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

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:

- bc: Used as an abbreviation for a backwards-compatible change.

- nbc: Used as an abbreviation for a non-backwards-compatible change.

- editorial change: A backwards-compatible change that does not change the YANG module semantics in any way.

- YANG schema: The combined set of schema nodes for a set of YANG module revisions, taking into consideration any deviations and enabled features.

- versioned schema: A YANG schema with an associated YANG semantic version number, e.g., as might be described by a YANG package.

- 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 any many examples of where this approach has been successfully used.

4. Objectives

The goals of the schema version selection draft are:

- 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.

- 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.

- 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:

- This draft does not provide a mechanism to allow clients to choose arbitrary sets of YANG module versions to interoperate with the server.

- 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.

- 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.

- 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 exact 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.


```
// 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