Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 18, 2020

R. Wilton
R. Rahman
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia
B. Wu
Huawei

March 17, 2020

YANG Schema Selection
draft-ietf-netmod-yang-ver-selection-00

Abstract

This document defines a mechanism to allow clients, using NETCONF or RESTCONF, to configure and select YANG schema for interactions with a server. This functionality can help servers support clients using older revisions of YANG modules even if later revisions contain non-backwards-compatible changes. It can also be used to allow clients to select between YANG schema defined by different organizations.

This draft provides a solution to YANG versioning requirements 3.1 and 3.2.

Table of Contents

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 [I-D.verdt-netmod-yang-versioning-reqs], YANG Module Versioning [I-D.verdt-netmod-yang-module-versioning] and YANG Packages [I-D.rwilton-netmod-yang-packages].

This document makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

- datastore schema

This document defines the following terminology:

- 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[I-D.rwilton-netmod-yang-packages].

- schema-set: A named set of datastore schemas for supported datastores, where every datastore schema is specified as a union of compatible YANG packages. A schema-set is the basic unit of client schema selection.

2. Introduction

This document describes how NETCONF and RESTCONF clients can use

protocol extensions, along with a schema selection YANG module, to choose particular YANG schema for interactions with a server.

[I-D.verdt-netmod-yang-versioning-reqs] defines requirements that any solution to YANG versioning must have.

YANG Semver [I-D.verdt-netmod-yang-semver], based on YANG Module Versioning, specifies a partial solution to the YANG versioning requirements that focuses on using semantic versioning within individual YANG modules, but does not address all requirements listed in the YANG versioning requirements document. Of particular relevance here, requirements 3.1 and 3.2 are not addressed.

YANG Packages describes how sets of related YANG module revisions can be grouped together into a logical entity that defines a YANG schema. 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, or a vendor's YANG modules. Updated versions of these package definitions can be defined as the contents of these packages evolve over time, e.g., by using new revisions of YANG modules included in the package.

This document defines how YANG packages are used to represent versioned datastore schema, and how clients can choose which versioned schemas to use during protocol 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 schema selection, this is what the 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 document 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 are many examples of where this approach has been used successfully.

4. Objectives

The goals of YANG schema selection 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 management clients.

- 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.

- To closely align with existing NETCONF/RESTCONF protocol semantics. I.e., with the exception of the optional mechanism

that allows selection of the schema-set at the beginning of a NETCONF session or RESTCONF request, protocol interactions between client and server are the same as when schema selection is not being used.

- To allow considerable freedom for server implementations to decide how to implement schema selection, and choose the schema selection capabilities they support. In particular:
  - Servers determine which schema-sets can be selected by clients, and which combinations of schema-sets are compatible with each other during concurrent sessions/operations.
  - Servers can make some schema-sets automatically available for client selection, or require clients to configure the selectable schema-sets before they can be used.
  - Servers can limit clients to selecting only a single schema-set for all client connections, i.e., replacing the devices default schema-set, or allow clients to use different schema for concurrent sessions/operations.
  - Servers can restrict some read-write datastores to be read-only when accessed via a particular schema-set, i.e., providing a read-only view of configuration.

  - Servers may allow clients to combine schema-sets into named custom schema-sets, or only support predefined schema-sets.

The following points are non objectives of this document:

- This document 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, using YANG Semver versioning rules, 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 SHOULD be used, and otherwise an out of band mechanism SHOULD be used to indicate where a mapping has failed.

5. Solution

An outline of the solution is as follows:

1. YANG packages are used to define versioned schema that represent a partial or full datastore schema for one or more datastores.

2. Schema-sets are named structures that define a set of supported datastores, along with the schema associated with each of those datastores, specified via leaf references to YANG packages.

3. The configurable 'selectable' leaf-list defines which schema-sets may be selected by clients, and the associated configurable 'default' leaf defines the schema-set used by clients that do not select a schema-set.

4. Clients choose which selectable schema-set to use via NETCONF capability exchange during session initiation, or RESTCONF path.

5. Optionally, the configurable 'custom-schema-set' list allows clients to combine schema-sets together into new named schema-sets for selection.

5.1. Packages

Section 5.3 of YANG Packages specifies how packages SHOULD be used to represent datastore schema.

Additional guidance on how YANG packages are specified for schema selection are:

- Separate packages MAY be defined for different families of schema, e.g., SDO, OpenConfig, or vendor native.
- Separate packages MAY be defined for different versions of schema.
- All packages referenced for schema selection, and any recursively included package dependencies, MUST be available on the server in the '/packages' container in the operational state datastore.
- Globally scoped packages used for schema selection SHOULD be made available offline in YANG instance data documents, as described in section 6 of YANG Packages.

5.2. Schema-sets

A schema-set defines a set of datastore schema(s) that could potentially be used for client schema selection.

The schema-sets supported by the server are available at '/schema-set-selection/schema-set' in the operational state datastore.

Each schema-set has a unique name that identifies the schema-set during schema selection protocol operations, e.g., it is used in both the NETCONF capabilities exchange and the RESTCONF path.

A schema-set defines the set of supported datastores that are available when clients use the selected schema-set. The schema for each datastore is specified as the union of one or more compatible YANG packages. Writable datastores (e.g., running) can also be labelled as being read-only if they cannot be written to via client interactions using the schema-set.

Not all schema-sets are necessarily compatible with each other,

allowing one schema-set to be selected may prevent other schema-sets from being selected at the same time. Hence, each schema-set lists the other schema-sets that it is compatible with.

5.3. Defining and changing client selectable schema-sets

A device may have to allocate resources to support selectable schema-sets, and the selectable leaf-list '/schema-set-selection/selectable' is the mechanism to constrain the set of schema-sets that can be selected by clients.

In the operational state datastore, the 'selectable' leaf-list contains the names of all schema-sets that a client may select from. Entries in this list MAY be added by the system without prior configuration. In addition, the 'default' leaf indicates which schema-set is used by clients that do not explicitly select a schema-set.

In configuration, the 'selectable' leaf-list allows clients to configure the schema-sets that are available for clients to select from. A client can also choose the default schema-set that is used by any client that connects without naming a schema-set.

5.4. Schema selection protocol operations

5.4.1. NETCONF

The schema-set being used in a NETCONF session is established at the start of the session through the exchange of capabilities and never changes for the duration of the session. If the server can no longer support the schema-set in use in a NETCONF session, then it MUST disconnect the session.

5.4.1.1. schema-sets NETCONF capability

A new NETCONF :schema-sets capability, using base:1.1 defined in [RFC6241]

This capability is used by the server is advertise selectable schema. The server sends a comma separated list (with no white spaces) of selectable schema-sets it supports. For consistency, the list SHOULD contain the default selectable schema-set first, followed by the remaining selectable schema-sets, matching the order in the '/schema-set-selection/selectable' leaf-list.

This capability is used by clients to select a particular schema. The client sends an ordered list of selectable schema that it is willing to use.

The selected schema is the first entry in the client schema-set list that is also contained in the server schema-set list. If there is no common entry then the session is terminated with an error.

If the client does not include the schema-set capability during the capability exchange then the default selectable schema-set is used.

The :schema-sets capability is identified by the following capability string:

urn:ietf:params:netconf:capability:schema-sets:1.0

In this example, the server advertises its (abbreviated) <hello> as follows. This indicates that the server supports the following schema-sets:

example-ietf-routing: Versions 2.1.0 (default) and 1.3.1

example-vendor-xxx: Versions 9.2.3 and 8.4.2

Some extra white spaces have been added for display purposes only.

```
 <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <capabilities>
 <capability>urn:ietf:params:netconf:base:1.0</capability>
 <capability>urn:ietf:params:netconf:base:1.1</capability>
 <capability>
 urn:ietf:params:netconf:capability:schema-sets:1.0?list=
 example-ietf-routing@2.1.0,example-ietf-routing@1.3.1,
 example-vendor-xxx@9.2.3,example-vendor-xxx@8.4.2
 </capability>
 </capabilities>
 </hello>
```

The client advertises its (abbreviated) <hello> as follows. This indicates the the client prefers to use the example-ietf-routing schema version 2.1.0, but can also use version 1.3.1. Because both the client and server support version 2.1.0, and because the client listed it first, the selected schema-set is example-ietf-routing@2.1.0: Some extra white spaces have been added for display purposes only.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>
urn:ietf:params:netconf:capability:schema-sets:1.0?list=
example-ietf-routing@2.1.0,example-ietf-routing@1.3.1
</capability>
</capabilities>
</hello>
```

5.4.2. RESTCONF

For RESTCONF, schema-sets are chosen via use of their name in the request URI.

Clients select the desired schema-set by choosing the corresponding RESTCONF root resource. This is done by appending the schema-set name to the RESTCONF API root [RFC8040]. This updates Section 3.1 of [RFC8040] and Section 3 of [RFC8527].

Clients will use RESTCONF root resource {+restconf}/schema/<schema-name> for all requests pertaining to resources specific to a schema-set. Consider a device which supports schema-sets vendor-schema@3.0.0, vendor-schema@2.1.0 and vendor-schema@1.4.5: clients will use RESTCONF root resource {+restconf}/schema/vendor-schema@X.Y.Z for all requests pertaining to resources defined in schema-set vendor-schema@X.Y.Z. This applies to all RESTCONF resources defined in vendor-schema@X.Y.Z.

Here are some examples where {+restconf} is /restconf/.

Examples for servers which are NOT NMDA-compliant as per [RFC8527]:

1. /restconf/schema/vendor-schema@X.Y.Z/data for datastore resources.

2. /restconf/schema/vendor-schema@X.Y.Z/data/module-A:data-X for data resource "data-X" defined in module "module-A".

3. /restconf/schema/vendor-schema@X.Y.Z/operations/module-B:op-Y for RPC "op-Y" defined in module "module-B"

4. /restconf/schema/vendor-schema@X.Y.Z/data/module-C:containerZ/myaction for action "myaction" defined in top-container "containerZ" of module "module-C"

Examples for servers which are NMDA-compliant as per [RFC8527]:

1. /restconf/schema/vendor-schema@X.Y.Z/ds/<datastore>/ for datastore resources, e.g. /restconf/schema/vendor-schema@X.Y.Z/ds/ietf-datastores:running refers to the Running configuration datastore.

2. /restconf/schema/vendor-schema@X.Y.Z/ds/ietf-datastores:operational for YANG actions.

If the chosen schema-set is not available then an error response containing a "404 Not Found" status-line MUST be returned by the server.

5.5. Custom schema-sets

Custom schema-sets, if supported by the server, allow clients to:

- Configure client meaningful names for selectable schema-sets.

- Combine compatible schema-sets together into a combined named schema-set that is then selectable by clients. E.g. a client might want to use a combination of standard configuration models along with vendor configuration models to manage functionality not covered by the standard models.

6. Schema selection from a server perspective

The general premise of this solution is that servers generally implement one native schema, and the schema 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 device's native schema version.

Datastore validation MAY need to be performed in two places, firstly in whichever schema a client 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.

7. Schema selection from a client perspective

Clients can use configuration to choose which schema-sets are available for selection.

Clients cannot choose arbitrary individual YANG module versions, and are instead constrained by the versions that the server makes available via schema-sets.

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 using the selected schema. If the client queries YANG library, then the version of YANG Library that is returned matches the schema-set that has been selected by the client.

The selection of a schema-set by a client MUST NOT change the behaviour of the server experienced by other clients. For example, the get-data response to one client MUST be the same before and after another client selects a schema-set.

The server may not support a schema with the exact version desired by the client, and the client may have to choose 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 expect that all data nodes in later module versions can always be backported to older schema versions.

7.1. Examples

Here is a simple example of a NETCONF client migrating to a new schema-set with a server that has multiple schema-sets in the 'selectable' leaf-list:

1. Disconnect the current session

2. Reconnect and select a new schema-set from the 'selectable' leaf-list

If a server, for example, only supports a single schema-set at a time by only allowing a single entry in the 'selectable' leaf-list (the default), then a change of the schema-set in the 'selectable' leaf-list (and default) would cause all previously established NETCONF sessions (using the previous 'selectable' schema-set) to be disconnected.

If a server only supports a single schema-set at a time (across all sessions), a NETCONF client can migrate to a new schema-set by using the following sequence of steps:

1. Configure a new schema-set in the 'selectable' leaf-list, remove the old schema-set, and set the 'default' leaf to that new schema-set. Other configuration can also be done in the same request (using the current schema-set in use on the session).

2. The server will process the request and then disconnect the session (since the current schema-set of the session can no longer be supported). All other NETCONF sessions would also be disconnected at this point.

3. The client reconnects using the new schema-set (either by selecting it during capability exchange, or by using no selection and relying on the new default schema-set).

4. The client can then optionally send a complete new configuration using the new schema (i.e. if the client knows that the server can't perfectly convert everything from the old schema to the new schema).

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 Selection YANG module

The YANG schema selection YANG module is used by a server to report the schema-sets that are generally available, and to allow clients to configure which schema-sets are available for client selection and which is the default.

Custom schema-sets, if supported, allows clients to configure custom combinations of schema-sets that can then be selected by clients.

The "ietf-schema-selection" YANG module has the following structure:

```
module: ietf-schema-selection
  +--rw schema-set-selection!
     +--rw selectable*   -> /schema-set-selection/schema-set/name
     +--rw default       -> /schema-set-selection/selectable
     +--rw custom* [name] {custom-schema-set}?
```

```
      |  +--rw name                string
      |  +--rw description?        string
      |  +--rw included-schema*
      |           -> /schema-set-selection/schema-set/name
      +--ro schema-set* [name]
         +--ro name                   string
         +--ro partial?               empty
         +--ro datastore* [name]
         |  +--ro name          ds:datastore-ref
         |  +--ro read-only?    empty
         |  +--ro package* [name version]
         |     +--ro name         -> /pkgs:packages/package/name
         |     +--ro version      leafref
         |     +--ro checksum?    leafref
         +--ro selectable-with*
         |       -> /schema-set-selection/schema-set/name
         +--ro custom-selectable! {custom-schema-set}?
            +--ro combinable-with*
                    -> /schema-set-selection/schema-set/name
```

10. YANG Module

The YANG module definition for the module described in the previous sections.

```
<CODE BEGINS> file "ietf-schema-selection@2020-02-29.yang"
module ietf-schema-selection {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-schema-selection";
  prefix "schema";

  import ietf-yang-revisions {
    prefix rev;
    reference "XXXX: Updated YANG Module Revision Handling";
```



```
  }

  import ietf-datastores {
    prefix ds;
```

```
    rev:revision-or-derived 2018-02-14;
    reference
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }

  import ietf-yang-packages {
    prefix pkgs;
    rev:revision-or-derived 0.2.0;
    reference "RFC XXX: YANG Packages.";
  }


  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>

     Author:   Joe Clarke
               <mailto:jclarke@cisco.com>

     Author:   Jason Sterne
               <mailto:jason.sterne@nokia.com>

     Author:   Bo Wu
               <mailto:lana.wubo@huawei.com>";

  description
    "This module provide a data model to advertise and allow the
     selection of schema versions by clients.

     Copyright (c) 2019 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.

      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.";


  // 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 2020-02-29 {
    description
      "Initial revision";
    reference
      "RFC XXXX: YANG Schema Selection";
  }

  /*
   * Features
   */
  feature "custom-schema-set" {
    description
      "Feature to choose whether clients may configurable custom
       schema definitions.";
  }

  /*
   * Top level data nodes.
   */

  container schema-set-selection {
    presence "Enable schema selection";

    description
      "YANG schema-set selection.

       Contains configuration and state related to client selectable
       YANG schema-sets.";
```

```
      leaf-list selectable {
```



```
        type leafref {
          path "/schema-set-selection/schema-set/name";
          require-instance false;
        }
        min-elements 1;

        description
          "Specifies the selectable schema used by this server, that
           can be selected by clients (either through NETCONF
           capability negotiation or RESTCONF schema specific path).";
      }

      leaf default {
        type leafref {
          path "/schema-set-selection/selectable";
        }
        mandatory true;
        description
          "Defines the default schema-set used by this server.

           This is the schema-set that is chosen if a NETCONF client
           doesn't select a schema during capability negotiation, or if
           the standard RESTCONF (or NMDA datastore URLs) are used.";
      }

      list custom {
        if-feature "custom-schema-set";
        key "name";

        description
          "Defines a custom selectable schema constructed from
           compatible schema";

        leaf name {
          type "string";
          description
            "Name of custom schema.

             Format can either be the form of a YANG identifer, or
             '<name>@<rev-label>'.
```

```
            The selectable-schema name is used in NETCONF capabilties
            negotiation and also the RESTCONF path (XXX, is encoding
            required, e.g. for '@'?)";
        }

        leaf description {
          type string;
```



```
          description
            "The description associated with this custom package.";
        }

        leaf-list included-schema {
          type leafref {
            path "/schema-set-selection/schema-set/name";
            require-instance false;
          }
          description
            "Lists the schema that are combined together into a single
             selectable schema (i.e. via a union operation on each
             datastore schema package).";
        }
      }

      list schema-set {
        key "name";
        config false;

        description
          "Lists all available schema-set's that can be used in schema
           selection.";

        leaf name {
          type string;

          description
            "Name of the schema.

             Do we restrict what is allowed, specifically, do we allow
             '@'";
        }
```

```
        leaf partial {
          type empty;

          description
            "Indicates that this schema-set only represents a subset of
             the full configurable schema of the device.";
        }

        list datastore {
          key "name";

          description
            "The list of datastores supported for this pan-datastore
             selectable-package, along with the package schema
```



```
             associated with that datastore.";

          leaf name {
            type ds:datastore-ref;
            description
              "The datastore that this datastore schema is associated
               with.";
            reference
              "RFC 8342: Network Management Datastore Architecture
               (NMDA)";
          }

          leaf read-only {
            type empty;
            description
              "For datastores that are normally writable, the read-only
               flag indicates that the datastore cannot be written
               using this schema-set.  E.g., if <running> was a
               supported datastore then it could be read, but not
               written.

               This flag is not required for datastores, like
               operational, that are defined as being read-only.";
          }

          uses pkgs:yang-ds-pkg-ref;
```

```
        }

        leaf-list selectable-with {
          type leafref {
            path "/schema-set-selection/schema-set/name";
          }
          description
            "Lists other schema-sets that MAY be selected at the same
             time as this schema.";
        }

        container custom-selectable {
          if-feature "custom-schema-set";
          presence
            "This schema MAY be selected as part of a custom
             schema-set.";
          description
            "Indicates that this schema-set is selectable as part of a
             custom schema-set and also lists other schema-sets that
             may be combined together into a custom schema-set.";

          leaf-list combinable-with {
```



```
            type leafref {
              path "/schema-set-selection/schema-set/name";
            }
            description
              "Lists the schema-sets that MAY be combined with this
               schema into a single custom schema-set'.";
          }
        }
      }
    }
  }
  <CODE ENDS>
```

11. Security Considerations

To be defined.

12. IANA Considerations

This document requests IANA to registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations are requested.

URI: urn:ietf:params:xml:ns:yang:ietf-schema-selection
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document requests that the following YANG modules are added in the "YANG Module Names" registry [RFC6020]:

Name: ietf-schema-selection.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-schema-selection
Prefix: schema
Reference: RFC XXXX

This document registers a URI.

12.1. NETCONF Capability URNs

This document registers a URI in the IETF XML registry [RFC3688]. The IANA registry "Network Configuration Protocol (NETCONF) Capability URNs" needs to be updated to include the following capability.


```
Index
    Capability Identifier
-------------------------
:schema-sets
    urn:ietf:params:netconf:capability:schema-sets:1.0
```

13. Open Questions/Issues

All issues, along with the draft text, are currently being tracked at: https://github.com/netmod-wg/yang-ver-dt/labels/version-selection-solution

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-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-08 (work in progress), March 2020.

[I-D.rwilton-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Packages", draft-rwilton-netmod-yang-packages-03 (work in progress), February 2020.

[I-D.verdt-netmod-yang-module-versioning]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "Updated YANG Module Revision Handling", draft-verdt-netmod-yang-module-versioning-01 (work in progress), October 2019.

[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", draft-verdt-netmod-yang-semver-01 (work in progress), October 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>.

[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>.

[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>.

[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>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

[RFC8527] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "RESTCONF Extensions to Support the Network Management Datastore Architecture", RFC 8527, DOI 10.17487/RFC8527, March 2019, <https://www.rfc-editor.org/info/rfc8527>.

15.2. Informative References

[I-D.ietf-netmod-artwork-folding]
Watsen, K., Auerswald, E., Farrel, A., and Q. WU, "Handling Long Lines in Inclusions in Internet-Drafts and RFCs", draft-ietf-netmod-artwork-folding-12 (work in progress), January 2020.

[I-D.verdt-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-verdt-netmod-yang-solutions-03 (work in progress), February 2020.

Appendix A. Schema selection examples

Some common simplifications have been made to examples in this section:

- Simplified package definitions have been defined, e.g., the packages do not include other packages, and the module lists have been elided.

- Package and module checksums have been omitted to minimize line wrapping.

A.1. Supporting older versions of a schema

To facilitate an easier migration path for clients, a server may choose to support older versions of a schema, for example this might be done to minimize impact on clients if non-backwards-compatible changes have been made between schema versions.

It is expected that the server can dynamically translate from the older schema version to the latest, but this is not possible in all cases (e.g., if support for some functionality has been removed from the server), when deviations against the old schema version may be required to accurately describe the supported functionality.

This example illustrates a device that is capable of supporting three different versions of its vendor-schema, reported in '/schema-set-selection/schema-set':

vendor-schema@3.0.0, the device defaults to the latest version of the schema

vendor-schema@2.1.0, a preceding NBC software version

vendor-schema@1.4.5, a preceding NBC software version

Schema selection data from the operational state datastore:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <data>
    <packages>
      <package>
        <name>vendor-schema</name>
        <version>3.0.0</version>
        <!-- package modules elided -->
      </package>
      <package>
        <name>vendor-schema</name>
        <version>2.1.0</version>
        <!-- package modules elided -->
      </package>
      <package>
        <name>vendor-schema</name>
        <version>1.4.5</version>
        <!-- package modules elided -->
      </package>
    </packages>
    <schema-set-selection>
      <schema-set>
        <name>vendor-schema@3.0.0</name>
        <datastore>
          <name>ds:running</name>
          <package>
            <name>vendor-schema</name>
            <version>3.0.0</version>
          </package>
        </datastore>
        <datastore>
```

```
            <name>ds:operational</name>
            <package>
              <name>vendor-schema</name>
              <version>3.0.0</version>
            </package>
          </datastore>
```



```
          <selectable-with>vendor-schema@1.4.5</selectable-with>
          <selectable-with>vendor-schema@2.1.0</selectable-with>
        </schema-set>
        <schema-set>
          <name>vendor-schema@2.1.0</name>
          <datastore>
            <name>ds:running</name>
            <package>
              <name>vendor-schema</name>
              <version>2.1.0</version>
            </package>
          </datastore>
          <datastore>
            <name>ds:operational</name>
            <package>
              <name>vendor-schema</name>
              <version>2.1.0</version>
            </package>
          </datastore>
          <selectable-with>vendor-schema@1.4.5</selectable-with>
          <selectable-with>vendor-schema@3.0.0</selectable-with>
        </schema-set>
        <schema-set>
          <name>vendor-schema@1.4.5</name>
          <datastore>
            <name>ds:running</name>
            <package>
              <name>vendor-schema</name>
              <version>1.4.5</version>
            </package>
          </datastore>
          <datastore>
            <name>ds:operational</name>
            <package>
              <name>vendor-schema</name>
```

```
            <version>1.4.5</version>
          </package>
        </datastore>
        <selectable-with>vendor-schema@2.1.0</selectable-with>
        <selectable-with>vendor-schema@3.0.0</selectable-with>
      </schema-set>
    </schema-set-selection>
  </data>
</rpc-reply>
```

The client may configure the device to instead use an earlier version of the schema-set, 'vendor-schema@1.4.5'. By configuring only a

single selectable schema-set, and making it the default, means that this can be the only schema-set that clients use to interact to the device using. On committing this configuration change, all existing NETCONF sessions (that were previously interacting using vendor-schema@3.0.0) will be closed, and clients need to reconnect, at which point they will interact with schema-set vendor-schema@1.4.5, and also see the contents of YANG library updated to reflect the datastore-schema reported in vendor-schema@1.4.5.

Configuration a new default selectable schema:

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <schema-set-selection>
    <selectable>vendor-schema@1.4.5</selectable>
    <default>vendor-schema@1.4.5</default>
  </schema-set-selection>
</config>
```

A.1.1. Variation - Multiple selectable schema-sets

The client may wish to interact with multiple different versions of the schema at the same time. E.g., this could be achieved with the following configuration:

Configuring multiple selectable schema:

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <schema-set-selection>
    <selectable>vendor-schema@1.4.5</selectable>
    <selectable>vendor-schema@3.0.0</selectable>
    <default>vendor-schema@1.4.5</default>
  </schema-set-selection>
</config>
```

As before, by default clients will use the vendor-schema@1.4.5, but by using NETCONF capabilities exchange, or through use of the RESTCONF path that may select version-schema@3.0.0 instead. Clients could also explicitly select vendor-schema@1.4.5, even though it is also the default schema-set.

A.2. Supporting different schema families

Some devices may allow clients to configure the device using different YANG schema (e.g. vendor native, vs IETF, vs OpenConfig).

This example illustrates a device is that capable of supporting 3 different schema families (native, oc, ietf).

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
<data>
  <packages>
    <package>
      <name>vendor-schema</name>
      <version>1.0.0</version>
      <module>
        <name>vendor-interfaces</name>
      </module>
      <!--Module list of vendor YANG modules elided -->
```

```
        </package>
        <package>
          <name>ietf-schema</name>
          <version>1.0.0</version>
          <module>
            <name>ietf-interfaces</name>
          </module>
          <!--Module list of IETF YANG modules elided -->
        </package>
        <package>
          <name>oc-schema</name>
          <version>1.0.0</version>
          <module>
            <name>openconfig-interfaces</name>
          </module>
          <!--Module list of OpenConfig YANG modules elided -->
        </package>
      </packages>
      <schema-set-selection>
        <schema-set>
          <name>combined-schema</name>
          <datastore>
            <name>ds:running</name>
            <package>
              <name>vendor-schema</name>
              <version>1.0.0</version>
            </package>
```



```
            <package>
              <name>ietf-schema</name>
              <version>1.0.0</version>
            </package>
            <package>
              <name>oc-schema</name>
              <version>1.0.0</version>
            </package>
          </datastore>
          <datastore>
            <name>ds:operational</name>
            <package>
              <name>vendor-schema</name>
              <version>1.0.0</version>
```

```
        </package>
        <package>
          <name>ietf-schema</name>
          <version>1.0.0</version>
        </package>
        <package>
          <name>oc-schema</name>
          <version>1.0.0</version>
        </package>
      </datastore>
    </schema-set>
    <schema-set>
      <name>vendor-schema</name>
      <partial/>
      <datastore>
        <name>ds:running</name>
        <package>
          <name>vendor-schema</name>
          <version>1.0.0</version>
        </package>
      </datastore>
      <datastore>
        <name>ds:operational</name>
        <package>
          <name>vendor-schema</name>
          <version>1.0.0</version>
        </package>
      </datastore>
      <selectable-with>combined-schema</selectable-with>
      <selectable-with>ietf-schema</selectable-with>
      <selectable-with>oc-schema</selectable-with>
      <custom-selectable>
        <combinable-with>ietf-schema</combinable-with>
        <combinable-with>oc-schema</combinable-with>
```



```
      </custom-selectable>
    </schema-set>
    <schema-set>
      <name>ietf-schema</name>
      <partial/>
      <datastore>
        <name>ds:running</name>
```

```
        <package>
          <name>ietf-schema</name>
          <version>1.0.0</version>
        </package>
      </datastore>
      <datastore>
        <name>ds:operational</name>
        <package>
          <name>ietf-schema</name>
          <version>1.0.0</version>
        </package>
      </datastore>
      <selectable-with>combined-schema</selectable-with>
      <selectable-with>vendor-schema</selectable-with>
      <selectable-with>oc-schema</selectable-with>
      <custom-selectable>
        <combinable-with>vendor-schema</combinable-with>
      </custom-selectable>
    </schema-set>
    <schema-set>
      <name>oc-schema</name>
      <partial/>
      <datastore>
        <name>ds:running</name>
        <package>
          <name>oc-schema</name>
          <version>1.0.0</version>
        </package>
      </datastore>
      <datastore>
        <name>ds:operational</name>
        <package>
          <name>oc-schema</name>
          <version>1.0.0</version>
        </package>
      </datastore>
      <selectable-with>combined-schema</selectable-with>
      <selectable-with>vendor-schema</selectable-with>
      <selectable-with>ietf-schema</selectable-with>
      <custom-selectable>
        <combinable-with>vendor-schema</combinable-with>
```

```
          </custom-selectable>
        </schema-set>
      </schema-set-selection>
    </data>
  </rpc-reply>
```

The clients may configure the device in three different ways.

A.2.1. Choosing a single schema family

A client that wishes to use a single schema family for all interactions with the device can choose a single schema-set and configure it as the default schema-set:

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <schema-set-selection>
    <selectable>oc-schema</selectable>
    <default>oc-schema</default>
  </schema-set-selection>
</config>
```

A.2.2. Restricting some sessions to particular schema family

If a client wishes to use multiple schema families for configuration, but restrict some sessions to a particular schema family, then they may configure the default schema as "combined-schema", but also 'oc-schema' that can be selected via client sessions as a named schema-set.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <schema-set-selection>
    <selectable>combined-schema</selectable>
    <selectable>oc-schema</selectable>
    <default>combined-schema</default>
  </schema-set-selection>
</config>
```

A.2.3. Custom combinable schema-set

If there is a need for the client to use IETF or OC schema alongside the vendor schema, then this can be achieved by configuring a custom schema-set. Two custom schema-sets can be configured, either "vendor + ietf schema", or "vendor + oc schema". The example below defines and selects a custom schema-set that combines the vendor and OC schema-sets.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <schema-set-selection>
    <custom>
      <name>my-custom-schema</name>
      <description>Vendor and OC schema-sets only</description>
      <included-schema>vendor-schema</included-schema>
      <included-schema>oc-schema</included-schema>
    </custom>
    <selectable>my-custom-schema</selectable>
    <default>my-custom-schema</default>
  </schema-set-selection>
</config>
```

Note, for the last case, rather than requiring the client to configure custom schema, the device could predefine "vendor + ietf" and "vendor + oc" as named schema-sets available for selection.

Authors' Addresses

Robert Wilton
Cisco Systems, Inc.

Email: rwilton@cisco.com

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Joe Clarke
Cisco Systems, Inc.

Email: jclarke@cisco.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Bo Wu
Huawei

Email: lana.wubo@huawei.com