Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 20, 2016

M. Bjorklund
Tail-f Systems

December 18, 2015

YANG Structural Mount
draft-bjorklund-netmod-structural-mount-00

Abstract

This document defines a mechanism to combine YANG modules into the schema defined in other YANG modules.

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 June 20, 2016.

Table of Contents

1. Introduction

1.1. Terminology

The keywords "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].

1.1.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration data (read-write) and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Background

YANG has two mechanisms for extending a data model with additional nodes; "uses" and "augment". The "uses" statement explicitly incorporates the contents of a "grouping" defined in some other module. The "augment" statement explicitly adds contents to a target node defined in some other module. In both these cases, the source and/or target model explicitly defines the relationship between the models.

In some cases these mechanisms are not sufficient. For example, suppose we have a model like ietf-interfaces [RFC7223] that is defined to be implemented in a device. Now suppose we want to model a device that supports multiple logical devices [I-D.rtgyangdt-rtgwg-device-model], where each such logical device has its own instantiation of ietf-interfaces (and other models), but at the same time, we'd like to be able to manage all these logical devices from the main device. We would like something like this:

```
  +--rw interfaces
  | +--rw interface* [name]
  |    ...
  +--rw logical-device* [name]
     +--rw name             string
     |   ...
     +--rw interfaces
       +--rw interface* [name]
          ...
```

With the "uses" approach, ietf-interfaces would have to define a grouping with all its nodes, and the new model for logical devices would have to use this grouping. This is a not a scalable solution, since every time there is a new model defined, we would have to update our model for logical devices to use a grouping from the new model. Another problem is that this approach cannot handle vendor-specific modules.

With the "augment" approach, ietf-interfaces would have to augment the logical-device list with all its nodes, and at the same time define all its nodes on the top-level. This approach is also not scalable, since there may be other models to which we would like to add the interface list.

3. Structural Mount

The structural mount mechanism defined in this document takes a different approach. It decouples the definition of the relation between the source and target models from the definitions of the models themselves.

This is accomplished with a YANG extension statement that is used to specify a mount point in a data model. The purpose of a mount point is to define a place in the node hierarchy where other YANG data models may be attached, without any special notation in the other YANG data models.

For each mount point supported by a server, the server populates an operational state node hierarchy with information about which models it has mounted. This node hierarchy can be read by a client in order to learn what is implemented on a server.

Structural mount applies to the schema, and specifically does not assume anything about how the mounted data is implemented. It may be implemented using the same instrumentation as the rest of the system, or it may be implemented by querying some other system.

This document allows mounting of complete data models only. Other specifications may extend this model by defining additional mechanisms, for example mounting of sub-hierarchies of a module.

3.1. Validation of Mounted Data

All paths (in leafrefs, instance-identifiers, XPath expressions, and target nodes of augments) in the data models mounted at a mount point are interpreted with the mount point as the root node, and the mounted data nodes as its children. This means that data within a mounted subtree can never refer to data outside of this subtree.

3.2. Top-level RPCs

If any mounted data model defines RPCs, these RPCs can be invoked by treating them as actions defined where the mount point is specified.

3.3. Top-level Notifications

If the server emits a notification defined at the top-level in any mounted data model, it is treated as if the notification was attached to the data node where the mount point is specified.

4. Data Model

This document defines the YANG module "ietf-yang-structural-mount", which has the following structure:

```
  module: ietf-yang-structural-mount
     +--ro mount-points
        +--ro mount-point* [module name]
           +--ro module                yang:yang-identifier
           +--ro name                  yang:yang-identifier
           +--ro (data-model)?
              +--:(inline-yang-library)
              |  +--ro inline-yang-library?   empty
              +--:(modules)
                 +--ro modules
                    +--ro module* [name revision]
                       +--ro name           yang:yang-identifier
                       +--ro revision       union
                       +--ro schema?        inet:uri
                       +--ro namespace      inet:uri
                       +--ro feature*       yang:yang-identifier
                       +--ro deviation* [name revision]
                       |  +--ro name        yang:yang-identifier
                       |  +--ro revision    union
                       +--ro conformance    enumeration
                       +--ro submodules
                          +--ro submodule* [name revision]
                             +--ro name        yang:yang-identifier
                             +--ro revision    union
                             +--ro schema?     inet:uri
```

5. Structural Mount YANG Module

<CODE BEGINS> file "ietf-yang-structural-mount@2015-12-17.yang"

```
module ietf-yang-structural-mount {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-structural-mount";
  prefix yangmnt;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-yang-library {
    prefix yanglib;
  }

  organization
```

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

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

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

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

       WG Chair: Kent Watsen
                 <mailto:kwatsen@juniper.net>

       Editor:   Martin Bjorklund
                 <mailto:mbj@tail-f.com>";


    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.
    description
      "This module defines a YANG extension statement that can be used
       to incorporate data models defined in other YANG modules in a
       module.  It also defines a operational state data so that
       clients can learn which data models a server implements for the
       mount points.

       Copyright (c) 2015 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).

       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).

       This version of this YANG module is part of RFC XXXX
       (http://tools.ietf.org/html/rfcXXXX); see the RFC itself for
       full legal notices.";

    // RFC Ed.: update the date below with the date of RFC publication
```

```
    // and remove this note.
    revision 2015-12-17 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: YANG Structural Mount";
    }

    /*
     * Extension statements
     */

    extension mount-point {
      argument name;
      description
        "The argument 'name' is a yang-identifier.  The name of
         the mount point MUST be unique within the module where it
         is defined.

         The 'mount-point' statement can be present in 'container' and
         'list'.

         A mount point defines a place in the node hierarchy where other
         data models may be attached.  A server that implements a module
         with a mount point, populates the /mount-points/mount-point
         list with detailed information on which data models are mounted
         at each mount point.";

    }

    /*
     * Operational state data nodes
     */

    container mount-points {
      config false;
      description
        "Contains information about which mount points are implemented
         in the server, and their data models.";

      list mount-point {
        key "module name";
        description
          "Contains information about which data models are implemented
           for the mountpoint 'name' defined in 'module'.";

        leaf module {
          type yang:yang-identifier;
```

```
          description
            "The name of the module where the mount point is defined.";
        }
        leaf name {
          type yang:yang-identifier;
          description
            "The name of the mount point.";
        }
        choice data-model {
          mandatory true;
          description
            "Indicates which data models the server implements
             for this mount point.

             It is expected that this choice may be augmented with new
             data model discovery mechansisms.";

          leaf inline-yang-library {
            type empty;
            description
              "This leaf indicates that the server has mounted
               'ietf-yang-library' at the mount point, and that this
               instantiation of 'ietf-yang-library' contains the
               information about which modules are mounted.

               This is useful if the mount point is defined in a
               list, and different list entries may mount a different
               set of modules.";
          }

          container modules {
            description
              "The 'module' list contains the set of modules that are
               mounted at the mount point.";

            uses yanglib:module-list;
          }
        }
      }
    }
  }
```

<CODE ENDS>

6. Usage Example

A data model for logical devices may be defined as:

```
module example-logical-devices {
  namespace "http://example.com/logical-device";
  prefix exld;

  import ietf-yang-structural-mount {
    prefix yangmnt;
  }

  container logical-devices {
    list logical-device {
      key name;
      leaf name {
        type string;
      }

      yangmnt:mount-point logical-device;
    }
  }
}
```

A server that implements two logical devices might populate the "mount-points" container with:

```
<mount-points
  xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-structural-mount">
  <mount-point>
    <module>example-logical-devices</module>
    <name>device-root</name>
    <modules>
      <module>
        <name>ietf-interface</name>
        <revision>2014-05-08</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-interfaces
        </namespace>
        <conformance>implement</conformance>
      </module>
      <module>
        <name>ietf-yang-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-yang-types
        </namespace>
        <conformance>import</conformance>
      </module>
    </modules>
  </mount-point>
</mount-points>
```

and the "logical-devices" container might have:

```
<logical-devices xmlns="http://example.com/logical-device">
  <logical-device>
    <name>vrtrA</name>
    <interfaces
      xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        ...
      </interface>
    </interfaces>
  </logical-device>
  <logical-device>
    <name>vrtrB</name>
    <interfaces
      xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        ...
      </interface>
    </interfaces>
  </logical-device>
</logical-devices>
```

7. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-structural-mount

Registrant Contact: The IESG.

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

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

```
name:         ietf-yang-structural-mount
namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-structural-mount
prefix:       yangmnt
reference:    RFC XXXX
```

8. Security Considerations

TBD

9. Acknowledgements

The author would like to thank Lou Berger, Alexander Clemm, Ladislav Lhotka, and Jan Medved for useful comments and discussions on this topic.

10. References

10.1. Normative References

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

10.2. Informative References

[I-D.rtgyangdt-rtgwg-device-model] Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Organizational Model", draft-rtgyangdt-rtgwg-device-model-01 (work in progress), September 2015.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

Author's Address

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com