                   Defining and Using Metadata with YANG
                    draft-lhotka-netmod-yang-metadata-01

Abstract

   This document defines a YANG extension statement that allows for
   defining the syntax of metadata annotations in YANG modules. The
   document also specifies the XML and JSON encoding of annotations and
   other rules for annotating instances of YANG data nodes.

Status of This Memo

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

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at 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 1, 2015.

Copyright Notice

Table of Contents

1. Introduction

   There is a need to be able to annotate instances of YANG [3] data
   nodes with metadata. Typical use cases are:

   o Deactivating a subtree in a configuration datastore while keeping
      the data in place.

   o Complementing data model information with instance-specific data.

   o RPC operations may use metadata annotations for various purposes
      in both requests and responses. For example, the <edit-config>
      operation in the NETCONF protocol (see section 7.2 of [5]) uses
      annotations in the form of XML attributes for identifying the
      point in the configuration and type of the operation.

   However, metadata annotations could potentially lead to
   interoperability problems if they are used in an ad hoc way by
   different organizations and/or without proper documentation. A sound
   metadata framework for YANG should therefore satisfy these
   requirements:

   1. The set of annotations must be extensible in a distributed manner
       so as to allow for defining new annotations without running into

      the risk of collisions with annotations defined and used by
      others.

   2. Syntax and semantics of annotations must be documented and the
       documentation must be easily accessible.

   3. Clients of network management protocols such as NETCONF [5] or
       RESTCONF [10] must be able to learn all annotations supported by
       a given server and identify each of them correctly.

   This document proposes a systematic way for defining the syntax of
   metadata annotations. For this purpose, YANG extension statement
   "annotation" is defined in the module "ietf-yang-metadata"
   (Section 6). Other YANG modules importing this module can use the
   "annotation" statement for defining the syntax one or more
   annotations.

   Semantics of metadata annotations MUST be defined separately. How it
   is done is outside the scope of this document.

   The benefits of defining the syntax of metadata annotations in a YANG
   module are the following:

   o Each annotation is bound to a YANG module name, namespace URI and
      prefix. This makes its encoding in instance documents (both XML
      and JSON) straightforward and consistent with the encoding of YANG
      data node instances.

   o Annotations are indirectly registered through IANA YANG module
      registration.

   o Annotations are included in the data model. Specifically, servers
      indicate syntactic support for certain annotations using standard
      module advertisement methods, such as the <hello> message in
      NETCONF.

   o Values of annotations are not limited to strings; any YANG built-
      in or derived type may be used for them.

2. 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 [1].

   The following terms are defined in [5]:

   o client,

o datastore,

o message,

o operation,

o server.

The following terms are defined in [3]:

o anyxml,

o built-in type,

o derived type,

o container,

o data model,

o data node,

o derived type,

o extension,

o leaf-list,

o list,

o module,

o RPC operation,

o submodule,

o type.

The following terms are defined in [8]:

o attribute,

o document,

o element,

o namespace,

o prefix.

The following terms are defined in [6]:

o array,

o member,

o object,

o primitive type.

In the following text, XML element names and YANG extension
statements are always written with explicit namespace prefixes that
are assumed to be bound to URI references as shown in Table 1.

```
+--------+-----------------------------------------------+
| Prefix | URI Reference                                 |
+--------+-----------------------------------------------+
| rng    | http://relaxng.org/ns/structure/1.0           |
| md     | urn:ietf:params:xml:ns:yang:ietf-yang-metadata |
| ein    | http://example.org/example-inactive           |
+--------+-----------------------------------------------+
```

   Table 1: Used namespace prefixes and corresponding URI references

3. Defining Annotations in YANG

Metadata annotations are defined with YANG extension statement
"md:annotation". This YANG language extension is defined in the
module "ietf-yang-metadata" (Section 6).

Substatements of "md:annotation" are shown in Table 2. They are all
core YANG statements, and the numbers in the second column refer to
the corresponding sections in RFC 6020 [3] where each statement is
described.

```
         +-------------+-----------------+-------------+
         | substatement | RFC 6020 section | cardinality |
         +-------------+-----------------+-------------+
         | description | 7.19.3          | 0..1        |
         | if-feature  | 7.18.2          | 0..n        |
         | reference   | 7.19.4          | 0..1        |
         | status      | 7.19.2          | 0..1        |
         | type        | 7.6.3           | 0..1        |
         | units       | 7.3.3           | 0..1        |
         +-------------+-----------------+-------------+
```

            Table 2: Substatements of "md:annotation".

Using the "type" statement, a type may be specified for the
annotation value according to the same rules as for YANG leaf or
leaf-list types. However, the "type" statement is optional as a
substatement of "md:annotation" statement. If it is not present, the
built-in "string" type is the default.

The "if-feature" statement, if present, makes the annotation
conditional: it is supported only by servers that advertise the
corresponding feature.

For example, the following module defines the "inactive" annotation:

```
module example-inactive {
  namespace "http://example.org/example-inactive";
  prefix "ein";
  import ietf-yang-metadata {
    prefix "md";
  }
  md:annotation inactive {
    type boolean;
    description
      "If this annotation is attached to a configuration data node,
       then the data subtree rooted at this node is deactivated.";
  }
}
```

By advertising a YANG module in which metadata annotation X is
defined using the "md:annotation" statement, a server specifies just
the syntax of annotation X. This means that configuration or state
data, RPC messages and notifications will be considered syntactically
valid if annotation X is attached to any data node instance, provided
that all rules stated in this document are observed. However, the
semantics of an annotation including the expected behavior of clients
and servers MUST be specified by other means that are outside the
scope of this document.

4. The Encoding of Annotations

   XML attributes are a natural choice for encoding metadata in XML
   instance documents. For JSON [6], there is no generally established
   method for encoding metadata. This document thus introduces a
   special encoding method that is consistent with the JSON encoding of
   YANG data node instances as defined in [7].

4.1. XML Encoding

   Metadata annotations are added to XML-encoded instances of YANG data
   nodes as XML attributes according to these rules:

   o The local name of the attribute SHALL be the same as the name of
      the annotation specified in the argument of the corresponding
      "md:annotation" statement.

   o The namespace of the attribute SHALL be identified by the URI that
      appears as the argument of the "namespace" statement in the YANG
      module where the annotation is defined. It is RECOMMENDED that
      the prefix specified by the "prefix" statement in the same module
      is used in the qualified name of the attribute.

   o The attribute value SHALL be encoded in the same way as the value
      of a YANG leaf instance having the same type.

   For example, the "inactive" annotation defined in Section 3 may be
   encoded as follows:

       <foo xmlns:ein="http://example.org/example-inactive"
            ein:inactive="true">
         ...
       </foo>

4.2. JSON Encoding

   The JSON metadata encoding defined in this section has the following
   properties:

   1. The encoding of YANG data node instances as defined in [7] does
       not change.

   2. Namespaces of metadata annotations are encoded in the same way as
       namespaces of YANG data node instances, see [7].

4.2.1. Metadata Object and Annotations

   All metadata annotations assigned to a YANG data node instance are
   encoded as members (name/value pairs) of a single JSON object,
   henceforth denoted as the metadata object. The placement and name of
   this object depends on the type of the data node as specified in the
   following subsections.

   The name of a metadata annotation (as a member of the metadata
   object) SHALL be of the following form:

        MODULE_NAME:LOCAL_NAME

   where MODULE_NAME is the name of the YANG module in which the
   annotation is defined, and LOCAL_NAME is the name of the annotation
   specified in the argument of the corresponding "md:annotation"
   statement.

   Note that unlike YANG data node instances, for annotations the
   explicit namespace identifier (MODULE_NAME) must always be used.

   The value of a metadata annotation SHALL be encoded in exactly the
   same way as the value of a YANG leaf node having the same type as the
   annotation.

4.2.2. Adding Annotations to Container and List Instances

   For an instance that is encoded as a JSON object (i.e., a container
   or list entry), the metadata object is added as a new member of that
   object with the name "@".

   Examples:

   o "cask" is a container node:

        "cask": {
          "@": {
            "example-inactive:inactive": true
          },
          ...
        }

   o "seq" is a list whose key is "name", annotation "inactive" is
      added only to the first entry:

```
    "seq": [
      {
        "@": {
          "example-inactive:inactive": true
        },
        "name": "one",
        ...
      },
      {
        "name": "two",
        ...
      }
    ]
```

4.2.3. Adding Annotations to Leaf or Anyxml Instances

For a leaf or anyxml instance, the metadata object is added as a
sibling name/value pair whose the name is the symbol "@" concatenated
with the name of the leaf or anyxml member that is being annotated.

For example, if "flag" is a leaf node of the "boolean" type:

```
    "flag": true,
    "@flag": {
      "example-inactive:inactive": true
    }
```

4.2.4. Adding Annotations to Leaf-list Instances

For a leaf-list instance, which is represented as a JSON array with
values of a primitive type, annotations may be assigned to one or
more entries by adding a name/array pair as a sibling the leaf-list
instance, where the name is the symbol "@" concatenated with the name
of the leaf-list that is being annotated, and the value is a JSON
array whose i-th element is the metadata object with annotations
assigned to the i-th entry of the leaf-list instance, or null if the
i-th entry has no annotations.

Trailing null values in the array, i.e., those following the last
non-null metadata object, MAY be omitted.

For example, in the following leaf-list instance with four entries,
the "inactive" annotation is added to the second and third entry in
the following way:

```
      "bibliomod:folio": [6, 3, 7, 8],
      "@bibliomod:folio": [
        null,
        {"example-inactive:inactive": true},
        {"example-inactive:inactive": true}
      ]
```

5. **Representing Annotations in DSDL Schemas**

   RFC 6110 [4] defines a standard mapping of YANG data models to
   Document Schema Definition Languages (DSDL) [9]. This section
   specifies the mapping for the extension statement "md:annotation"
   (Section 6), which enables validation of XML instance documents
   containing metadata annotations.

   The first step of the DSDL mapping procedure, i.e., the
   transformation of the YANG data model to the hybrid schema (see
   sec. 6 in [4]), is modified as follows:

   1. If the data model contains at least one "md:annotation"
       statement, then a RELAX NG named pattern definition MUST be added
       as a child of the root <rng:grammar> element in the hybrid
       schema. It is RECOMMENDED to use the name "__yang_metadata__"
       for this named pattern.

   2. A reference to the named pattern described in item 1 MUST be
       included as a child of every <rng:element> pattern that
       corresponds to a container, leaf, list or leaf-list data node.

   3. Every metadata annotation definition in the form

       md:annotation ARGUMENT;

       or

       md:annotation ARGUMENT {
          ...
       }

       is mapped to the following RELAX NG pattern:

       <rng:attribute name="PREFIX:ARGUMENT">
          ...
       </rng:attribute>

       where PREFIX is the namespace prefix bound to the namespace URI
       of the YANG module that contains the "md:annotation" statement.
       Each "rng:attribute" pattern SHALL be wrapped in the

<rng:optional> pattern, and this SHALL be inserted as a child of
the named pattern definition described in item 1.

4. Substatements of "md:annotation", if there are any, SHALL be
    mapped to children of the "rng:attribute" pattern exactly as
    described in sec. 10 of [4].

For example, the named pattern definition (item 1), when constructed
only for the "inactive" annotation, will have the following form:

```
<rng:define name="__yang_metadata__">
  <rng:optional>
    <rng:attribute name="ein:inactive">
      <rng:choice>
        <rng:value>true</rng:value>
        <rng:value>false</rng:value>
      </rng:choice>
    </rng:attribute>
  </rng:optional>
</rng:define>
```

Every "rng:element" pattern that corresponds to a container, leaf,
list or leaf-list data node will then contain a reference to the
above named pattern, for example

```
<rng:element name="foo:bar">
  <rng:ref name="__yang_metadata__"/>
  ...
</rng:element>
```

Note that it is not necessary to use such a reference for
"rng:element" patterns corresponding to anyxml data nodes because
they already permit any XML attributes to be attached to their
instances.

The second step of the DSDL mapping procedure, i.e., the
transformation of the hybrid schema to RELAX NG, Schematron and DSRL
schemas, is unaffected by the inclusion of "md:annotation".

6. Metadata YANG Module

RFC Editor: In this section, replace all occurrences of 'XXXX' with
the actual RFC number and all occurrences of the revision date below
with the date of RFC publication (and remove this note).

<CODE BEGINS> file "ietf-yang-metadata@2015-02-28.yang"

module ietf-yang-metadata {

```
   namespace "urn:ietf:params:xml:ns:yang:ietf-yang-metadata";

   prefix "md";

   organization
     "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

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

      WG Chair: Thomas Nadeau
                <mailto:tnadeau@lucidvision.com>

      WG Chair: Juergen Schoenwaelder
                <mailto:j.schoenwaelder@jacobs-university.de>

      Editor:   Ladislav Lhotka
                <mailto:lhotka@nic.cz>";

   description
     "This YANG module defines an extension statement that allows for
      defining metadata annotations.

      Copyright (c) 2014 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.";

   revision 2015-02-28 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: Defining and Using Metadata with YANG";
   }

   extension annotation {
     argument name;
     description
       "This extension allows for defining metadata annotations in
```

         YANG modules. The 'md:annotation' statement can appear only at
         the top level of a YANG module.

         An annotation defined with this extension statement inherits
         the namespace and other context from the YANG module in which
         it is defined.

         Other properties of the annotation and documentation may be
         specified using the following standard YANG substatements (all
         are optional): 'description', 'if-feature', 'reference',
         'status', 'type' and 'units'. If the 'type' statement is not
         present, the built-in 'string' type is used by default.

         A server announces syntactic support for a particular
         annotation by including the module in which the annotation is
         defined among the advertised YANG modules (e.g. in NETCONF
         hello message).

         The 'description' and/or 'reference' statements should provide
         links to the specification of the annotation's semantics.

         XML and JSON encoding of annotations is defined in
         RFC XXXX.";
     }
   }

   <CODE ENDS>

7. IANA Considerations

   RFC Ed.: In this section, replace all occurrences of 'XXXX' with the
   actual RFC number (and remove this note).

   This document registers the following namespace URI in the IETF XML
   registry [2]:

   URI: urn:ietf:params:xml:ns:yang:ietf-yang-metadata

   Registrant Contact: The IESG.

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

   This document registers the following YANG module in the YANG Module
   Names registry [3]:

```
     -------------------------------------------------------------------
     name:         ietf-yang-metadata
     namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-metadata
     prefix:       md
     reference:    RFC XXXX
     -------------------------------------------------------------------
```

8. Security Considerations

This document introduces a mechanism for defining the syntax of
metadata annotations in YANG modules and using them with instances of
YANG data nodes. By itself, this mechanism represents no security
threat. Security implications of a particular annotation defined
using this mechanism have to be duly considered and documented in the
specification of the annotation's semantics.

9. References

9.1. Normative References

[1]   Bradner, S., "Key words for use in RFCs to Indicate
           Requirement Levels", BCP 14, RFC 2119, March 1997.

[2]   Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
           January 2004.

[3]   Bjorklund, M., "YANG - A Data Modeling Language for the
           Network Configuration Protocol (NETCONF)", RFC 6020,
           October 2010.

[4]   Lhotka, L., "Mapping YANG to Document Schema Definition
           Languages and Validating NETCONF Content", RFC 6110,
           February 2011.

[5]   Enns, R., Bjorklund, M., Schoenwaelder, J., and A.
           Bierman, "Network Configuration Protocol (NETCONF)", RFC
           6241, June 2011.

[6]   Bray, T., "The JavaScript Object Notation (JSON) Data
           Interchange Format", RFC 7159, March 2014.

[7]   Lhotka, L., "JSON Encoding of Data Modeled with YANG",
           draft-ietf-netmod-yang-json-03 (work in progress),
           February 2015.

   [8]   Cowan, J. and R. Tobin, "XML Information Set (Second
              Edition)", World Wide Web Consortium Recommendation REC-
              xml-infoset-20040204, February 2004,
              <http://www.w3.org/TR/2004/REC-xml-infoset-20040204>.

9.2. Informative References

   [9]   International Organization for Standardization, "Document
              Schema Definition Languages (DSDL) - Part 1: Overview",
              ISO/IEC 19757-1, November 2004.

   [10]   Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", draft-ietf-netconf-restconf-04 (work in
              progress), January 2015.

Appendix A. Change Log

   RFC Editor: Remove this section upon publication as an RFC.

A.1. Changes Between Revisions -00 and -01

   o Encoding of annotations for anyxml nodes was changed to be the
      same as for leafs. This was necessary because anyxml value now
      needn't be an object.

   o It is stated that "md:annotation" statement defines only the
      syntax of an annotation.

   o Allowed "if-feature" as a substatement of "md:annotation".

Author's Address

   Ladislav Lhotka
   CZ.NIC

   Email: lhotka@nic.cz