NETMOD Working Group
INTERNET-DRAFT
Intended Status: Standards Track

Anil Kumar S N
Gaurav Agrawal
Vinod Kumar S
Huawei Technologies

Expires: January 1, 2017

June 30, 2016

YANG compiler annotation for data structure and inheritance
draft-agv-netmod-yang-annotation-ds-and-derived-00

Abstract

This document defines two new YANG compiler annotations as per draft-agv-netmod-yang-compiler-metadata-00. First annotation is used to define the data structure type to be generated corresponding to a schema node. Second annotation is used to generate a user defined inherited class corresponding to a schema node in which user can override the default implementation.

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 January 1, 2017

Table of Contents

1 Introduction

YANG defines single instance data structure as container or leaf and multi instance data structure as a list or leaf-list. Mapping of the YANG single instance constructs to a programming data structure is straight forward, it can be directly mapped to an object or an entity.

Mapping of the YANG multi instance construct to a programming data structure has many option, since it is a collection or objects / entities. Depending on the application use case and development environment, it will have other requirements to be considered for mapping it to a data structure. For example it needs to be optimized for storage or it need to be optimized for search efficiency or it may be required to have multiple key combinations for search. Based on this, we can see that there is no single option to be used by YANG utilities / compilers to auto generate the code corresponding to a multi instance construct.

Applications use the YANG to document the external interface designed. Applications need to design the alternatives for data structure and choose that is best suited for them. This is part of typical software engineering activity used by application. YANG utilities / compilers need to get these design related details to automate the code generation as per application design. To support this a new YANG compiler annotation @app-data-structure is defined.

Applications need to extend the auto generated code to suit their needs, The external world communication using protocol like NETCONF / RESTONF can be automated by automating the code generation based on the YANG structure. Applications have additional business logic to be taken care, wherein they need to extend the generated code. When applications extends or modifies the generated code, YANG utilities are unaware of it, and will not be able to correctly update the auto generated files when the schema changes. In such scenarios, the YANG utilities / compilers generated code needs to be provide a framework wherein application can extend the default auto-generated implementation provided by the utilities which is not impacted even if YANG utilities / compilers auto generates files for changed schema. To support this a new YANG compiler annotation @app-derived is defined.

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 RFC 2119 [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

- @app-data-structure: annotations for type of data structure to be generated for multi instance YANG construct.
- @app-derived: annotation for derived class to be generated for extending the generated class.

3. Defining @app-data-structure compiler annotations in YANG

@app-data-structure annotation is used to define the data structure to be used for the corresponding multi-instance YANG construct.

It has the following parameter.

- data-structure: Its value will specify the data structure to be used for code generation. It is a mandatory parameter.
- key-fields: This is used to specify the space separated list of key fields. This is not applicable for leaf-list YANG construct, it is optional parameter. If it is not defined, the list's key fields will be used to index the data structure.

3.1 Example usage

Application instructs to use a map data structure for maintaining servers information.

```
list server {
   ca:compiler-annotation{
     @app-data-structure(data-structure="map", key="name");
   }

   key "name";
   unique "ip port";
   leaf name {
     type string;
   }
   leaf ip {
```

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

4 Defining @app-derived compiler annotations in YANG

@app-derived is used to generate an inherited class, which can be used by the applications to extend/override the default implementations of application interface. It has the following parameter.

o extended-name: It is used to generate the extended class, which can be used by application for implementation. This is a mandatory parameter.

4.1 Example usage

Application instructs to generate an inherited class for implementation.

```
      list server {
        ca:compiler-annotation{
          @app-data-structure(data-structure="map", key="name");
          @app-derived(extended-name=special-server);
        }

        key "name";
        unique "ip port";
        leaf name {
          type string;
        }
        leaf ip {
          type inet:ip-address;
        }
        leaf port {
          type inet:port-number;
        }
      }
```

5 Security Considerations

This document introduces two annotations for defining compiler metadata in YANG modules and attaching them to instances of YANG schema nodes. By itself, this mechanism represents no security threat.

6 IANA Considerations

No specific IANA considerations for this document

7. Acknowledgments

8 References

8.1 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>.

[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>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

8.2 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., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

Authors' Addresses

Anil Kumar S N
Huawei Technologies India Pvt. Ltd,
Near EPIP Industrial Area,
Kundalahalli Village,
Whitefield,
Bangalore - 560037

EMail: anil.ietf@gmail.com

Gaurav Agrawal
Huawei Technologies India Pvt. Ltd,
Near EPIP Industrial Area,
Kundalahalli Village,
Whitefield,
Bangalore - 560037

EMail: gaurav.agrawal@huawei.com

Vinod Kumar S
Huawei Technologies India Pvt. Ltd,
Near EPIP Industrial Area,
Kundalahalli Village,
Whitefield,
Bangalore - 560037

EMail: vinods.kumar@huawei.com