non backwards compatible changes are introduced in a patch 'M' version. Generally, it is advisable to avoid making such changes.

Includes versions: 1.1.0, 1.1.1m, 1.2.0, 1.3.0, 1.3.1, 2.0.0, 3.0.0, and 3.1.0.

```
import example-module {  
    semver:version 1.1.0-1.1.1;  
    semver:version 1.2.0;  
    semver:version 1.3.0+;  
}
```

The last example selects all module versions with a major version number of 1. This form may be useful if significant non backwards compatible changes have been introduced in version 2.0.0 that break import backwards compatibility.

Includes versions: 1.0.0, 1.1.0, 1.1.1m, 1.1.2M, 1.2.0, 1.2.1M, 1.2.2M, 1.3.0 and 1.3.1.

```
import example-module {  
    semver:version 1.0.0-1.MAX.MAX;  
}
```

4. Classifying changes in YANG modules

[RFC7950] chapter 11 defines the rules for what constitutes a backwards compatible change in YANG 1.1. However, the YANG semantic versioning scheme defined in this document uses a slightly modified version of this scheme, and also provides rules to classify changes as editorial, backwards-compatible, or non-backwards-compatible.

4.1. Editorial changes

Any changes that do not change the ordering or meaning of the YANG module in any way are classified as 'editorial'. The following rules define 'editorial':

- o Changing any 'description' statement if it does not change the semantic meaning of the statement it relates to. E.g., fixing spelling or grammar, or changing layout, are all allowed.
- o Adding or updating 'reference' statements.
- o Adding or updating the 'organization' statement.
- o Adding a new 'revision' or 'semver:module-version' statement, or correcting a previous 'revision' or 'semver:module-version' statement.
- o A module may be split into a set of submodules or a submodule may be removed, provided the definitions in the module do not change except in the ways described above.

4.2. Backwards-compatible changes

[RFC7950] chapter 11 defines the rules for what constitutes a backwards-compatible change in YANG 1.1. The document updates these rules in the following ways:

- o Adding or changing a 'status' node to 'obsolete' is not a backwards-compatible change. Other changes/additions of status elements are backwards-compatible, as per [RFC7950].

- o Changing the ordering of statements is allowed if it does not change the ordering of an rpc's 'input' substatements.

4.3. Non-backwards-compatible changes

All other changes to YANG modules that are not classified as 'editorial' or 'backwards-compatible' are defined as being non-backwards-compatible.

Examples of non-backwards-compatible changes include:

- o Deleting a data node, or changing it to status obsolete.
- o Changing the name, type, or units of a data node.
- o Modifying the description in a way that changes the semantic meaning of the data node.
- o 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.
- o Adding or modifying 'when' statements that reduce when the data node is available in the schema.
- o Making the statement conditional on if-feature.

5. Updates to ietf-yang-library

YANG library [RFC7895] [RFC8525] is modified to support semantic versioning in three ways.

5.1. Advertising module version number

The ietf-semver YANG module augments the 'module' list in ietf-yang-library with a 'version' leaf to optionally declare the YANG semantic version of each module.

5.2. 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] that only a single revision of a YANG module may be implemented.

If a YANG module import statement does not specify a specific version or 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 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, but one of more modules revisions specify a YANG semantic version, then the import MUST resolve to the module with the greatest version number, according to the version comparison rules in Section 3.

If a module import statement could resolve to more than one module revision defined in the datastore schema, none of those revisions are implemented, and none of the modules revisions have a YANG semantic version number, then the import MUST resolve to the module that has the most recent revision date.

5.3. Reporting how deprecated and obsolete nodes are handled

The ietf-semver YANG module augments YANG library with two leaves to allow a server to report how it handles status 'deprecated' and status 'obsolete' nodes. The leaves are:

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

obsolete-nodes-absent: If present, this leaf indicates that the server does not implement any status 'obsolete' nodes. If this leaf is absent then the behaviour is unspecified.

Implementations that implement the YANG semantic versioning scheme defined in this document MUST set the 'deprecated-nodes-implemented' leaf because the refined module update rules in Section 4 require that this is how servers handle 'deprecated' and 'obsolete' nodes to comply with YANG module semantic versioning.

If a server does not set the 'deprecated-nodes-implemented' leaf, then clients MUST NOT rely solely on the YANG module semantic version number to determine whether two module versions 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. YANG status description extension

The ietf-semver module specifies the YANG extension 'status-description' that can be used as a substatement of the status statement. The argument to this extension can contain freeform text to help readers of the module understand why the node was deprecated or made obsolete, when it is anticipated that the node will no longer be available for use, and potentially reference other schema elements that can be used instead. An example is shown below.

```
leaf imperial-temperature {
  type int64;
  units "degrees Fahrenheit";
  status deprecated {
    semver:status-description
      "Imperial measurements are being phased out in favor
       of their metric equivalents. Use metric-temperature
       instead.";
  }
  description
    "Temperature in degrees Fahrenheit.";
}
```

7. Semantic versioning of YANG instance data

Instance data sets [I-D.ietf-netmod-yang-instance-file-format] do not have an associated YANG semantic version, as compatibility for instance data is undefined.

However, instance data may reference an associated YANG schema, and that schema could make use of semantic version numbers, both for the individual YANG modules that comprise the schema, and potentially for the entire schema itself (e.g., [I-D.rwilton-netmod-yang-packages]).

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

One common scenario, where instance data may have to cope with changes to the schema is for the <startup> datastore when a server is

restarted with a different YANG schema (e.g. due to a software upgrade or downgrade). How a server restores the configuration from <startup> during such upgrades or downgrades is outside the scope of this specification.

8. Guidelines

8.1. Guidelines to YANG model authors

NBC changes to YANG models may cause problems to clients, who are consumers of YANG models, and SHOULD be avoided. However, there are cases where NBC changes are required, e.g. to fix an incorrect YANG model.

YANG model authors are recommended to minimize NBC changes and keep changes BC whenever possible.

The use of status "deprecated" with the status-description statement allows clients to plan a migration to alternative data nodes.

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

8.1.1. Use of YANG semantic versioning

Module authors should use the following guidance when applying the module version update rules specified in Section 2.3.

Updates to modules SHOULD be applied to the latest version of YANG modules, avoiding the use of the 'm|M' patch modifier. When used in this way, the YANG semantic version numbers are compatible with the versioning scheme defined by the semver.org 2.0.0 rules.

Changes to older versions of published YANG modules SHOULD be minimized, since there may be a greater impact on clients, and comparing between version numbers becomes more limited if the 'm|M' modifiers are used. However, if it is necessary to make such changes then the following guidelines apply:

Any changes SHOULD also be made to a new latest version of the YANG module, if appropriate.

Where possible, changes SHOULD be restricted to backwards-compatible changes only.

NBC changes MAY be made, subject to the constraints defined in Section 2.3. The impact to clients SHOULD be carefully considered and minimized if possible.

The version numbers associated with a module MUST never be reused. E.g., when updating module version 3.4.0 in a NBC manner the module author must verify whether version 4.0.0 is available for use and if that version was already used, the updated module must use version 3.4.1M instead.

Patch modifier letters (i.e. 'm' or 'M') are sticky. For example if version 3.4.1M is modified in a BC way, the next version is 3.4.2M. This is to indicate that 3.4.2M is not BC with 3.4.0, however it comes at the cost of not being able to indicate the type of change between 3.4.1M and 3.4.2M.

As explained in Appendix A.2.2, while programatically determining a semantic version is possible using tools (e.g. the pyang utility), human oversight is highly recommended because of some special cases which can not be detected by tools. Therefore, a model author SHOULD use both means to determine a model's semantic version.

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

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

- o If the server can support NBC versions of the YANG module simultaneously using version selection, then the NBC changes MAY be done immediately. Clients would be required to select the version which they support and the NBC change would have no impact on them.
- o When possible, NBC changes are done incrementally to provide clients time to adapt to NBC changes.

Here are some guidelines on how non-backwards compatible changes can be made incrementally:

1. The changes should be made incrementally, 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" and then when support is removed its status MUST be changed to "obsolete". Instead of using the "obsolete" status, the data node MAY be removed from the model but this has the risk of breaking modules which import the modified module.

2. A node with status "deprecated" MUST be supported for the solution described here to function properly.
3. A node with status "deprecated" SHOULD be available for at least one year before its status is changed to "obsolete", see Section 4.7 of [RFC8407].
4. Support for a node which is "obsolete" is indicated by the node "obsolete-nodes-present", see Section 5.
5. The new extension "status-description" SHOULD be used for nodes which are "obsolete" or "deprecated".
6. For status "deprecated", the "status-description" SHOULD also indicate until when support for the node is guaranteed. If there is a replacement data node, rpc, action or notification for the deprecated node, this SHOULD be stated in the "status-description".
7. When obsoleting or deprecating data nodes, the "deprecated" or "obsolete" status SHOULD be applied at the highest possible level in the data tree. For example, when deprecating all data nodes in a container, the "deprecated" status SHOULD be applied to the container. For clarity, the status MAY be added in all the affected nodes but the status-description SHOULD be added only at the highest level in the tree.

The following sections have examples on how non-backwards-compatible changes can be made.

8.1.2.1. 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 node's status SHOULD be changed to "deprecated" and it MUST be supported for at least one year. This is a backwards-compatible change.
2. When the node is not available anymore, its status MUST be changed to "obsolete" and the "status-description" updated, this is a non-backwards-compatible change. The "status-description" SHOULD be used to explain why the node is not available anymore.

8.1.2.2. Changing the type of a leaf node

Changing the type of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed to a string:

1. The status of node "vpn-id" SHOULD be changed to "deprecated" and the node SHOULD be available for at least one year. This is a backwards-compatible change.
2. A new 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 SHOULD explain that it is replacing node "vpn-id".
3. During the period of time where both nodes are available, how the server behaves when either node is set 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). The new node MAY have a when statement to achieve this. The old node MUST NOT have a when statement since this would be a non-backwards-compatible change, but the server MAY 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 MAY map the value. For example, if the new node "vpn-name" has value "123" then the server MAY return integer value 123 for the old node "vpn-id". However, if the value can not be mapped, we need a way of returning "unsupported" TBD.
4. When node "vpn-id" is not available anymore, its status MUST be changed to "obsolete" and the "status-description" is updated. This is a non-backwards-compatible change.

8.1.2.3. Reducing the range of a leaf node

8.1.2.4. Changing the key of a list

8.1.2.5. Renaming a node

8.1.2.6. Changing a default value

8.2. Guidelines to YANG model clients

Guidelines for clients of modules using YANG semantic versioning:

- o 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
- o Clients SHOULD monitor changes to published YANG modules through their version numbers, and use appropriate tooling to understand the specific changes between module versions. In particular, clients SHOULD NOT migrate to NBC versions of a module without first understanding the specifics of the NBC changes.
- o 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.

9. Semantic Version Extension YANG Modules

YANG module with extensions for defining a module's YANG semantic version number, and importing by version.

```
<CODE BEGINS> file "ietf-semver@2019-02-07.yang"
module ietf-semver {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-semver";
  prefix semver;

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

     Author:  Benoit Claise
              <mailto:bclaise@cisco.com>

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

     Author:  Reshad Rahman
              <mailto:rrahman@cisco.com>

     Author:  Robert Wilton
```

```
<mailto:rwilton@cisco.com>

Author:   Kevin D'Souza
         <mailto:kd6913@att.com>

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

Author:   Jason Sterne
         <mailto:jason.sterne@nokia.com>";
description
  "This module contains a definition for a YANG 1.1 extension to
  express the semantic version of YANG modules.";

revision 2019-02-27 {
  description
    "* Move YANG library augmentations into a separate module.
    * Update references.";
  reference
    "draft-verdt-netmod-yang-semver:
    YANG Semantic Versioning for Modules";
  semver:module-version "0.3.0";
}

revision 2018-04-05 {
  description
    "* Properly import ietf-yang-library.
    * Fix the name of module-semver => module-version.
    * Fix regular expression syntax.
    * Augment yang-library with booleans as to whether or not
      deprecated and obsolete nodes are present.
    * Add an extension to enable import by semantic version.
    * Add an extension status-description to track deprecated
      and obsolete reasons.
    * Fix yang-library augments to use 7895bis.";
  reference
    "draft-clacla-netmod-yang-model-update:
    New YANG Module Update Procedure";
  semver:module-version "0.2.1";
}
revision 2017-12-15 {
  description
    "Initial revision.";
  reference
    "draft-clacla-netmod-yang-model-update:
    New YANG Module Update Procedure";
  semver:module-version "0.1.1";
}
```

```
typedef version {
  type string {
    pattern '[0-9]{1,5}\.[0-9]{1,5}\.[0-9]{1,5}(m|M)?';
  }
  description
    "The type used to represent a YANG semantic version number.

    The YANG semver version number is expressed as a string of the
    form: 'X.Y.Zv'; where X, Y, and Z each represent non-negative
    integers smaller than 32768, and v represents an optional
    single character suffix: 'm' or 'M'.

    o '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.

    o '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.

    o 'Zv' 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 'v' takes:

      * 'M' - the change represents a non-backwards-compatible
        change

      * 'm' - the change represents a backwards-compatible change

      * If the modifier letter is absent, the change represents an
        editorial change";

  reference
    "draft-verdt-netmod-yang-semver: YANG Semantic Versioning";
}

extension module-version {
  argument semver;
  description
    "The version number for the module revision it is used in.

    This format of the argument matches the type version.

    The rules for updating the module-version number are described
    in section XXX of 'YANG Semantic Versioning for Modules';
```

By comparing the module-version between two revisions of a given module, one can determine if different revisions are backwards compatible or not, as well as whether or not new features have been added to a newer revision.

If a module contains this extension it indicates that for this module the updated status and update rules as this described in RFC XXXX are used.

The statement MUST only be a substatement of the 'revision' statements. Zero or one module-version statement is allowed per parent statement. No substatements are allowed.

'revision' statements in submodules MAY contain a 'module-version' statement for documentation purposes, but its meaning is undefined, and has no effect on the including module's semantic version.";

reference

```
"draft-verdt-netmod-yang-semver:
  YANG Semantic Versioning for Modules";
```

```
}
```

```
extension import-versions {
  argument version-clause;
  description
```

```
"This extension specifies an acceptable set of semantic
  versions of a given module that may be imported.
```

The statement MUST only be a substatement of the import statement.

The statement MUST NOT be present if the import has a revision-date substatement.

The statement MUST NOT be present if the imported module does not support semantic versioning.

Zero or more versions statements are allowed per parent statement. No substatements are allowed.

The version-clause argument MUST follow one of the below patterns:

```
(i)  "+" \d+\.\d+\.\d+ '+'
```

Matches exact version, e.g. 3.6.1

```
(ii) "+ '\d+\.\d+\.\d+\+ '+'
```

Matches exact version or greater, e.g. 3.6.1+

```
(iii) "+" \d+\.\d+\.\d+[-\d+\.\d+|MAX)\.(\d+|MAX)+ '+'
Matches inclusive range,
e.g. 3.6.1-7.8.4, or 3.2.1-3.MAX.MAX";
```

```
reference
  "draft-verdt-netmod-yang-semver: Import by Semantic Version";
}

extension status-description {
  argument description;
  description
    "Freeform text that describes why a given node has been
    deprecated or made obsolete. This may point to other schema
    elements that can be used in lieu of the given node.

    This statement MUST only be used as a substatement of the
    status statement

    Zero or more status-description statements are allowed per
    parent statement. No substatements are allowed.";
  reference
    "draft-verdt-netmod-yang-semver: YANG status description
    extension";
}
}
<CODE ENDS>
```

YANG module with augmentations to YANG Library to support semantic version numbers.

```
<CODE BEGINS> file "ietf-yl-semver@2019-02-07.yang"
module ietf-yl-semver {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yl-semver";
  prefix yl-semver;

  import ietf-semver {
    prefix semver;
  }

  import ietf-yang-library {
    prefix yanglib;
  }

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

```
WG List:  <mailto:netmod@ietf.org>

Author:   Benoit Claise
          <mailto:bclaise@cisco.com>

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

Author:   Reshad Rahman
          <mailto:rrahman@cisco.com>

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

Author:   Kevin D'Souza
          <mailto:kd6913@att.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 semantic version numbers and to provide an indication of
  how deprecated and obsolete nodes are handled by the server.";

semver:module-version "0.1.0";

revision 2019-02-27 {
  description
    "Moved YANG library augmentations into a separate module.";
  reference
    "draft-verdt-netmod-yang-semver:
    YANG Semantic Versioning for Modules";
  semver:module-version "0.1.0";
}

augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
  description
    "Augmentation modules with a semantic version.";
  leaf version {
    type semver:version;
    description
      "The semantic version for this module. The version MUST
      match the semver:version value in specific revision of the
      module loaded in this module-set.";
    reference
```



```
        "draft-verdt-netmod-yang-semver: YANG Semantic Versioning";
    }
}

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 empty;
        description
            "If present, this leaf indicates that all schema nodes with a
            status 'deprecated' child statement are implemented
            equivalently as if they had status 'current', or otherwise
            deviations MUST be used to explicitly remove 'deprecated'
            nodes from the schema.  If this leaf is absent then the
            behavior is unspecified.";
        reference
            "draft-verdt-netmod-yang-semver: Reporting how deprecated and
            obsolete nodes are handled";
    }
    leaf obsolete-nodes-absent {
        type empty;
        description
            "If present, this leaf indicates that the server does not
            implement any status 'obsolete' nodes.  If this leaf is
            absent then the behaviour is unspecified.";
        reference
            "draft-verdt-netmod-yang-semver: Reporting how deprecated and
            obsolete nodes are handled";
    }
}
}
}
<CODE ENDS>
```

10. 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:

- o Balazs Lengyel
- o Benoit Claise
- o Ebben Aries

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

The initial revision of this document was refactored and built upon [I-D.claccla-netmod-yang-model-update].

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

11. Security Considerations

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

12. IANA Considerations

12.1. YANG Module Registrations

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

The ietf-semver module:

Name: ietf-semver

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

Prefix: semver

Reference: [RFCXXXX]

The ietf-yl-semver module:

Name: ietf-yl-semver

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yl-semver

Prefix: yl-semver

Reference: [RFCXXXX]

13. References

13.1. Normative References

- [I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 2018.
- [RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <<https://www.rfc-editor.org/info/rfc7895>>.
- [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>>.
- [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>>.

13.2. Informative References

- [I-D.clacla-netmod-model-catalog]
Clarke, J. and B. Claise, "YANG module for yangcatalog.org", draft-clacla-netmod-model-catalog-03 (work in progress), April 2018.
- [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.claise-semver]
Claise, B., Barnes, R., and J. Clarke, "Semantic Versioning and Structure for IETF Specifications", draft-claise-semver-02 (work in progress), January 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-02 (work in progress), February 2019.
- [I-D.openconfig-netmod-model-catalog]
Shaikh, A., Shakir, R., and K. D'Souza, "Catalog and registry for YANG models", draft-openconfig-netmod-model-catalog-02 (work in progress), March 2017.
- [I-D.rwilton-netmod-yang-packages]
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-00 (work in progress), December 2018.
- [openconfigsemver]
"Semantic Versioning for Openconfig Models",
<<http://www.openconfig.net/docs/semver/>>.
- [semver] "Semantic Versioning 2.0.0", <<https://www.semver.org>>.
- [yangcatalog]
"YANG Catalog", <<https://yangcatalog.org>>.

13.3. URIs

- [1] <https://github.com/netmod-wg/yang-ver-dt/issues/14>
- [2] <https://github.com/netmod-wg/yang-ver-dt/issues/11>
- [3] <https://github.com/netmod-wg/yang-ver-dt/issues/13>
- [4] <https://github.com/netmod-wg/yang-ver-dt/issues/12>
- [5] <https://github.com/netmod-wg/yang-ver-dt/issues/10>
- [6] <https://github.com/netmod-wg/yang-ver-dt/issues/9>
- [7] <https://github.com/netmod-wg/yang-ver-dt/issues/8>
- [8] <https://github.com/netmod-wg/yang-ver-dt/issues/7>
- [9] <https://github.com/netmod-wg/yang-ver-dt/issues/6>

- [10] <https://github.com/netmod-wg/yang-ver-dt/issues/5>
- [11] <https://github.com/netmod-wg/yang-ver-dt/issues/4>
- [12] <https://github.com/netmod-wg/yang-ver-dt/issues/15>
- [13] <https://github.com/netmod-wg/yang-ver-dt/issues/2>

Appendix A. Appendix

A.1. Open Issues

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

- o Do we need a new version of YANG? #14 [1]
- o Add guidance text about warning NBC changes might break imports #11 [2]
- o Add a naming convention for versioned YANG file#13 [3]
- o Define editorial, bc, nbc impact of adding, changing, removing extension stmts#12 [4]
- o How to version modules in IETF drafts (after they have been published at 1.0.0 or later#10 [5]
- o The solution does not strictly support semver 2.0.0#9 [6]
- o Are whitespace changes allow between two module instances with the same version (or revision)?#8 [7]
- o Do we assume that a module has an implicit semver if none as been specified?#7 [8]
- o Is changing the ordering of nodes an NBC change?#6 [9]
- o Should version statement be at top level or under revision statement?#5 [10]
- o Figure out whether changing the imports constitute a BC or NBC change#4 [11]
- o Does BC or NBC depend on whether the node is config true/false?#15 [12]
- o Status obsolete nodes#2 [13]

A.2. Derived Semantic Version

This temporary text is intended to be moved to a separate draft that describes the tool based approach for versioning YANG modules mentioned in Section 1.2.

A.2.1. The Derived Semantic Version

If an explicitly defined semantic version is not available in the YANG module, it is possible to algorithmically calculate a derived semantic version. This can be used for modules not containing a definitive semantic-version as defined in this document or as a starting value when specifying the definitive semantic-version. Be aware that this algorithm may sometimes incorrectly classify changes between the categories non-compatible, compatible or error-correction.

A.2.2. Implementation Experience

[yangcatalog] uses the pyang utility to calculate the derived-semantic-version for all of the modules contained within the catalog. [yangcatalog] contains many revisions of the same module in order to provide its derived-semantic-version for module consumers to know what has changed between revisions of the same module.

Two distinct leafs in the YANG module

[I-D.clacla-netmod-model-catalog] contain this semver notation:

- o the semantic-version leaf contains the value embedded within a YANG module (if it is available).
- o the derived-semantic-version leaf is established by examining the the YANG module themselves. As such derived-semantic-version only takes syntax into account as opposed to the meaning of various elements when it computes the semantic version.
- o The algorithm used to produce the derived-semantic-version is as follows:
 1. Order all modules of the same name by revision from oldest to newest. Include module revisions that are not available, but which are defined in the revision statements in one of the available module versions.
 2. If module A, revision N+1 has failed compilation, bump its derived semantic MAJOR version. For unavailable module versions assume non-backward compatible changes were done., thus bump its derived semantic MAJOR version.

3. Else, run "pyang --check-update-from" on module A, revision N and revision N+1 to see if backward-incompatible changes exist.
4. If backward-incompatible changes exist, bump module A, revision N+1's derived MAJOR semantic version.
5. If no backward-incompatible changes exist, compare the pyang trees of module A, revision N and revision N+1.
6. If there are structural differences (e.g., new nodes), bump module A, revision N+1's derived MINOR semantic version.
7. If no structural differences exist, bump module A, revision N+1's derived PATCH semantic version.

The pyang utility checks many of the points listed in section 11 of [RFC7950] for known module incompatibilities. While this approach is a good way to programmatically obtain a semantic version number, it does not address all cases whereby a major version number might need to be increased. For example, a node may have the same name and same type, but its meaning may change from one revision of a module to another. This represents a semantic change that breaks backward compatibility, but the above algorithm would not find it. Therefore, additional, sometimes manual, rigor must be done to ensure a proper version is chosen for a given module revision.

Authors' Addresses

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

Joe Clarke
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

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Balazs Lengyel
Ericsson
Magyar Tudosok Korutja
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
United States of America

Email: kd6913@att.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 27, 2019

J. Clarke, Ed.
Cisco Systems, Inc.
November 23, 2018

YANG Module Versioning Requirements
draft-verdt-netmod-yang-versioning-reqs-02

Abstract

This document describes the problems that can arise because of the YANG language module update rules, that require all updates to YANG module preserve strict backwards compatibility. It also defines the requirements on any solution designed to solve the stated problems. This document does not consider possible solutions, nor endorse any particular solution.

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 May 27, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Background	2
2.1. Striving for model perfection	3
2.2. Some YANG Modules Are Not Backwards-Compatible	3
2.3. Non-Backwards-Compatible Errors	4
2.4. No way to easily decide whether a change is Backwards-Compatible	4
2.5. No good way to specify which module revision to import	5
2.6. Early Warning about Removal	6
2.7. Clear Indication of Node Support	6
3. Terminology and Conventions	7
4. The Problem Statement	7
5. Requirements of a YANG Versioning Solution	9
6. Contributors	11
7. Acknowledgments	11
8. Security Considerations	11
9. IANA Considerations	12
10. References	12
10.1. Normative References	12
10.2. Informative References	12
Author's Address	12

1. Introduction

This requirements document initially considers some of the existing YANG module update rules, then describes the problems that arise due to those rules embracing strict backwards compatibility, and finally defines requirements on any solution that may be designed to solve these problems by providing an alternative YANG versioning strategy.

2. Background

The YANG data modeling language [RFC7950] specifies strict rules for updating YANG modules (see section 11 "Updating a Module"). Citing a few of the relevant rules:

1. "As experience is gained with a module, it may be desirable to revise that module. However, changes to published modules are not allowed if they have any potential to cause interoperability problems between a client using an original specification and a server using an updated specification."

2. "Note that definitions contained in a module are available to be imported by any other module and are referenced in "import" statements via the module name. Thus, a module name MUST NOT be changed. Furthermore, the "namespace" statement MUST NOT be changed, since all XML elements are qualified by the namespace."
3. "Otherwise, if the semantics of any previous definition are changed (i.e., if a non-editorial change is made to any definition other than those specifically allowed above), then this MUST be achieved by a new definition with a new identifier."
4. "deprecated indicates an obsolete definition, but it permits new/continued implementation in order to foster interoperability with older/existing implementations."

The rules described above, along with other similar rules, causes various problems, as described in the following sections:

2.1. Striving for model perfection

The points made above lead to the logical conclusion that the standardized YANG modules have to be perfect on day one (at least the structure and meaning), which in turn might explain why IETF YANG modules take so long to standardize. Shooting for perfection is obviously a noble goal, but if the perfect standard comes too late, it doesn't help the industry.

2.2. Some YANG Modules Are Not Backwards-Compatible

As we learn from our mistakes, we're going to face more and more non-backwards-compatible YANG modules. An example is the YANG data model for L3VPN service delivery [RFC8049], which, based on implementation experience, has been updated in a non-backwards-compatible way by [RFC8299].

While Standards Development Organization (SDO) YANG modules are obviously better for the industry, we must recognize that many YANG modules are actually generated YANG modules (for example, from internal databases), which is sometimes the case for vendor modules [RFC8199]. From time to time, the new YANG modules are not backwards-compatible.

Old module parts that are no longer needed, no longer supported, or are not used by consumers need to be removed from modules. It is often hard to decide which parts are no longer needed/used; still the need and practice of removing old parts exist. While it is rare in standard modules it is more common in vendor YANG modules where the usage of modules is more controlled.

The problems described in Section 2.7 may also result in incompatible changes.

In such cases, it would be better to indicate how backwards-compatible a given YANG module actually is.

As modules are sometimes updated in an incompatible way the current assumption that once a YANG module is defined all further revisions can be freely used as they are compatible is not valid.

2.3. Non-Backwards-Compatible Errors

Sometimes small errors force us to make non-backwards-compatible updates. As an example imagine that we have a string with a complex pattern (e.g., an IP address). Let's assume the initial pattern incorrectly allows IP addresses to start with 355. In the next version this is corrected to disallow addresses starting with 355. Formally this is a non-backwards-compatible change as the value space of the string is decreased. In reality an IP address and the implementation behind it was never capable of handling an address starting with 355. So practically this is a backwards-compatible change, just like a correction of the description statement. Current YANG rules are ambiguous as to whether non-backwards-compatible bug fixes are allowed without also requiring a module name change.

2.4. No way to easily decide whether a change is Backwards-Compatible

A management system, SDN controller, or any other user of a module should be capable of easily determining the compatibility between two module versions. Higher level logic for a network function, something that cannot be implemented in a purely model driven way, is always dependent on a specific version of the module. If the client finds that the module has been updated on the network node, it has to decide if it tries to handle it as it handled the previous version of the model or if it just stops, to avoid problems. To make this decision the client needs to know if the module was updated in a backwards-compatible way or not.

This is not possible to decide today because of the following:

- o It is sometimes necessary to change the semantic behavior of a data node, action or rpc while the YANG definition does not change (with the possible exception of the description statement). In such a case it is impossible to determine whether the change is backwards-compatible just by looking at the YANG statements. It's only the human model designer who can decide.

- o Problems with the deprecated and obsolete status statement, Section 2.7
- o YANG module authors might decide to violate YANG 1.1 update rules for some of the reasons above.

Finding status changes or violations of update rules need a line-by-line comparison of the old and new modules is a tedious task.

2.5. No good way to specify which module revision to import

If a module (MOD-A) is imported by another one (MOD-B) the importer may specify which revision must be imported. Even if MOD-A is updated in a backwards-compatible way not all revisions will be suitable, e.g., a new MOD-B might need the newest MOD-A. However, both specifying or omitting the revision date for import leads to problems.

If the import by revision-date is specified

- o If corrections are made to MOD-A these would not have any effect as the import's revision date would still point to the uncorrected earlier YANG module revision.
- o If MOD-A is updated in a backwards-compatible way because another importer (MOD-C) needs some functionality, the new MOD-A could be used by MOD-B, but specifying the exact import revision-date prevents this. This will force the implementers to import two different revisions of MOD-A, forcing them to maintain old MOD-A revisions unnecessarily.
- o If multiple modules import different revisions of MOD-A the human user will need to understand the subtle differences between the different revisions. Small differences would easily lead to operator mistakes as the operator will rarely check the documentation.
- o Tooling/SW is often not prepared to handle multiple revisions of the same YANG module.

If the import revision-date is not specified

- o any revision of MOD-A may be used including unsuitable ones. Older revisions may be lacking functionality MOD-B needs. Newer MOD-A revisions may obsolete definitions used by MOD-B in which case these must not be used by MOD-B anymore.

- o As it is not specified which revisions of MOD-A are suitable for MOD-B. The problem has to be solved on a case by case basis studying all the details of MOD-A and MOD-B which is considerable work.

2.6. Early Warning about Removal

If a schema part is considered old/bad we need to be able to give advance warning that it will be removed. As this is an advance warning the part must still be present and usable in the current revision; however, it will be removed in one of the next revisions. The deprecated statement cannot be reliably used for this purpose both because deprecated nodes may not be implemented and also there is no mandate that text be provided explaining the deprecation.

We need the advance warning to allow users of the module time to plan/execute migration away from the deprecated functionality. Deprecation should be accompanied by information whether the functionality will just disappear or that there is an alternative, possibly more advanced solution that should be used.

Vendors use such warnings often, but the NMDA related redesign of IETF modules is also an example where it would be useful for IETF. As another example, see the usage of deprecated in the Java programming language.

2.7. Clear Indication of Node Support

The current definition of deprecated and obsolete in [RFC7950] (as quoted below) is problematic and should be corrected.

- o "deprecated" indicates an obsolete definition, but it permits new/continued implementation in order to foster interoperability with older/existing implementations.
- o "obsolete" means that the definition is obsolete and SHOULD NOT be implemented and/or can be removed from implementations.

YANG is considered an interface contract between the server and the client. The current definitions of deprecated and obsolete mean that a schema node that is either deprecated or obsolete may or may not be implemented. The client has no way to find out which is the case except for by trying to write or read data at the leaf in question. This probing would need to be done for each separate data-node, which is not a trivial thing to do. This "may or may not" is unacceptable in a contract. In effect, this works as if there would be an if-feature statement on each deprecated schema node where the server

does not advertise whether the feature is supported or not. Why is it not advertised?

3. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In addition, this document uses the following terminology:

- o YANG module revision: An instance of a YANG module, with no implied ordering or backwards compatibility between different revisions of the same module."
- o YANG module version: A YANG module revision, but also with an implied partial ordering relationship between other versions of the same module. Each module version must be uniquely identifiable.
- o Non-backwards-compatible (NBC): In the context of this document, the term 'non-backwards-compatible' refers to a change or set of changes between two YANG module revisions that do not adhere to the list of allowable changes specified in Section 11 "Updating a Module" of [RFC7950], with the following additional clarification:
 - * Any addition of, or change to, a "status" statement that allows a server to remove support for a schema node is considered a non-backwards-compatible change

4. The Problem Statement

Considering the issues described in the background, the problem definition can be summarized as follows.

Development of data models for a large collection of communication protocols and system components is difficult and typically only manageable with an iterative development process. Agile development approaches advocate evolutionary development, early delivery, and continual improvement. They are designed to support rapid and flexible response to change. Agile development has been found to be very successful in a world where the objects being modeled undergo constant changes.

The current module versioning scheme relies on the fundamental idea that a definition, once published, never changes its semantics. As a consequence, if a new definition is needed with different non-backwards-compatible semantics, then a new definition must be created

to replace the old definition. The advantage of this versioning scheme is that a definition identified by a module name and a path has fixed semantics that never change. (The details are a bit more nuanced but we simplify things here a bit in order to get the problems worked out clearly.)

There are two main disadvantages of the current YANG versioning scheme:

- o Any non-backwards-compatible change of a definition requires either a new module name or a new path. This has been found costly to support in implementations, in particular on the client side.
- o Since non-backwards-compatible changes require either a new module name or a new path, such changes will impact other modules that import definitions. In fact, with the current module versioning scheme other modules have to opt-in in order to use the new version. This essentially leads to a ripple effect where a non-backwards-compatible change of a core module causes updates on a potentially large number of dependent modules.

Other problems experienced with the current YANG versioning scheme are the following:

- o YANG has a mechanism to mark definitions deprecated but it leaves it open whether implementations are expected to implement deprecated definitions and there is no way (other than trial and error) for a client to find out whether deprecated definitions are supported by a given implementation.
- o YANG does not have a robust mechanism to document which data definitions have changed and to provide guidance how implementations should deal with the change. While it is possible to have this described in general description statements, having these details embedded in general description statements does not make this information accessible to tools.
- o YANG data models often do not exist in isolation and they interact with other software systems or data models that often do allow (controlled) non-backwards-compatible changes. In some cases, YANG models are mechanically derived from other data models that do allow (controlled) non-backwards-compatible changes. In such situations, a robust mapping to YANG requires to have version numbers exposed as part of the module name or a path definition, which has been found to be expensive on the client side (see above).

Given the need to support agile development processes and the disadvantages and problems of the current YANG versioning scheme described above, it is necessary to develop requirements and solutions for a future YANG versioning scheme that better supports agile development processes, whilst retaining the ability for servers to handle clients using older versions of YANG modules.

5. Requirements of a YANG Versioning Solution

The following is a list of requirements that a solution to the problems mentioned above MUST or SHOULD have. The list is grouped by similar requirements but is not presented in a set priority order.

1. Requirements related to making non-backwards-compatible updates to modules:
 - 1.1 A mechanism is REQUIRED to update a module in a non-backwards-compatible way without forcing all modules with import dependencies on the updated module from being updated at the same time (e.g. to change its import to use a new module name).
 - 1.2 Non-backwards-compatible updates of a module MUST not impact clients that only access data nodes of the module that have either not been updated or have been updated in backwards-compatible ways.
 - 1.3 A refined form of YANG's 'import' statement MUST be provided that is more restrictive than "import any revision" and less restrictive than "import a specific revision". Once non-backwards-compatible changes to modules are allowed, the refined import statement is used to express the correct dependency between modules.
 - 1.4 The solution MUST allow for backwards-compatible enhancements and bug fixes, as well as non-backwards-compatible bug fixes in non-latest-release modules.
2. Requirements related to identifying changes between different module revisions:
 - 2.1 Readers of modules, and tools that use modules, MUST be able to determine whether changes between two revisions of a module constitute a backwards-compatible or non-backwards-compatible version change. In addition, it MAY be helpful to identify whether changes represent bug fixes, new functionality, or both.

- 2.2 A mechanism SHOULD be defined to determine whether data nodes between two arbitrary YANG module revisions have (i) not changed, (ii) changed in a backwards-compatible way, (iii) changed in a non-backwards-compatible way.
3. Requirements related to supporting existing clients in a backwards-compatible way:
 - 3.1 The solution MUST provide a mechanism to allow servers to support existing clients in a backwards-compatible way.
 - 3.2 The solution MUST provide a mechanism to support clients that expect an older version of a given module when the current version has had non-backwards-compatible changes.
 - 3.3 Clients are expected to be able to handle unexpected instance data resulting from backwards-compatible changes.
4. Requirements related to managing and documenting the life cycle of data nodes:
 - 4.1 A mechanism is REQUIRED to allow a client to determine whether deprecated nodes are implemented by the server.
 - 4.2 If a data node is deprecated or obsolete then it MUST be possible to document in the YANG module what alternatives exist, the reason for the status change, or any other status related information.
 - 4.3 A mechanism is REQUIRED to indicate that certain definitions in a YANG module will become status obsolete in future revisions but definitions marked as such MUST still be implemented by compliant servers.
5. Requirements related to documentation and education:
 - 5.1 The solution MUST provide guidance to model authors and clients on how to use the new YANG versioning scheme.
 - 5.2 The solution is REQUIRED to describe how to transition from the existing YANG 1.0/1.1 versioning scheme to the new scheme.
 - 5.3 The solution MUST describe how the versioning scheme affects the interpretation of instance data and references to instance data, for which the schema definition has been updated in a non-backwards-compatible way.

6. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following people are members of that design team and have contributed to defining the problem and specifying the requirements:

- o Balazs Lengyel
- o Benoit Claise
- o Ebben Aries
- o Jason Sterne
- o Joe Clarke
- o Juergen Schoenwaelder
- o Mahesh Jethanandani
- o Michael (Wangzitao)
- o Qin Wu
- o Reshad Rahman
- o Rob Wilton

7. Acknowledgments

The design team would like to thank Christian Hopps and Vladimir Vassilev for their feedback and perspectives in shaping and fine tuning the versioning requirements.

One of the inspirations for solving the YANG module versioning comes from OpenConfig. The authors would like to thank Anees Shaikh and Rob Shakir for their helpful input.

8. Security Considerations

The document does not define any new protocol or data model. There is no security impact.

9. IANA Considerations

None

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>>.
- [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>>.

10.2. Informative References

- [RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <<https://www.rfc-editor.org/info/rfc8049>>.
- [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>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.

Author's Address

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

R. Wilton
R. Rahman
Cisco Systems, Inc.
March 11, 2019

YANG Schema Version Selection
draft-wilton-netmod-yang-ver-selection-00

Abstract

This document defines protocol mechanisms to allow clients to choose which YANG schema to use for interactions with a server, out of the available YANG schema supported by a server. The provided functionality allow servers to support clients in a backwards compatible way, at the same time allowing for non-backwards-compatible updates to YANG modules.

This draft provides a solution to YANG versioning requirements 3.1 and 3.2.

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 September 12, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology and Conventions	2
2. Introduction	3
3. Background	4
4. Objectives	4
5. Solution Overview	5
6. Version selection from a server perspective	6
7. Version selection from a clients perspective	7
8. Limitations of the solution	7
9. Schema Version Selection YANG module	8
10. YANG Module	9
11. Security Considerations	13
12. IANA Considerations	14
13. Open Questions/Issues	14
14. Acknowledgements	14
15. References	14
15.1. Normative References	14
15.2. Informative References	15
Authors' Addresses	16

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.verdt-netmod-yang-versioning-reqs].

This document also makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

- o datastore schema

In addition, this document makes use of the following terminology:

- o bc: Used as an abbreviation for a backwards-compatible change.
- o nbc: Used as an abbreviation for a non-backwards-compatible change.

- o editorial change: A backwards-compatible change that does not change the YANG module semantics in any way.
- o YANG schema: The combined set of schema nodes for a set of YANG module revisions, taking into consideration any deviations and enabled features.
- o versioned schema: A YANG schema with an associated YANG semantic version number, e.g., as might be described by a YANG package.
- o schema set: A set of related versioned YANG schema, one for each datastore that is supported.

TODO - the bc/nbc/editorial terminology should probably be defined and referenced from the YANG module versioning solution draft. 'schema' and 'versioned schema' could be defined in the packages draft.

2. Introduction

This document describes how NETCONF and RESTCONF clients can choose a particular YANG schema they wish to choose to interact with a server with.

[I-D.verdt-netmod-yang-versioning-reqs] defines requirements that any solution to YANG versioning must have.

[I-D.verdt-netmod-yang-semver] specifies a partial solution to the YANG versioning requirements that focuses on using semantic versioning within individual YANG modules, but does not address all the requirements listed in the requirements draft. Of particular relevance here, requirements 3.1 and 3.2 are not addressed.

[I-D.rwilton-netmod-yang-packages] describes how sets of related YANG modules can be grouped together into a logical entity that is versioned using the YANG semantic versioning number scheme. Different packages can be defined for different sets of YANG modules, e.g., packages could be defined for the IETF YANG modules, OpenConfig YANG modules, a vendor's YANG modules. Different versions of these package definitions can be defined as the contents of these packages evolve over time, and as the versions of the YANG modules included in the package evolve.

This draft defines how YANG packages can be used to represent versioned datastore schema, and how clients can choose which versioned schemas to use during interactions with a device.

3. Background

There are three ways that the lifecycle of a data model can be managed:

1. Disallow all non-backwards-compatible updates to a YANG module. Broadly this is the approach adopted by [RFC7950], but it has been shown to be too inflexible in some cases. E.g. it makes it hard to fix bugs in a clean fashion - it is not clear that allowing two independent data nodes (one deprecated, one current) to configure the same underlying property is robustly backwards compatible in all scenarios, particularly if the value space and/or default values differ between the module revisions.
2. Allow non-backwards-compatible updates to YANG modules, and use a mechanism such as semantic version numbers to communicate the likely impact of any changes to module users, but require that clients handle non-backwards-compatible changes in servers by migrating to new versions of the modules. Without version selection, this is what the [I-D.verdt-netmod-yang-semver] approach likely achieves.
3. Allow non-backwards-compatible updates to YANG modules, but also provide mechanisms to allow servers to support multiple versions of YANG modules, and provide clients with some ability to select which versions of YANG modules they wish to interact with, subject to some reasonable constraints. This is the approach that this draft aims to address. It is worth noting that the idea of supporting multiple versions of an API is not new in the wider software industry, and there are many examples of where this approach has been successfully used.

4. Objectives

The goals of the schema version selection draft are:

- o To provide a mechanism where non-backwards-compatible changes and bug fixes can be made to YANG modules without forcing clients to immediately migrate to new versions of those modules as they get implemented.
- o To allow servers to support multiple versions of a particular YANG schema, and to allow clients to choose which YANG schema version to use when interoperating with the server. The aim here is to give operators more flexibility as to when they update their software.

- o To provide a mechanism to allow different YANG schema families (e.g., SDO models, OpenConfig models, Vendor models) to be supported by a server, and to allow clients to choose which YANG schema family is used to interoperate with the server.

The following points are non objective of this draft:

- o This draft does not provide a mechanism to allow clients to choose arbitrary sets of YANG module versions to interoperate with the server.
- o Servers are not required to concurrently support clients using different YANG schema families or versioned schema. A server MAY choose to only allow a single schema family or single versioned schema to be used by all clients.
- o There is no requirement for a server to support every published version of a YANG package, particularly if some package versions are backwards compatible. Clients are required to interoperate with backwards compatible updates of YANG modules. E.g., if a particular package was available in versions 1.0.0, 1.1.0, 1.2.0, 2.0.0, 3.0.0 and 3.1.0, then a server may choose to only support versions 1.2.0, 2.0.0, and 3.1.0, with the knowledge that all clients should be able to interoperate with the server.
- o There is no requirement to support all parts of all versioned schemas. For some nbc changes in modules, it is not possible for a server to support both the old and new module versions, and to convert between the two. Where appropriate deviations can be used, and otherwise an out of band mechanism is used to indicate where a mapping has failed.

5. Solution Overview

An overview the solution is as follows:

1. YANG packages are defined for the different versioned schema supported by a server:
 - * Separate packages can be defined for different families of schema, e.g., SDO, OpenConfig, or vendor native.
 - * Separate packages can be defined for each versioned schema within a schema family.
 - * Separate packages may be defined for different datastores, if the datastores use different datastore schema. For example, a

different datastore schema, and hence package, might be used for <operational> vs the conventional datastores.

2. Each server advertises, via an operational data model:
 - * All of the YANG packages that may be used during version selection. The packages can also be made available for offline consumption via instance data documents, as described in [I-D.rwilton-netmod-yang-packages].
 - * Grouped sets of versioned schema, where each set defines the versioned schema used by each supported datastore, and each versioned schema is represented by a YANG package instance.
3. Each server supports configuration to:
 - * Allow a client to configure which schema version set to use for the default NETCONF/RESTCONF connections.
 - * Allow a client to configure additional separate NETCONF and RESTCONF protocol instances, which use different schema version sets on those protocol instances.
 - * An RPC mechanism could also be defined to select schema, but is not currently discussed in this draft.
4. The server internally maps requests between the different protocol instances to the internal device implementation.

6. Version selection from a server perspective

The general premise of this solution is that servers generally implement one native schema, and the version selection scheme is used to support older version of that native schema and also foreign schema specified by external entities.

Overall the solution relies on the ability to map instance data between different schema versions. Depending on the scope of difference between the schema versions then some of these mappings may be very hard, or even impossible, to implement. Hence, there is still a strong incentive to try and minimize nbc changes between schema versions to minimize the mapping complexity.

Server implementations MUST serialize configuration requests across the different schema. The expectation is that this would be achieved by mapping all requests to the devices native schema version.

Datastore validation needs to be performed in two places, firstly in whichever schema a clients is interacting in, and secondly in the native schema for the device. This could have a negative performance impact.

Depending on the complexity of the mappings between schema versions, it may be necessary for the mappings to be stateful.

TODO - Figure out how hot fixes that slightly modify the schema are handled.

7. Version selection from a clients perspective

Clients can use configuration to choose which schema sets are available.

Clients cannot choose arbitrary individual YANG module versions, and are instead constrained by the versions that the server makes available.

Each client protocol connection is to one particular schema set. From that client session perspective it appears as if the client is interacting with a regular server. If the client queries YANG library that the version of YANG Library that is returned matches the schema set that is being used for that server instance.

The server may not support a schema with the exact version desired by the client, and they have to accept a later version that is backwards compatible with their desired version. Clients may also have to accept later schema versions that contain NBC fixes, although the assumption is that such nbc fixes should be designed to minimize the impact on clients.

There is no guarantee that servers will always be able to support all older schema versions. Deviations should be used where necessary to indicate that the server is unable to faithfully implement the older schema version.

If clients interact with a server using multiple versions, they should not expect that all data nodes in later module versions can always be backported to older schema versions. TODO - Specify how mapping errors can be reported to client.

8. Limitations of the solution

Not all schema conversions are possible. E.g. an impossible type conversion, or something has been removed. The solution is fundamentally limited by how the schemas actually change, this

solution does not provide a magic bullet that can solve all versioning issues.

9. Schema Version Selection YANG module

The YANG schema version selection YANG module is used by a device to report the schema-sets that are available, and to allow clients to choose which schema-set they wish to use.

Feature are used to allow servers to decide whether they allow the primary schema-set to be changed, and/or allow secondary schema-sets to be configured.

The primary schema-set is the datastore schema reported by YANG Library if a client connects to the device using the standard NETCONF/RESTCONF protocol numbers.

If secondary schema-sets are configured, then the client can choose whether NETCONF or RESTCONF is supported, which port numbers the protocols should run on (if available), and what RESTCONF root path prefix to use (e.g. if all of the RESTCONF protocol instances run on port 443).

Different schema-sets may support different datastores.

The "ietf-schema-version-selection" YANG module has the following structure:

```
module: ietf-schema-version-selection
  +--rw schema-selection
    +--rw schema-sets* [name]
      +--rw name string
      +--rw netconf! {secondary-schema-set}?
      | +--rw port? inet:port-number
      +--rw restconf! {secondary-schema-set}?
      | +--rw port? inet:port-number
      | +--rw root-path? inet:uri
      +--ro datastores* [datastore]
      +--ro datastore ds:datastore-ref
      +--ro package
      | +--ro name?
      | | -> /yanglib:yang-library/pkg:package/name
      | +--ro version? leafref
    +--rw default-schema-set?
      -> /schema-selection/schema-sets/name
      {default-schema-set}?
```

10. YANG Module

The YANG module definition for the module described in the previous sections.

```
<CODE BEGINS> file "ietf-schema-version-selection@2019-03-11.yang"
module ietf-schema-version-selection {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-schema-version-selection";
  prefix "ver-sel";

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types.";
  }
  import ietf-datastores {
    prefix ds;
    reference
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }
  import ietf-yang-library {
    prefix yanglib;
    reference "RFC 8525: YANG Library";
  }
  import ietf-yang-library-packages {
    prefix pkg;
    reference "draft-rwilton-netmod-yang-packages-01";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

    Author:   Reshad Rahman
              <mailto:rrahman@cisco.com>

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

  description
    "This module provide a data model to advertise and allow the
    selection of schema versions by clients."
```

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

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

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

```
// 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 2019-03-11 {
  description
    "Initial revision";
  reference
    "RFC XXXX: YANG Schema Version Selection";
}

/*
 * Typedefs
 */

typedef yang-sem-ver {
  type string {
    pattern '\d+[\.]\d+[\.]\d+[mM]?';
  }
  description
    "Represents a YANG semantic version number.";
  reference
    "TODO - Should be defined by YANG versioning types module";
}

feature "default-schema-set" {
  description
    "Feature that allows clients to choose the default schema set
    to be used for clients that connect using the standard network
    configuration protocol port number or URL.

    Implementations may choose to only support this feature in
    <operational> to report the default-schema-set without
    allowing it to be configured.";
}
```

```
feature "secondary-schema-set" {
  description
    "Feature to choose if secondary schema sets may be configured
    by clients.

    Implementations may choose to only support this feature in
    <operational> to report secondary schema sets without
    allowing them to be configured.";
}

container schema-selection {
  description
    "YANG schema version selection";

  list schema-sets {
    key "name";

    description
      "All schema-sets that are available for client selection.";

    leaf name {
      type "string" {
        length "1..255";
      }
      description
        "The server assigned name of the schema-set.

        This should include the schema family, and appropriate
        versioning or release information";
    }
  }

  container netconf {
    if-feature "secondary-schema-set";

    presence "Make this schema-set available via NETCONF";
    description
      "NETCONF protocol settings for this schema set, if
      available";

    leaf port {
      type inet:port-number;
      description
        "The port numnber to use for interacting with this
        schema-set.  If not configured, then the port number is
        server allocated.";
      reference
        "RFC 6242: Using the NETCONF Protocol over SSH";
    }
  }
}
```

```
    }

    container restconf {
        if-feature "secondary-schema-set";

        presence
            "Make this schema-set available via RESTCONF";
        description
            "RESTCONF protocol settings for this schema set, if
            available";

        leaf port {
            type inet:port-number;
            default "443";
            description
                "The port numnber to use for interacting with this
                schema-set.  If not configured, then the port number
                defaults to the standard RESTCONF https port number of
                443";
            reference
                "RFC 8040: RESTCONF Protocol, section 2.1";
        }

        leaf root-path {
            type inet:uri;
            default "/restconf";
            description
                "The default root path to use to access the RESTCONF
                protocol instance for this schema-set";
        }
    }
}

list datastores {
    key "datastore";
    config false;

    description
        "The list of datastores supported for this schema set";

    leaf datastore {
        type ds:datastore-ref;
        description
            "The datastore that this datastore schema is associated
            with";
        reference
            "RFC 8342: Network Management Datastore Architecture
            (NMDA)";
    }
}
```



```
    }

    container package {
      description
        "YANG package associated with this datastore schema";

      leaf name {
        type leafref {
          path "/yanglib:yang-library/pkg:package/pkg:name";
        }
        description
          "The name of the YANG package this schema relates to";
      }
      leaf version {
        type leafref {
          path '/yanglib:yang-library/'
            + 'pkg:package[pkg:name = current()/../name]/'
            + 'pkg:version';
        }

        description
          "The version of the YANG package this schema relates
            to";
      }
    }
  }
}

leaf default-schema-set {
  if-feature "default-schema-set";
  type leafref {
    path '/schema-selection/schema-sets/name';
  }
  description
    "Specifies the default schema-set used by this device. This
      is the set of datastore schema that is used if a client
      connects using the standard protocol port numbers and URLs";
}
}
}
<CODE ENDS>
```

11. Security Considerations

To be defined.

12. IANA Considerations

TODO - Add registrations for YANG modules defined in this draft.

13. Open Questions/Issues

All issues, along with the draft text, are currently being tracked at: TODO - URL

14. Acknowledgements

The ideas that formed this draft are based on discussions with the YANG versioning design team, and other members of the NETMOD WG.

15. References

15.1. Normative References

- [I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.
- [I-D.ietf-netmod-module-tags]
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", draft-ietf-netmod-module-tags-07 (work in progress), March 2019.
- [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-02 (work in progress), February 2019.
- [I-D.rwilton-netmod-yang-packages]
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-00 (work in progress), December 2018.
- [I-D.verdt-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning for Modules", draft-verdt-netmod-yang-semver-00 (work in progress), March 2019.
- [I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 2018.

- [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>>.
- [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>>.
- [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>>.

15.2. Informative References

- [I-D.bierman-netmod-yang-package]
Bierman, A., "The YANG Package Statement", draft-bierman-netmod-yang-package-00 (work in progress), July 2015.

- [I-D.ietf-netmod-artwork-folding]
Watsen, K., Wu, Q., Farrel, A., and B. Claise, "Handling
Long Lines in Inclusions in Internet-Drafts and RFCs",
draft-ietf-netmod-artwork-folding-01 (work in progress),
March 2019.
- [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>>.

Authors' Addresses

Robert Wilton
Cisco Systems, Inc.

Email: rwilton@cisco.com

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 2, 2019

Q. Wu
Huawei
B. Lengyel
Ericsson Hungary
Y. Niu
Huawei
November 29, 2018

Factory default Setting
draft-wu-netmod-factory-default-02

Abstract

This document defines a method to reset a YANG datastore to its factory-default content. The reset operation may be used e.g. during initial zero-touch configuration or when the existing configuration has major errors, so re-starting the configuration process from scratch is the best option.

A new reset-datastore RPC is defined. Several methods of documenting the factory-default content are specified.

Optionally a new "factory-default-running" read-only datastore is defined, that contains the data that will be copied over to the running datastore at reset.

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 June 2, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
2. Reset-Datastore RPC	4
3. Factory-Default Datastore	4
4. YANG Module	5
5. IANA Considerations	7
6. Security Considerations	8
7. Acknowledgements	8
8. Contributors	8
9. References	8
9.1. Normative References	8
9.2. Informative References	9
Appendix A. Open Issues	9
Appendix B. Difference between <startup> datastore and <factory- default> datastore	9
Appendix C. Changes between revisions	9
Authors' Addresses	10

1. Introduction

This document defines a method to reset a YANG datastore to its factory-default content. The reset operation may be used e.g. during initial zero-touch configuration or when the existing configuration has major errors, so re-starting the configuration process from scratch is the best option. When resetting a datastore all previous configuration settings will be lost and replaced by the factory-default content.

A new reset-datastore RPC is defined. Several methods of documenting the factory-default content are specified.

Optionally a new "factory-default-running" read-only datastore is defined, that contains the data that will be copied over to the running datastore at reset. This datastore can also be used in <get-data> or <copy-config> operations.

NETCONF defines the <delete> operation that allows resetting the <startup> datastore and the <discard-changes> operation that copies the content of the <running> datastore into the <candidate> datastore. However it is not possible to reset the running datastore, to reset the candidate datastore without changing the running datastore or to reset any dynamic datastore.

A RESTCONF server MAY implement the above NETCONF operations, but that would still not allow it to reset the running configuration.

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 [RFC8342] and are not redefined here:

- o startup configuration datastore
- o candidate configuration datastore
- o running configuration datastore
- o intended configuration datastore
- o operational state datastore

The following terms are defined in this document as follows:

- o factory-default datastore: A read-only datastore holding a preconfigured minimal initial configuration that can be used to initialize the configuration of a server. The content of the datastore is usually static, but MAY depend on external factors like available HW.

2. Reset-Datastore RPC

A new "reset-datastore" RPC is introduced. It will have a target datastore as a parameter. Upon receiving the RPC the YANG server resets the content of the target datastore to its factory-default content. Only writable datastores can be specified as a target. Read-only datastores receive their content from other datastores (e.g. <intended> gets its content from <running>).

Factory-default content SHALL be specified by one of the following means in order of precedence

1. For the <running>, <candidate> and <startup> datastores as the content of the <factory-default> datastore, if it exists
2. YANG Instance Data [I-D.ietf-netmod-yang-instance-file-format]
3. In some implementation specific manner
4. For dynamic datastores unless otherwise specified the factory-default content is empty.

3. Factory-Default Datastore

This document introduces a new datastore resource named 'Factory-Default' that represents a preconfigured minimal initial configuration that can be used to initialize the configuration of a server.

- o Name: "factory-default"
- o YANG modules: all
- o YANG nodes: all "config true" data nodes
- o Management operations: The content of the datastore is set by the YANG server in an implementation dependent manner. The content can not be changed by management operations via NETCONF, RESTCONF, the CLI etc. unless specialized, dedicated operations are provided. The contents of the datastore can be read using NETCONF, RESTCONF <get-data> operation. The operations <reset-datastore> or <copy-config> can be used to copy the content of the datastore to another datastore. The content of the datastore is not propagated automatically to any other datastores.
- o Origin: This document does not define a new origin identity as it does not interact with <operational> datastore.

- o Protocols: RESTCONF, NETCONF and other management protocol.
- o Defining YANG module: "ietf-factory-default"

The datastore content is usually defined by the device vendor. It is usually static, but MAY change e.g. depending on external factors like HW available or during device upgrade.

On devices that support non-volatile storage, the contents of <factory > MUST persist across restarts

4. YANG Module

```
<CODE BEGINS> file "ietf-factory-default.yang"
module ietf-factory-default {
  yang-version 1.1;
  namespace urn:ietf:params:xml:ns:yang:ietf-factory-default;
  prefix fdef;

  import ietf-netconf { prefix nc ; }
  import ietf-datastores { prefix ds; }

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

    WG Chair: Lou Berger
                <mailto:lberger@labn.net>
    WG Chair: Joel Jaeggli
                <mailto:joelja@bogus.com>
    WG Chair: Kent Watsen
                <mailto:kwatsen@juniper.net>

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

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

  description
    "This module defines the
    - reset-datastore RPC
    - factory-default datastore
    - an extension to the Netconf <copy-config> operation to
    allow it to operate on the factory-default datastore."
```

It provides functionality to reset a YANG datastore to its factory-default content.

Copyright (c) 2018 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>).

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in the module text are to be interpreted as described in RFC 2119 (<https://tools.ietf.org/html/rfc2119>).

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

```
revision 2018-10-09 {
  description
    "Initial revision.";
  reference "RFC XXXX: Factory default Setting Capability for
    RESTCONF";
}

feature factory-default-as-datastore {
  description "Indicates that the factory default configuration is
    also available as a separate datastore";
}

rpc reset-datastore {
  description "The target datastore is reset to its factory
    default content. ";

  input {
    leaf-list target-datasore {
      type identityref {
        base "ds:datastore" ;
      }
      min-elements 1;
      description "The datastore(s) whose content will be
        replaced by the factory-default configuration.";
    }
    // Do we need an extra parameter that may order a restart of
```

```
    // the YANG-server or the whole system?
  }
}

identity factory-default {
  if-feature factory-default-as-datastore;
  base ds:datastore;
  description "The read-only datastore contains the configuration that
    will be copied into e.g. the running datastore by the
    reset-datastore operation if the target is the running
    datastore.";
}

augment /nc:copy-config/nc:input/nc:source/nc:config-source {
  if-feature factory-default-as-datastore;
  description " Allows the copy-config operation to use the
    factory-default datastore as a source";
  leaf factory-default {
    type empty ;
    description
      "The factory-default datastore is the source.";  }
}
}
<CODE ENDS>
```

5. IANA Considerations

This document registers one URI in the IETF XML Registry [RFC3688]. The following registration has been made:

URI: urn:ietf:params:xml:ns:yang:ietf-factory-default

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers one YANG module in the YANG Module Names Registry [RFC6020]. The following registration has been made:

name: ietf-factory-default

namespace: urn:ietf:params:xml:ns:yang:ietf-factory-default

prefix: fdef

RFC: xxxx

6. Security Considerations

The <reset-datastore> RPC can overwrite important and security sensitive information in one of the other datastores e.g. running, therefore it is important to restrict access to this RPC using the standard access control methods. [RFC8341]

The content of the factory-default datastore is usually not security sensitive as it is the same on any device of a certain type.

7. Acknowledgements

Thanks to Juergen Schoenwaelder, Ladislav Lhotka to review this draft and provide important input to this document.

8. Contributors

Rohit R Ranade
Huawei
Email: rohitrranade@huawei.com

9. References

9.1. 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>>.
- [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>>.
- [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>>.

9.2. Informative References

[I-D.ietf-netconf-zerotouch]

Watsen, K., Abrahamsson, M., and I. Farrer, "Zero Touch Provisioning for Networking Devices", draft-ietf-netconf-zerotouch-25 (work in progress), September 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-00 (work in progress), November 2018.

Appendix A. Open Issues

- o Do we need a restart after <reset-datastore> ? What kind of restart, just the YANG-Server or the full system?
- o Do we need the concept of reboot? How is that different from a restart? Does it result in some sort of reset-datastore?

Appendix B. Difference between <startup> datastore and <factory-default> datastore

When the device first boots up, the content of the <startup> and <factory-default> will be identical. The content of <startup> can be subsequently changed by using <startup> as a target in a <copy-config> operation. The <factory-default> is a read-only datastore and it is usually static as described in earlier sections.

Appendix C. Changes between revisions

v01 - v02

- o Add copy-config based on Rob's comment.
- o Reference Update.

v3 - v00 - v01

- o Changed name from draft-wu-netconf-restconf-factory-restore to draft-wu-netmod-factory-default
- o Removed copy-config ; reset-datastore is enough

v02 - v03

- o Restructured

- o Made new datastore optional
- o Removed Netconf capability
- o Listed Open issues

v01 - v02

- o -

v00 - v01

- o -

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Balazs Lengyel
Ericsson Hungary
Magyar Tudosok korutja 11
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Ye Niu
Huawei

Email: niuye@huawei.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 2, 2019

M. Wang
Q. Wu
Huawei
C. Xie
China Telecom
March 1, 2019

A YANG Data model for Policy based Event Management
draft-wwx-netmod-event-yang-01

Abstract

[RFC8328] defines a policy-based management framework that allow definition of a data model to be used to represent high-level, possibly network-wide policies. This document defines an YANG data model for the policy based event management [RFC7950]. The policy based Event YANG provides the ability for the network management function (within a controller, an orchestrator, or a network element) to control the configuration and monitor state change on the network element and take simple and instant action when a trigger condition on the system state is met.

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 September 2, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions used in this document	2
2.1. Terminology	2
2.2. Tree Diagrams	3
3. Objectives	3
4. Relationship to YANG Push	4
5. Relationship to EVENT MIB	5
6. Model Overview	6
7. EVENT TRIGGER YANG Module	10
8. EVENT YANG Module	14
9. Security Considerations	19
10. IANA Considerations	20
11. Normative References	20
Appendix A. Example of Event	21
Appendix B. Changes between revisions	23
Authors' Addresses	23

1. Introduction

[RFC8328] defines a policy-based management framework that allow definition of a data model to be used to represent high-level, possibly network-wide policies. This document defines an policy based Event Management YANG data model [RFC7950]. The policy based Event management YANG provides the ability for the network management function (within a controller, an orchestrator, or a network element) to monitor state changes on the network element and take simple and instant action when a trigger condition on the system state is met.

The data model in this document is designed to be compliant with the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Conventions used in this document

2.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when

in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

This document uses the following terms:

Error A deviation of a system from normal operation [RFC3877].

Fault Lasting error or warning condition [RFC3877].

Event Something that happens which may be of interest or trigger the invocation of the rule. A fault, an alarm, a change in network state, network security threat, hardware malfunction, buffer utilization crossing a threshold, network connection setup, an external input to the system, for example [RFC3877].

2.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

3. Objectives

This section describes some of the design objectives for the policy based Event management Data Model:

- o The policy based Event management YANG should provide the ability for the network management function to control configuration and monitor state changes on a network element using the NETCONF/RESTCONF, and initiate simple actions whenever a trigger condition is met. For example, a NETCONF subscribed notification can be generated when a system state value exceeds the threshold.
- o Clear and precise identification of policy based Event types and managed objects.
- o Allow the server to inform the client that certain Event are related to other Event.
- o Allow one event to be able to trigger another external event or generate derived events.
- o The event data model defined in this document can be implemented on the management system that also implements EVENT-MIB; thus, the mapping between the event data model and ENTITY-MIB should be clear.

4. Relationship to YANG Push

YANG-push mechanism provides a subscription service for updates from a datastore. And it supports two types of subscriptions which are distinguished by how updates are triggered: periodic and on-change.

The On-change Push allow receivers to receive updates whenever changes to target managed objects occur. This document specifies a mechanism that provides three trigger conditions:

- o Existence: When a specific managed object appears, the trigger fires, e.g. reserved ports are configured.
- o Boolean: The user can set the type of boolean comparison (e.g. unequal, equal, less, less-or-equal, greater, greater-or-equal, etc). If the test result is true, the trigger fires. The trigger will not fire again until the test result has become false and fall back to be true. e.g., when the boolean comparison type is 'less', the trigger will be fired if the system state value of managed object is less than the value set for the target managed object.
- o Threshold: The event that may be triggered when a managed object in multiple instances of the data tree is found and exceed the pre-configured threshold. For example, when the system state value of that managed object is greater than or equal to 'rising-value' and the 'startup' is equal to 'rising', then one threshold rising event is triggered for that managed object. Another example is if the system state value of the managed object is less than or equal to 'falling-value' and the 'startup' is equal to 'falling' then one threshold falling event is triggered for that managed object.

And the YANG Push mechanism more focus on the remote mirroring and monitoring of configuration and operational state. For example, for on change method, the subscriber will receive a notification if the changes occurs. The model defined in this document provides a method which allow automatic adjusting the value of the corresponding managed object when some event is triggered. It establishes connection between network service monitoring and network service provision and can use output generated by network service monitoring as input of network service provision and thereby provide automated network management. The details of the usage example is described in Appendix A.

5. Relationship to EVENT MIB

If the device implements the EVENT-MIB [RFC2981], each entry in the `"/events/event/trigger"` list is mapped to `MteTriggerEntry`, `MteTriggerExistenceEntry`, `MteTriggerBooleanEntry`, `MteTriggerThresholdEntry`, `MteObjectsEntry`, `MteEventEntry`, `MteEventSetEntry`. respectively.

The following table lists the YANG data nodes with corresponding objects in the EVENT-MIB [RFC2981].

YANG data node in ietf-event.yang	EVENT-MIB Objects (RFC2981)
min-data-object	mteResourceSampleMinimum
max-data-object	mteResourceSampleInstanceMaximum
traget	mteObjectsName
event-name	mteEventName
event-description	mteEventComment
value	mteEventSetValue
events/event/trigger/name	mteTriggerName
trigger-description	mteTriggerComment
frequency	mteTriggerFrequency
comparison	mteTriggerBooleanComparison
value	mteTriggerBooleanValue
rising-event	mteTriggerThresholdRising
falling-event	mteTriggerThresholdFalling
delta-rising-event	mteTriggerThresholdDeltaRising
threshold/startup	mteTriggerThresholdStartup
existence/enable	mteTriggerExistenceStartup
boolean/enable	mteTriggerBooleanStartup

6. Model Overview

The YANG data model for the Event management has been split into two modules:

- o The `ietf-event-trigger.yang` module defines a grouping for a generic trigger. It is intended that this grouping will be used by the policy based event management model or other models that require the trigger conditions. In this model, three trigger conditions are defined under the "test" choice node:
 - * **Existence:** When a specific managed object appears, the trigger fires.
 - * **Boolean:** If the test result on specific managed object is true the trigger fires. The Boolean trigger condition is used when threshold value is a static value or pre-configured value.
 - * **Threshold:** The event that may be triggered when the system state value of a managed object within data-instance list exceed the threshold. The threshold trigger condition is used when threshold value changes over time or the system state value of managed object changes in ascend or descend order.
- o The `ietf-event.yang` module defines four lists: trigger, target, event, and action. Triggers define the targets meeting some conditions that lead to events. Events trigger corresponding actions:
 - * Each trigger can be seen as a logical test that, if satisfied or evaluated to be true, cause the action to be carried out. The `ietf-event.yang` module uses groupings defined in `ietf-event-trigger.yang` to present the trigger attributes.
 - * The target list defines managed objects that can be added to notifications or be set to a new value on the trigger, the trigger test type, or the event that resulted in the actions.
 - * The event list defines what happens when an event is triggered, i.e., trigger corresponding action, e.g., sending a notification, setting a value to the managed object or both.
 - * The action list consists of updates or invocations on local managed object attributes and defines a set of actions which will be performed (e.g. notification, set, another event, etc) when corresponding event be triggered. The value to be set can use many variations on rule structure.

The following tree diagrams [RFC8340] provide an overview of the data model for "ietf-event-trigger" module and the "ietf-event" module.

groupings:
trigger-grouping

```

+---- (test)?
+---:(existences)
|   +---- existences
|       +---- target*           target
+---:(boolean)
|   +---- boolean
|       +---- comparison?      enumeration
|       +---- value?           match-value
|       +---- target*          target
+---:(threshold)
+---- threshold
|   +---- rising-value?         match-value
|   +---- rising-target*        target
|   +---- falling-value?       match-value
|   +---- falling-target*       target
|   +---- delta-rising-value?   match-value
|   +---- delta-rising-target*  target
|   +---- delta-falling-value?  match-value
|   +---- delta-falling-target* target
|   +---- startup?              enumeration

```

```

module: ietf-event

```

```

+--rw events
+--rw min-data-object?      uint32
+--rw max-data-object?      uint32
+--rw event* [event-name type]
|   +--rw event-name         string
|   +--rw type                identityref
|   +--rw event-description?  string
|   +--rw target*             trig:target
|   +--rw clear?              boolean
|   +--rw related-event* [event-name type]
|   |   +--rw event-name     string
|   |   +--rw type           identityref
+--rw trigger* [name]
|   +--rw name                string
|   +--rw type?               enumeration
|   +--rw trigger-description? string
|   +--rw frequency
|   |   +--rw type?           identityref
|   |   +--rw periodic
|   |   |   +--rw interval    uint32
|   |   |   +--rw start?      yang:date-and-time
|   |   |   +--rw end?        yang:date-and-time
|   |   +--rw scheduling
|   |   |   +--rw month*       string
|   |   |   +--rw day-of-month* uint8

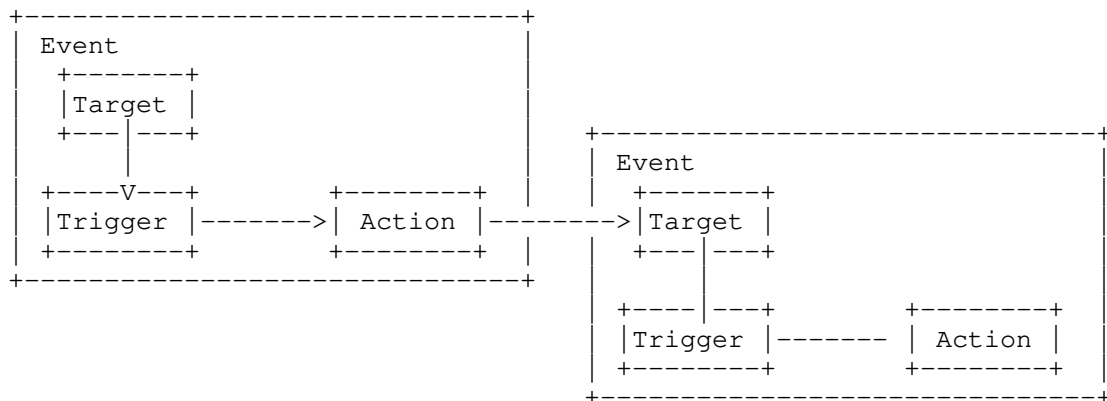
```

```

|         +---rw day-of-week*      uint8
|         +---rw hour*             uint8
|         +---rw minute*          uint8
|         +---rw second*          uint8
|         +---rw start?           yang:date-and-time
|         +---rw end?             yang:date-and-time
+---rw (test)?
|   +---:(existences)
|   |   +---rw existences
|   |   |   +---rw target*      -> /events/event/target
|   +---:(boolean)
|   |   +---rw boolean
|   |   |   +---rw comparison?  enumeration
|   |   |   +---rw value?       match-value
|   |   |   +---rw target*      target
|   +---:(threshold)
|   |   +---rw threshold
|   |   |   +---rw rising-value? match-value
|   |   |   +---rw rising-target* target
|   |   |   +---rw falling-value? match-value
|   |   |   +---rw falling-target* target
|   |   |   +---rw delta-rising-value? match-value
|   |   |   +---rw delta-rising-target* target
|   |   |   +---rw delta-falling-value? match-value
|   |   |   +---rw delta-falling-target* target
|   |   |   +---rw startup?     enumeration
+---rw action* [action-name]
|   +---rw action-name            string
+---n event-notification
|   +---- event-name?      -> /events/event/event-name
|   +---- type?           -> /events/event/type
|   +---- target*         trig:target
+---x set
|   +---w input
|   |   +----w target*     trig:target
|   |   +----w value?     <anydata>
+---rw trigger-event*      -> ../../event-name

```

The relation between Event, Trigger, Target and Action is described as follows:



One event may trigger another event, i.e., the action output in the first event can be input to target in the second event, but if it does not trigger another event, the relation between action and target should be ignored.

7. EVENT TRIGGER YANG Module

```

<CODE BEGINS> file "ietf-event-trigger@2018-12-18.yang"
module ietf-event-trigger {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-event-trigger";
  prefix trig;

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF xxx Working Group";
  contact
    "Zitao Wang: wangzitao@huawei.com
     Qin Wu: bill.wu@huawei.com";
  description
    "This module defines a reusable grouping for event trigger.";

  revision 2018-12-18 {
    description
      "Initial revision.";
    reference "foo";
  }

  typedef match-value {
    type union {

```



```
    type yang:xpath1.0;
    type yang:object-identifier;
    type string;
}
description
  "This type is used to match resources of type 'target'.
  Since the type 'target' is a union of different types,
  the 'match-value' type is also a union of corresponding
  types."
}

typedef target {
  type union {
    type instance-identifier;
    type yang:object-identifier;
    type yang:uuid;
    type string;
  }
  description
    "If the target is modelled in YANG, this type will
    be an instance-identifier.
    If the target is an SNMP object, the type will be an
    object-identifier.
    If the target is anything else, for example a distinguished
    name or a CIM path, this type will be a string.
    If the target is identified by a UUID use the uuid
    type.
    If the server supports several models, the presedence should
    be in the order as given in the union definition."
}

grouping trigger-grouping {
  description
    "A grouping that provides event trigger.";
  choice test {
    description
      "Choice test";
    container existences {
      leaf-list target {
        type target;
        description
          "List for target objects";
      }
      description
        "Container for existence";
    }
    container boolean {
      leaf comparison {
```

```
type enumeration {
  enum "unequal" {
    description
      "Indicates that the comparision type is unequal to.";
  }
  enum "equal" {
    description
      "Indicates that the comparision type is equal to.";
  }
  enum "less" {
    description
      "Indicates that the comparision type is less than.";
  }
  enum "less-or-equal" {
    description
      "Indicates that the comparision type is less than
      or equal to.";
  }
  enum "greater" {
    description
      "Indicates that the comparision type is greater than.";
  }
  enum "greater-or-equal" {
    description
      "Indicates that the comparision type is greater than
      or equal to.";
  }
}
description
  "Comparison type.";
}
leaf value {
  type match-value;
  description
    "Compation value which is static threshold value.";
}
leaf target {
  type target;
  description
    "List for target management objects.";
}
description
  "Container for boolean test.";
}
container threshold {
  leaf rising-value {
    type match-value;
    description
```

```
        "Sets the rising threshold to the specified value,
        when the current sampled value is greater than or equal to
        this threshold, and the value at the last sampling interval
        was less than this threshold, the event is triggered. ";
    }
    leaf-list rising-target {
        type target;
        description
            "List for target objects.";
    }
    leaf falling-value {
        type match-value;
        description
            "Sets the falling threshold to the specified value.";
    }
    leaf-list falling-target {
        type target;
        description
            "List for target objects.";
    }
    leaf delta-rising-value {
        type match-value;
        description
            "Sets the delta rising threshold to the specified value.";
    }
    leaf-list delta-rising-target {
        type target;
        description
            "List for target objects.";
    }
    leaf delta-falling-value {
        type match-value;
        description
            "Sets the delta falling threshold to the specified value.";
    }
    leaf-list delta-falling-target {
        type target;
        description
            "List for target objects.";
    }
    leaf startup {
        type enumeration {
            enum "rising" {
                description
                    "If the first sample after this
                    managed object becomes active is greater than or equal
                    to 'rising-value' and the 'startup' is equal to
                    'rising' then one threshold rising event is
```

```

        triggered for that managed object.";
    }
    enum "falling" {
        description
            "If the first sample after this managed object becomes
            active is less than or equal to 'falling-value' and
            the 'startup' is equal to 'falling' then one
            threshold falling event is triggered for that managed
            object.";
    }
    enum "rising-or-falling" {
        description
            "That event may be triggered when the
            'startup' is equal to 'rising-or-falling'.
            'rising-or-falling' indicate the state value of the
            managed object may less than or greater than the
            specified threshold value.";
    }
    }
    description
        "Startup setting.";
    }
    description
        "Container for the threshold trigger condition.
        Note that the threshold here may change over time
        or the state value changes in either ascend order
        or descend order.";
    }
    }
    }
}

```

<CODE ENDS>

8. EVENT YANG Module

```

<CODE BEGINS> file "ietf-event@2018-09-18.yang"

module ietf-event {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-event";
    prefix evt;

    import ietf-yang-types {
        prefix yang;
    }

    import ietf-event-trigger {

```

```
    prefix trig;
}

organization
  "IETF xxx Working Group";
contact
  "Zitao Wang: wangzitao@huawei.com
  Qin Wu: bill.wu@huawei.com";
description
  "This module defines a model for the service topology.";

revision 2018-12-18 {
  description
    "Initial revision.";
  reference "foo";
}

identity event-type {
  description
    "Base identity for event type";
}

identity frequency {
  description
    "Base identity for frequency";
}

identity periodic {
  base frequency;
  description
    "Identity for periodic trigger";
}

identity scheduling {
  base frequency;
  description
    "Identity for scheduling trigger";
}

typedef match-value {
  type union {
    type yang:xpath1.0;
    type yang:object-identifier;
    type string;
  }
  description
    "This type is used to match resources of type 'target'.
    Since the type 'target' is a union of different types,
```

```
        the 'match-value' type is also a union of corresponding
        types.";
    }

typedef target {
    type union {
        type instance-identifier;
        type yang:object-identifier;
        type yang:uuid;
        type string;
    }
    description
        "If the target is modelled in YANG, this type will
        be an instance-identifier.
        If the target is an SNMP object, the type will be an
        object-identifier.
        If the target is anything else, for example a distinguished
        name or a CIM path, this type will be a string.
        If the target is identified by a UUID use the uuid
        type.
        If the server supports several models, the presedence should
        be in the order as given in the union definition.";
}

grouping start-end-grouping {
    description
        "A grouping that provides start and end times for
        Event objects.";
    leaf start {
        type yang:date-and-time;
        description
            "The date and time when the Event object
            starts to create triggers.";
    }
    leaf end {
        type yang:date-and-time;
        description
            "The date and time when the Event object
            stops to create triggers.
            It is generally a good idea to always configure
            an end time and to refresh the end time as needed
            to ensure that agents that lose connectivity to
            their Controller do not continue executing Schedules
            forever.";
    }
}

container events {
```

```
leaf min-data-object {
    type uint32;
    description
        "Sets the minimum number for a set of data collected
        or selected during the service monitoring.";
}
leaf max-data-object {
    type uint32;
    description
        "Sets the maximum number for a set of data collected or selected
        during the service monitoring.";
}
list event {
    key "event-name type";
    leaf event-name {
        type string;
        description
            "Event name";
    }
    leaf type {
        type identityref {
            base event-type;
        }
        description
            "Type of event";
    }
    leaf event-description {
        type string;
        description
            "Event description";
    }
    leaf-list target {
        type target;
        description
            "targeted objects";
    }
    leaf clear {
        type boolean;
        default "false";
        description
            "A flag indicate whether the event be closed";
    }
    list related-event {
        key "event-name type";
        leaf event-name {
            type string;
            description
                "Event name";
        }
    }
}
```

```
    }
    leaf type {
        type identityref {
            base event-type;
        }
        description
            "Type of event";
    }
    description
        "List for related events";
}

uses trig:trigger-grouping;

list action {
    key "action-name";
    leaf action-name {
        type string;
        description
            "Action name";
    }
    notification event-notification {
        leaf event-name {
            type leafref {
                path "/events/event/event-name";
            }
            description
                "Report the event name";
        }
        leaf type {
            type leafref {
                path "/events/event/type";
            }
            description
                "Report the event type";
        }
        leaf-list target {
            type target;
            description
                "Report the target objects";
        }
        description
            "This notification is used to report that an operator
            acted upon an Event.";
    }
    action set {
        input {
            leaf-list target {
```



```
        type target;
        description
            "Report the target objects";
    }
    anydata value {
        description
            "Inline set content.";
    }
}
leaf-list trigger-event {
    type leafref {
        path "../../event-name";
    }
    description
        "This action trigger another event";
}
description
    "List for Actions";
}
description
    "List for Events";
}
description
    "YANG data module for defining event triggers and
    actions for network management purposes";
}
```

<CODE ENDS>

9. Security Considerations

The YANG modules defined in this document MAY be accessed via the RESTCONF protocol [RFC8040] or NETCONF protocol ([RFC6241]). The lowest RESTCONF or NETCONF layer requires that the transport-layer protocol provides both data integrity and confidentiality, see Section 2 in [RFC8040] and [RFC6241]. 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.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- o /events/event/event-name
- o /events/event/target
- o /events/action/set/target
- o /events/event/trigger/name

10. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

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

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
-----
Name:          ietf-event
Namespace:     urn:ietf:params:xml:ns:yang:ietf-event
Prefix:        evt
Reference:     RFC xxxx
-----
```

11. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.
- [RFC2981] Kavasseri, R., Ed., "Event MIB", RFC 2981, DOI 10.17487/RFC2981, October 2000, <<https://www.rfc-editor.org/info/rfc2981>>.

- [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>>.
- [RFC6370] Bocci, M., Swallow, G., and E. Gray, "MPLS Transport Profile (MPLS-TP) Identifiers", RFC 6370, DOI 10.17487/RFC6370, September 2011, <<https://www.rfc-editor.org/info/rfc6370>>.
- [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>>.
- [RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <<https://www.rfc-editor.org/info/rfc7952>>.
- [RFC8328] Liu, W., Xie, C., Strassner, J., Karagiannis, G., Klyus, M., Bi, J., Cheng, Y., and D. Zhang, "Policy-Based Management Framework for the Simplified Use of Policy Abstractions (SUPA)", RFC 8328, DOI 10.17487/RFC8328, March 2018, <<https://www.rfc-editor.org/info/rfc8328>>.

Appendix A. Example of Event

For example, some service requires to monitoring the "in-errors" state of the interface, and if the value of "in-errors" exceeds the threshold, the event should reset the interface's enabled value to false:

```
<events>
  <event>
    <event-name>interface-state-exception</event-name>
    <type>interface-exception</type>
    <target>/if:interfaces/if:interface[if:name='eth1']</target>
    <target>/if:interfaces/if:interface[if:name='eth2']</target>
    <target>/if:interfaces/if:interface[if:name='eth3']</target>
    <trigger>
      <name>evaluate-in-errors</name>
      <trigger-description>evaluate the number of
        the packets that contained errors
      </trigger-description>
      <frequency>10m</frequency>
      <type>threshold</type>
      <test>
        <threshold>
          <startup>rising</startup>
          <rising-value>100</rising-value>
          <rising-target>/if:interfaces/if:interface[if:name='eth1']
            /if:statistic/if:in-errors</rising-target>
          <rising-target>/if:interfaces/if:interface[if:name='eth2']
            /if:statistic/if:in-errors</rising-target>
        </threshold>
      </test>
    </trigger>
    <action>
      <name>interface-exception</name>
      <event-notification>
        <event-name>interface-state-exception</event-name>
        <type>interface-exception</type>
        <target>/if:interfaces/if:interface[if:name='eth1']</target>
      </event-notification>
      <set>
        <target>/if:interfaces/if:interface[if:name='eth1']</target>
        <interger-value>
          <interfaces>
            <interface>
              <name>eth1</name>
              <enable>false</enable>
            </interface>
          </interfaces>
        </interger-value>
      </set>
    </action>
  </event>
</events>
```

Appendix B. Changes between revisions

v00 - v01

- o Separate ietf-event-trigger.yang from Event management model and ietf-event.yang and make it reusable in other YANG models.
- o Clarify the difference between boolean trigger condition and threshold trigger condition.
- o Change evt-smp-min and evt-smp-max into min-data-object and max-data-object in the data model.

Authors' Addresses

Michael Wang
Huawei Technologies, Co., Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Chongfeng Xie
China Telecom

Email: xiechf@ctbri.com.cn