Internet Draft
NETMOD Working Group
Category: Informational
Expires: January 9, 2017

Vinod Kumar S
Gaurav Agrawal
Anil Kumar S N

July 8, 2016

Defining and Using Metadata for YANG compilers
draft-agv-netmod-yang-compiler-metadata-01

Abstract

This document defines mechanism to defining compiler metadata (annotations) in YANG modules using YANG extension statement.

Status of this Memo

Copyright Notice

Table of Contents

```
   1. Introduction  . . . . . . . . . . . . . . . . . . . . . . . . .  3
   2.  Terminology  . . . . . . . . . . . . . . . . . . . . . . . . .  6
     2.1.  Keywords . . . . . . . . . . . . . . . . . . . . . . . . .  6
     2.2.  Terms Defined in Other Documents . . . . . . . . . . . . .  6
     2.4.  Definitions of New Terms . . . . . . . . . . . . . . . . .  7
   3.  Defining compiler annotations in YANG  . . . . . . . . . . . .  7
     3.1.  Example Definition . . . . . . . . . . . . . . . . . . . .  8
   4.  Using Annotations  . . . . . . . . . . . . . . . . . . . . . . 10
   5.  Metadata YANG Module . . . . . . . . . . . . . . . . . . . . . 10
   8.  IANA Considerations  . . . . . . . . . . . . . . . . . . . . . 12
   9 Security Considerations  . . . . . . . . . . . . . . . . . . . . 13
   10.  Acknowledgments . . . . . . . . . . . . . . . . . . . . . . . 13
   11 References  . . . . . . . . . . . . . . . . . . . . . . . . . . 14
     11.1 Normative References  . . . . . . . . . . . . . . . . . . . 14
     11.2 Informative References  . . . . . . . . . . . . . . . . . . 14
   Authors' Addresses . . . . . . . . . . . . . . . . . . . . . . . . 16
```

1. Introduction

YANG started as a data modeling language used to model configuration data, state data, remote procedure calls, and notifications for network management protocols. The purpose of YANG was focused on expressing the data organization while being exchanged over network. It also addressed the constraints that needs to be taken care by the data set exchanged.

YANG has out grown the purpose for which it was invented. Due to its simplicity in structure and flexibility many tools have been developed which tries to ease the application development by automating the code generated corresponding to defined schema.

All the implementation related data structures / classes could be auto generated and applications only concentrate on the business logic implementation. Applications are being relieved from actual protocol implementation for exchanging information with external system.

The purpose of YANG was focused on expressing the data organization while being exchanged over network. Hence the scope of automation in application development cannot cater to data organization / processing within the system.

This gap needs to be addressed in a standardized way so that it's not compiler / utilities / platform / language specific. This enable application to be portable across multiple platforms without any additional effort with respect to data organization.

Also it is required that the mechanism of annotation should not be in-line with original YANG module/sub-module, so that it will not result in maintenance issues.

So there is a need to support compiler annotations in YANG, by which applications can instruct the YANG utilities or compilers to automate the application development related to data organization / processing. These annotations should be maintained in additional YANG module / sub-module which can be optionally consumed by supporting compilers.

Typical use cases are:

- o Feed input to YANG compiler/utilities which can be used to automate the code generation based on standard data structure or collections.

- o Enable the code generation to incorporate the design patterns required by applications.

- o Enable the data structure or collections to have multiple indexes beyond the current supported list's key(s). Since the actual implementation would required searching the data based on different leaf combinations.

- o Enable applications to model internal data organization, required for business logic implementation, and not exposed to outside world.

Usage of compiler annotations are dependent on the compiler consuming it. This draft is intended to document YANG extension to support

defining compiler annotation framework.It is outside the scope of this document about the specific compiler annotation(s) definition / usage. Individual annotation definition and usage SHOULD be standardized in other docs.

Definition and usage of compiler annotation is limited to a particular protocol or application development within a device, it has no effect on how the management information is exchanged between 2 devices over network. A server SHOULD share its YANG file(s) after removing the compiler annotations that was added for its implementation. A client MUST ignore any compiler annotations present in the YANG file(s). A client MAY redefine the compiler annotation as per its implementation requirements. Clients MAY also add new annotation depending on its implementation requirements.

This document proposes a systematic way for defining compiler metadata annotations. For this purpose, YANG extension statement "compiler-annotation" is defined in the module "agv-yang-compiler-annotation" (Section 5). Other YANG modules importing this module can use the "compiler-annotation" statement for defining one or more compiler annotations.

The benefits of defining the compiler-annotations in a YANG module are the following:

- o Applications can use YANG as a tool to design the application implementation.

- o Enhance the YANG compiler(s) capability to automate the application development process.

- o Enhance the protocol development to provide better application development framework.

- o YANG could be extended to support data modeling for protocol beyond NETCONF or RESTCONF.

Due to the rules for YANG extensions (see sec. 6.3.1 in [I-D.ietf-netmod-rfc6020bis]), compiler-annotation definitions posit relatively weak conformance requirements. The alternative of introducing a new built-in YANG mechanism for compiler annotations was considered, but it was seen as a major change to the language that is inappropriate for YANG 1.1, which was chartered as a maintenance revision. After evaluating real-life usage of compiler metadata annotations, it is designed in such way that the ABNF grammar can seamlessly adapt the current defined compiler annotations.

2. Terminology

2.1. Keywords

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].

2.2. Terms Defined in Other Documents

The following terms are defined in [RFC6241]:

- capability,
- client,
- datastore,
- message,
- protocol operation,
- server.

The following terms are defined in [I-D.ietf-netmod-rfc6020bis]:

- action,
- anydata,
- anyxml,
- built-in type,
- container,
- data model,
- data node,
- data tree,
- derived type,
- extension,
- leaf,

- leaf-list,
- list,
- module,
- RPC input and output.

2.4. Definitions of New Terms

- compiler-annotation: a single item of compiler metadata that is attached to YANG constructs.
- compiler metadata: additional information that complements a schema tree.

3. Defining compiler annotations in YANG

Compiler metadata annotations are defined by YANG extension statement "ca:compiler-annotation". This YANG language extension is defined in the module "ietf-yang-compiler-annotation" (Section 5).

Sub-statements of "ca:compiler-annotation" are shown in Table 2. They are all core YANG statements, and the numbers in the second column refer to the corresponding section in [I-D.ietf-netmod- rfc6020bis] where each statement is described.

sub-statement	RFC 6020bis section	cardinality
description	7.21.3	0..1
if-feature	7.20.2	0..n
reference	7.21.4	0..1
status	7.21.2	0..1
units	7.3.3	0..1
...	Current Section 5	1..n

Table 2: Substatements of "ca:compiler-annotation".

This draft only specifies a mechanism to define compiler metadata (annotations) in YANG modules using YANG extension statement. It provides a generic extension based mechanism, to define the annotations, specific extensions needs to be defined for specific data organization annotations as sub statement to compiler annotation extension.

An compiler annotation can be made conditional by using one or more "if- feature" statements; the compiler annotation is then consumed by

compilers and perform the desired operation in compilation.

The semantics and usage rules for a specific compiler-annotation extensions SHOULD be fully specified in another document.

A compiler-annotation MUST NOT change the schema tree semantics defined by YANG. For example, it is illegal to define and use an compiler-annotation that allows modification to data-def-stmts.

The "status" statement can be used exactly as for YANG schema nodes.

3.1. Example Definition

```
module example-yang {

  namespace "urn:ietf:params:xml:ns:yang:example-yang";

  prefix "example-yang";

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

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

  description
       "This YANG module demonstrates the usage of compiler
       annotation by any module.";

  revision 2016-07-08 {
       description
         "Initial revision.";
       reference
         "draft-agv-netmod-yang-compiler-metadata: example YANG which
          is annotated";
  }

  container candidate-servers {
     list server {
         key "name";
         unique "ip port";
         leaf name {
           type string;
         }
         leaf ip {
           type inet:ip-address;
         }
```

```
            leaf port {
              type inet:port-number;
            }
        }
     }
   }
```

The following module defines the "app-data-structure" compiler-annotation as specific compiler annotation extension:

```
   module example-compiler-annotation {

   namespace "urn:ietf:params:xml:ns:yang:example-compiler-annotation";

   prefix "example";

   import ietf-yang-compiler-annotation {
          prefix "ca";
   }

   import ietf-yang-app-data-structure-annotation {
          prefix "ds";
   }

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

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

        description
          "This YANG module demonstrates the usage of compiler
          annotation by any module.";

        revision 2016-07-08 {
          description
            "Initial revision.";
          reference
            "draft-agv-netmod-yang-compiler-metadata: example of
             defining and using compiler annotations with YANG";
        }

   ca:compiler-annotation /candidate-servers/server {
          ds:app-data-structure queue;
        }
   }
```

4. Using Annotations

By defining a YANG module in which a compiler metadata annotation is defined using the "ca:compiler-annotation" statement, an application indicates compiler to handle that compiler-annotation according to the compiler-annotation's definition. That is, the compiler-annotation uses it as input for automation of code generation or applications development.

Depending on its semantics, an annotation may have an effect only in certain schema trees and/or on specific schema node types.

A client MUST NOT use the compiler-annotation to interpret the schema even if it is advertised by a server.

5. Metadata YANG Module

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

FC Editor: Also please replace all occurrences of 'RFC 6020bis' with he actual RFC number that will be assigned to [I-D.ietf-netmod-fc6020bis].

```
CODE BEGINS> file "ietf-yang-compiler-annotation.yang"

odule ietf-yang-compiler-annotation {

amespace "urn:ietf:params:xml:ns:yang:ietf-yang-compiler-annotation";

refix "ca";

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

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

escription
 "This YANG module defines an extension statement that allows for
  defining compiler annotations.

  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 (http://tools.ietf.org/html/rfc2119).";
```

```
    revision 2016-07-08 {
    description
      "Initial revision.";
    reference
      "draft-agv-netmod-yang-compiler-metadata:
       Defining and Using compiler annotations with YANG";
    }

    extension compiler-annotation {
    argument target;
    description
      "This extension allows for defining compiler annotations for
       any body-stmts. The 'ca:compiler-annotation' statement
       contains annotations applicable to its target statement
       identified by the argument.

       It's purpose is to provide additional information to compiler
       about implementation of the modeled information.

   he argument is a string that identifies a node in the
   chema tree.  This node is called the compiler annotation's
       target node.  The target node MUST be a body-stmt as defined
       in RFC6020bis.

       It MAY be consumed by the compiler of the device supporting
       the schema.

       The compiler annotations must be defined in a seperate YANG file,
       so that there are no maintenance issues.

       The ca:compiler-annotation defined with this extension
       statement do not affect the namespace or get impacted by
       the namespace of the YANG file where it is used.

       Semantics of the annotation and other documentation can be
       specified using the following standard YANG substatements (all
       are optional):  'description', 'reference', 'status', and
       'units'.

       The presence of a 'if-feature' child to the ca:
       compiler-annotation, means the compiler consumes the
       annotation when the feature is supported by the device.

       Server SHOULD NOT share ca:compiler-annotations YANG files
       while sharing schema with a client in protocol exchange.

       Client receiving the schema from a Server in protocol
       exchange, MUST ignore the YANG files with any
```

```
       ca:compiler-annotations extension.

       There must be one or more sub statements with specific compiler
       annotation extensions. (Note: Specific compiler annotation
       extensions SHOULD be covered as a part of other standard
       documents.)

     } // compiler-annotation
    //module agv-yang-compiler-annotation

   CODE ENDS>
```

8. IANA Considerations

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).

This document registers a URI in the "IETF XML registry" [RFC3688]. Following the format in RFC 3688, the following registration has been made.

```
      ------------------------------------------------------------------
      URI: urn:agv:params:xml:ns:yang:agv-yang-compiler-annotation

      Registrant Contact: The NETMOD WG of the IETF.

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

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

```
      ------------------------------------------------------------------
      name:         agv-yang-compiler-annotation
      namespace:    urn:agv:params:xml:ns:yang:
                                            agv-yang-compiler-annotation
      prefix:       md
      reference:    RFC XXXX
      ------------------------------------------------------------------
```

9 Security Considerations

This document introduces a mechanism for defining compiler metadata annotations in YANG modules and attaching them to instances of YANG schema nodes. By itself, this mechanism represents no security threat. Security implications of a particular compiler-annotation defined using this mechanism MUST be duly considered and documented in the the compiler-annotation's definition.

10. Acknowledgments

11 References

11.1 Normative References

I-D.ietf-netmod-rfc6020bis]
Bjorklund, M., "The YANG 1.1 Data Modeling Language", draft-ietf-netmod-rfc6020bis-11 (work in progress), February 2016.

I-D.ietf-netmod-yang-json]
Lhotka, L., "JSON Encoding of Data Modeled with YANG", draft-ietf-netmod-yang-json-09 (work in progress), March 2016.

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

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

RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

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

11.2 Informative References

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-10 (work in progress), March 2016.

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997,

<http://www.rfc-editor.org/info/rfc2119>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.

Authors' Addresses

Vinod Kumar S
Huawei Technologies India Pvt. Ltd,
Near EPIP Industrial Area,
Kundalahalli Village,
Whitefield,
Bangalore - 560066

EMail: vinods.kumar@huawei.com

Gaurav Agrawal
Huawei Technologies India Pvt. Ltd,
Near EPIP Industrial Area,
Kundalahalli Village,
Whitefield,
Bangalore - 560066

EMail: gaurav.agrawal@huawei.com

Anil Kumar S N
Huawei Technologies India Pvt. Ltd,
Near EPIP Industrial Area,
Kundalahalli Village,
Whitefield,
Bangalore - 560066

EMail: anil.ietf@gmail.com