                     YANG Data Structure Extensions

Abstract

   This document describes YANG mechanisms for defining abstract data
   structures with YANG.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc8791.

Table of Contents

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.

   This document defines a new YANG extension statement called
   "structure", which is similar to but more flexible than the "yang-
   data" extension from [RFC8040]. There is no assumption that a YANG
   data structure can only be used as a top-level abstraction, and it
   may also be 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 and is 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 term is used within this document:

   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:

*   configuration

*   operational state

1.1.2. YANG

The following terms are defined in [RFC7950] and are not redefined
here:

*   absolute-schema-nodeid

*   container

*   data definition statement

*   data node

*   leaf

*   leaf-list

*   list

2. Definitions

A YANG data structure is defined with the "structure" extension
statement, which is 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, the seventh bullet states:

> 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, once defined, has to be
used as a top-level protocol message or other top-level data
structure.

A YANG data structure is encoded in the same way as an "anydata"
node. This means that the name of the structure is encoded as a
"container", with the instantiated children encoded as child nodes to
this node. For example, this structure:

```
module example-errors {
  ...
```

```
   sx:structure my-error {
     leaf error-number {
       type int;
     }
   }
 }
```

can be encoded in JSON as:

```
"example-errors:my-error": {
  "error-number": 131
}
```

3. YANG Data Structures in YANG Tree Diagrams

A YANG data structure can be printed in a YANG tree diagram
[RFC8340]. This document updates RFC 8340 [RFC8340] by defining two
new sections in the tree diagram for a module:

1. YANG data structures, which are offset by two spaces and
    identified by the keyword "structure" followed by the name of the
    YANG data structure and a colon (":") character.

2. YANG data structure augmentations, which are offset by 2 spaces
    and identified by the keyword "augment-structure" followed by the
    augment target structure name and a colon (":") character.

The new sections, including spaces conventions, appear as follows:

```
  structure <structure-name>:
    +--<node>
       +--<node>
       |  +--<node>
       +--<node>
  structure <structure-name>:
    +--<node>

  augment-structure <structure-name>:
    +--<node>
       +--<node>
       |  +--<node>
       +--<node>
  augment-structure <structure-name>:
    +--<node>
```

Nodes in YANG data structures are printed according to the rules
defined in Section 2.6 of [RFC8340]. The nodes in YANG data
structures do not have any <flags>.

4. YANG Data Structure Extensions Module

```
<CODE BEGINS> file "ietf-yang-structure-ext@2020-06-17.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:   <https://datatracker.ietf.org/wg/netmod/>
   WG List:  <mailto:netmod@ietf.org>

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

   Author:   Martin Bjorklund
             <mailto:mbj+ietf@4668.se>

   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) 2020 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 8791
   (https://www.rfc-editor.org/info/rfc8791); see the RFC itself
   for full legal notices.";

revision 2020-06-17 {
  description
    "Initial revision.";
  reference
    "RFC 8791: YANG Data 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 substatement to 'module' or 'submodule'.

The substatements 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' node. 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 are 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 substatements 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 substatements 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 a 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 and is
         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 substatement to 'module' or 'submodule'.

         The substatements 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 are 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 substatements
         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 substatements 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>
```

5. IANA Considerations

5.1. YANG Module Registry

IANA has registered the following URI in the "ns" subregistry within
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.

IANA has registered the following YANG module in the "YANG Module
Names" subregistry [RFC6020] within the "YANG Parameters" registry:

Name: ietf-yang-structure-ext
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-structure-ext
Prefix: sx
Reference: RFC 8791

6. 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].

7. References

7.1. 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>.

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

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

[W3C.REC-xml-20081126]
             Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and
             F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth
             Edition)", World Wide Web Consortium Recommendation REC-
             xml-20081126, November 2008,
             <http://www.w3.org/TR/2008/REC-xml-20081126>.

7.2. 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";
      }
    }
  }
}
```

Below is the tree diagram of this module:

```
module: example-module

  structure address-book:
    +-- address* [last first]
       +-- last      string
       +-- first     string
       +-- street?   string
       +-- city?     string
       +-- state?    string
```

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";
    }
  }
}
```

Below is the tree diagram of this module:

```
module: example-module-aug

  augment-structure /exm:address-book/exm:address:
    +-- county?    string
    +-- zipcode?   string
```

A.3. XML Encoding Example

This example shows how an address book can be encoded in XML
[W3C.REC-xml-20081126]:

```
<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>
    <last>Root</last>
    <first>Charlie</first>
    <street>4711 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"
    },
    {
      "city": "Bedrock",
      "example-module-aug:zipcode": "70777",
      "first": "Charlie",
      "last": "Root",
      "street": "4711 Cobblestone Way"
    }
  ]
}
```

A.5. "structure" Example That Defines a Non-top-level Structure

The following example defines a data structure with error information that can be included in an <error-info> element in an <rpc-error>:

```
module example-error-info {
  yang-version 1.1;
  namespace "urn:example:example-error-info";
  prefix exei;

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

  sx:structure my-example-error-info {
    leaf error-code {
      type uint32;
    }
  }

}
```

The example below shows how this structure can be used in an <rpc-error>:

```
<rpc-reply message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <my-example-error-info
          xmlns="urn:example:example-error-info">
```

```
            <error-code>42</error-code>
          </my-example-error-info>
        </error-info>
      </rpc-error>
    </rpc-reply>
```

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Cisco

Email: mbj+ietf@4668.se

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net