Network Working Group
Internet-Draft
Obsoletes: rfc7895 (if approved)
Updates: rfc7950, rfc8040 (if approved)
Intended status: Standards Track
Expires: January 4, 2018

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
K. Watsen
Juniper Networks

July 3, 2017

YANG Library
draft-nmdsdt-netconf-rfc7895bis-01

Abstract

This document describes a YANG library that provides information about all the YANG modules used by a network management server (e.g., a Network Configuration Protocol (NETCONF) server). Simple caching mechanisms are provided to allow clients to minimize retrieval of this information.

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 4, 2018.

Copyright Notice

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
1.1. Terminology 3
1.2. Tree Diagrams 4
1.3. Motivation for rfc7895bis 4
1.4. Summary of Changes from RFC 7895 5
1.5. Summary of Updates to RFC 7950 6
1.6. Summary of Updates to RFC 8040 6
1.7. Open Issues 6
2. YANG Library 7
2.1. yang-library 9
2.1.1. yang-library/modules/module 9
2.1.2. yang-library/module-sets/module-set 9
2.1.3. yang-library/datastores/datastore 9
2.2. modules-state 9
2.2.1. modules-state/module-set-id 9
2.2.2. modules-state/module 10
2.3. YANG Library Module 10
3. IANA Considerations 20
3.1. YANG Module Registry 20
4. Security Considerations 21
5. Acknowledgements 21
6. References 21
6.1. Normative References 21
6.2. Informative References 23
Authors' Addresses 23

1. Introduction

There is a need for standard mechanisms to provide the operational state of the server. This includes, for instance, identifying the YANG modules and datastores that are in use by a server and how they relate to each other.

If a large number of YANG modules are utilized by the server, then the YANG library contents needed can be relatively large. This information changes very infrequently, so it is important that clients be able to cache the YANG library contents and easily

identify whether their cache is out of date.

YANG library information can be different on every server and can change at runtime or across a server reboot.

If the server implements multiple protocols to access the YANG-defined data, each such protocol has its own conceptual instantiation of the YANG library.

The following information is needed by a client application (for each YANG module in the library) to fully utilize the YANG data modeling language:

- identifier: a unique identifier for the module that includes the module's name, revision, features, and deviations.
- name: The name of the YANG module.
- revision: Each YANG module and submodule within the library has a revision. This is derived from the most recent revision statement within the module or submodule. If no such revision statement exists, the module's or submodule's revision is the zero-length string.
- submodule list: The name and revision of each submodule used by the module MUST be identified.
- feature list: The name of each YANG feature supported by the server, in a given context, MUST be identified.
- deviation list: The name of each YANG module used for deviation statements, in a given context, MUST be identified.

The following information is needed by a client application (for each datastore supported by the server) to fully access all the YANG-modelled data available on the server:

- identity: the YANG identity for the datastore.
- properties: properties supported by the datastore.

- modules: modules supported by the datastore, including any features and deviations.

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].

The following terms are defined in [RFC6241]:

- client
- server

The following terms are defined in [RFC7950]:

- module
- submodule

The following terms are used within this document:

- YANG library: A collection of metadata describing the server's operational state.

1.2. 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.

1.3. Motivation for rfc7895bis

RFC Ed.: delete this section, including this note, at time of publication.

All NETCONF servers supporting YANG 1.1 [RFC7950] MUST support YANG Library (see Section 5.6.4 of RFC 7950). Similarly, all RESTCONF servers MUST support YANG Library (see Section 10 of RFC 8040). These requirements are independent of if the server supports NMDA or not.

RFC 7895 has a mandatory to implement 'modules-state' tree that a server uses to advertise all the modules it supports. However, this

module was designed assuming the all modules would be in all datastores, and with the same number of features and deviations. However, this is not the case with NMDA-compatible servers that may have some modules that only appear in <operational> (e.g., ietf-network-topo) or only also appear in a dynamic datastore (e.g., i2rs-ephemeral-rib). It is also possible that a server only implements a module in <running>, at it hasn't yet coded support for returning the module's opstate yet. Presumably, an NMDA-supporting server would return all modules implemented in every datastore, but this would be misleading to existing clients and unhelpful to NMDA-aware clients.

In the end, it appears that the 'modules-state' node should be for non-NMDA aware clients. For backwards compatability, an NMDA-supporting server SHOULD populate 'modules-state' in a backwards-compatible manner. The new 'yang-library' node would be ignored by legacy clients, while providing all the data needed for NMDA-aware clients, which would themselves ignore the 'modules-state' tree.

In addition to resolving the 'modules-state' node NMDA-incompatibility issue described above, the solution presented in this document is further motivated by the following desires:

- leverage Section 5.6.4 of RFC 7950 and Section 10 of RFC 8040.
- indicate which modules are supported by each datastore
- enable the features and deviations to vary by datastore
- structure extensible to support schema-mount
- provide a top-level container for all server metadata

1.4. Summary of Changes from RFC 7895

This document updates [RFC7895] in the following ways:

- Renames document title from "YANG Module Library" to "YANG Library".
- Adds new top-level "yang-library" container to hold many types of server metadata: modules supported, datastores supported, relationships between datastores and modules, etc.
- Deprecates the modules-state tree.

1.5. Summary of Updates to RFC 7950

This document updates [RFC7950] in the following ways:

1. Section 5.6.4 says:

   A NETCONF server MUST announce the modules it implements (see Section 5.6.5) by implementing the YANG module "ietf-yang-library" defined in [RFC7895] and listing all implemented modules in the "/modules-state/module" list.

This should be updated to allow for also listing all implemented modules in the "/yang-library/modules/module" list or, more generally, use the entire YANG Library.

2. Section 5.6.4 also says:

> The parameter "module-set-id" has the same value as the leaf "/modules-state/module-set-id" from "ietf-yang-library". This parameter MUST be present.

This should be updated to say that, for NMDA-capable servers, this parameter has the same value as the leaf "/yang-library/module-sets/module-set/id", for the module-set that is used by <running>.

1.6. Summary of Updates to RFC 8040

This document updates [RFC8040] in the following ways:

1. Section 10.1 says that the "modules-state/module" list is mandatory. This should be updated to allow for also listing all supported modules in the "/yang-library/modules/module" list or, more generally, use the entire YANG Library.

1.7. Open Issues

- The per-datastore 'properties' idea is still being discussed. It's included here so as to provide something to point at.

- There's debate if there should be a list of module-sets or if instead each 'module-set' should be embeded into the datastore definition. This discussion goes into if a datastore can reference more than one module-set.

2. YANG Library

The "ietf-yang-library" module provides information about the modules used by a server. This module is defined using YANG version 1, but it supports the description of YANG modules written in any revision of YANG.

Following is the YANG Tree Diagram for the "ietf-yang-library"

module, including the deprecated 'modules-state' tree:


```
    +--ro yang-library
```

```
|  +--ro modules
|  |  +--ro module* [id]
|  |     +--ro id                  string
|  |     +--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-type    enumeration
|  |     +--ro submodule* [name revision]
|  |        +--ro name         yang:yang-identifier
|  |        +--ro revision     union
|  |        +--ro schema?      inet:uri
|  +--ro module-sets
|  |  +--ro module-set*
|  |     +--ro id?       string
|  |     +--ro module*   -> /yang-library/modules/module/id
|  +--ro datastores
|     +--ro datastore* [name]
|        +--ro name           identityref
|        +--ro properties
|        |  +--ro property*   identityref
|        +--ro module-set?   -> /yang-library/module-sets/module-set/id
x--ro modules-state
   +--ro module-set-id    string
   +--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-type    enumeration
      +--ro submodule* [name revision]
         +--ro name         yang:yang-identifier
         +--ro revision     union
         +--ro schema?      inet:uri
```

2.1. yang-library

This mandatory container holds all of the server's metadata.

2.1.1. yang-library/modules/module

This mandatory list contains one entry for each unique instance of a module in use by the server. Each entry is distiguished by the module's name, revisions, features, and deviations.

2.1.2. yang-library/module-sets/module-set

This mandatory list contains one entry for each module-set in use by the server (e.g., presented by a datastore).

2.1.3. yang-library/datastores/datastore

This mandatory list contains one entry for each datastore supported by the server. Each datastore entry both identifies any special propoerties it has and any module-sets it uses.

2.2. modules-state

This mandatory container holds the identifiers for the YANG data model modules supported by the server.

2.2.1. modules-state/module-set-id

This mandatory leaf contains a unique implementation-specific identifier representing the current set of modules and submodules on a specific server. The value of this leaf MUST change whenever the set of modules and submodules in the YANG library changes. There is no requirement that the same set always results in the same "module-set-id" value.

This leaf allows a client to fetch the module list once, cache it, and only refetch it if the value of this leaf has been changed.

If the value of this leaf changes, the server also generates a "yang-library-change" notification, with the new value of "module-set-id".

Note that for a NETCONF server that implements YANG 1.1 [RFC7950], a change of the "module-set-id" value results in a new value for the :yang-library capability defined in [RFC7950]. Thus, if such a server implements NETCONF notifications [RFC5277], and the

notification "netconf-capability-change" [RFC6470], a

"netconf-capability-change" notification is generated whenever the "module-set-id" changes.

2.2.2. modules-state/module

This mandatory list contains one entry for each YANG data model module supported by the server. There MUST be an entry in this list for each revision of each YANG module that is used by the server. It is possible for multiple revisions of the same module to be imported, in addition to an entry for the revision that is implemented by the server.

2.3. YANG Library Module

The "ietf-yang-library" module defines monitoring information for the YANG modules used by a server.

The modules "ietf-yang-types" and "ietf-inet-types" from [RFC6991] and the module "ietf-datastores" from [I-D.ietf-netmod-revised-datastores] are used by this module for some type definitions.

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-yang-library@2017-07-03.yang"

```
 module ietf-yang-library {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-yang-library";
   prefix "yanglib";

   import ietf-yang-types {
     prefix yang;
     reference "RFC 6991: Common YANG Data Types.";
   }
   import ietf-inet-types {
     prefix inet;
     reference "RFC 6991: Common YANG Data Types.";
```

```
  }
  import ietf-datastores {
    prefix ds;
    reference "I-D.ietf-revised-datastoes:
               Network Management Datastore Architecture.";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
```



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

     Author:   Andy Bierman
               <mailto:andy@yumaworks.com>

     Author:   Martin Bjorklund
               <mailto:mbj@tail-f.com>

     Author:   Kent Watsen
               <mailto:kwatsen@juniper.net>";

  description
    "This module contains monitoring information about the YANG
     modules and submodules that are used within a YANG-based
     server.

     Copyright (c) 2016 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.";

  // 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 2017-07-03 {
    description
      "Updated revision.";
    reference
      "RFC XXXX: YANG Library.";
  }
  revision 2016-04-09 {
    description
      "Initial revision.";
    reference
      "RFC 7895: YANG Module Library.";
  }
```



```
  /*
   * Typedefs
   */

  typedef revision-identifier {
    type string {
      pattern '\d{4}-\d{2}-\d{2}';
    }
    description
      "Represents a specific date in YYYY-MM-DD format.";
  }


  /*
   * Groupings
   */
  grouping common-leafs2 {
    description
      "Common parameters for YANG modules and submodules.";

    leaf name {
      type yang:yang-identifier;
      description
        "The YANG module or submodule name.";
    }
```

```
      leaf revision {
        type union {
          type revision-identifier;
          type string { length 0; }
        }
        description
          "The YANG module or submodule revision date.
           A zero-length string is used if no revision statement
           is present in the YANG module or submodule.";
      }
    }

    grouping schema-leaf2 {
      description
        "Common schema leaf parameter for modules and submodules.";

      leaf schema {
        type inet:uri;
        description
          "Contains a URL that represents the YANG schema
           resource for this module or submodule.

           This leaf will only be present if there is a URL
```



```
           available for retrieval of the schema for this entry.";
      }
    }


    /*
     * Top-level container
     */
    container yang-library {
      config false;
      description
        "Top-level resource providing all the meta information the
         server possesses.";

      container modules {
        description
          "A container holding a list of modules.  Note, modules being
           listed here does not mean that they are supported by any
```

```
          particular datastore.";

        list module {
          key "id";

          description
            "Each entry represents one revision of one module
             currently supported by the server.";

          leaf id {
            type string;
            description
              "A system-generated value that uniquely represents the
               module listing, including its name, revision, features,
               and deviations.";
          }

          uses common-leafs2;
          uses schema-leaf2;

          leaf namespace {
            type inet:uri;
            mandatory true;
            description
              "The XML namespace identifier for this module.";
          }
          leaf-list feature {
            type yang:yang-identifier;
            description
              "List of YANG feature names from this module that are
```



```
               supported by the server, regardless whether they are
               defined in the module or any included submodule.";
          }
          list deviation {
            key "name revision";
            description
              "List of YANG deviation module names and revisions
               used by this server to modify the conformance of
               the module associated with this entry.  Note that
               the same module can be used for deviations for
               multiple modules, so the same entry MAY appear
```

```
            within multiple 'module' entries.

            The deviation module MUST be present in the 'module'
            list, with the same name and revision values.
            The 'conformance-type' value will be 'implement' for
            the deviation module.";
        uses common-leafs2;
      }
      leaf conformance-type {
        type enumeration {
          enum implement {
            description
              "Indicates that the server implements one or more
               protocol-accessible objects defined in the YANG module
               identified in this entry.  This includes deviation
               statements defined in the module.

               For YANG version 1.1 modules, there is at most one
               module entry with conformance type 'implement' for a
               particular module name, since YANG 1.1 requires that
               at most one revision of a module is implemented.

               For YANG version 1 modules, there SHOULD NOT be more
               than one module entry for a particular module name.";
          }
          enum import {
            description
              "Indicates that the server imports reusable definitions
               from the specified revision of the module, but does
               not implement any protocol accessible objects from
               this revision.

               Multiple module entries for the same module name MAY
               exist. This can occur if multiple modules import the
               same module, but specify different revision-dates in
               the import statements.";
          }
```



```
        }
        mandatory true;
        description
          "Indicates the type of conformance the server is claiming
```

```
            for the YANG module identified by this entry.";
        }
        list submodule {
          key "name revision";
          description
            "Each entry represents one submodule within the
             parent module.";
          uses common-leafs2;
          uses schema-leaf2;
        }
      }
    }

    container module-sets {
      description
        "A container for a list of module-sets.  Module-sets being
         listed here does not mean that they are used by any
         particular datastore.";
      list module-set {
        description
          "An arbitrary module-set definition provided by the server.";

        leaf id {
          type string;
          description
            "A server-supplied identifier for this set of modules.";
        }
        leaf-list module {
          type leafref {
            path "/yang-library/modules/module/id";
          }
          description
            "A module-instance supported by the server, including its
             features and deviations.";
        }
      }
    }

    container datastores {
      description
        "A container for a list of datastores supported by the server.
         Each datastore indicates which module-sets it supports.";

      list datastore {
```

```
        key name;
        leaf name {
          type identityref {
            base ds:datastore;
          }
          description
            "The identity of the datastore.";
        }
        container properties {
          leaf-list property {
            type identityref {
              base ds:property;
            }
            description
              "A property of the datastore.";
          }
          description
            "A list of properties supported by this datastore.";
        }
        leaf module-set {
          type leafref {
            path "/yang-library/module-sets/module-set/id";
          }
          description
            "A reference to a module-set supported by this datastore";
        }
        description
          "A datastore supported by this server.";
      }
    } // end 'datastores'

  } // end 'yang-library'



  /*
   * Legacy groupings
   */

  grouping module-list {
    description
      "The module data structure is represented as a grouping
       so it can be reused in configuration or another monitoring
       data structure.";

    grouping common-leafs {
      description
```

```
        "Common parameters for YANG modules and submodules.";
```



```
      leaf name {
        type yang:yang-identifier;
        description
          "The YANG module or submodule name.";
      }
      leaf revision {
        type union {
          type revision-identifier;
          type string { length 0; }
        }
        description
          "The YANG module or submodule revision date.
           A zero-length string is used if no revision statement
           is present in the YANG module or submodule.";
      }
    }

    grouping schema-leaf {
      description
        "Common schema leaf parameter for modules and submodules.";

      leaf schema {
        type inet:uri;
        description
          "Contains a URL that represents the YANG schema
           resource for this module or submodule.

           This leaf will only be present if there is a URL
           available for retrieval of the schema for this entry.";
      }
    }

    list module {
      key "name revision";
      description
        "Each entry represents one revision of one module
         currently supported by the server.";

      uses common-leafs;
      uses schema-leaf;
```

```
        leaf namespace {
          type inet:uri;
          mandatory true;
          description
            "The XML namespace identifier for this module.";
        }
        leaf-list feature {
```



```
          type yang:yang-identifier;
          description
            "List of YANG feature names from this module that are
             supported by the server, regardless whether they are
             defined in the module or any included submodule.";
        }
        list deviation {
          key "name revision";
          description
            "List of YANG deviation module names and revisions
             used by this server to modify the conformance of
             the module associated with this entry.  Note that
             the same module can be used for deviations for
             multiple modules, so the same entry MAY appear
             within multiple 'module' entries.

             The deviation module MUST be present in the 'module'
             list, with the same name and revision values.
             The 'conformance-type' value will be 'implement' for
             the deviation module.";
          uses common-leafs;
        }
        leaf conformance-type {
          type enumeration {
            enum implement {
              description
                "Indicates that the server implements one or more
                 protocol-accessible objects defined in the YANG module
                 identified in this entry.  This includes deviation
                 statements defined in the module.

                 For YANG version 1.1 modules, there is at most one
                 module entry with conformance type 'implement' for a
```

```
                particular module name, since YANG 1.1 requires that
                at most one revision of a module is implemented.

                For YANG version 1 modules, there SHOULD NOT be more
                than one module entry for a particular module name.";
            }
            enum import {
              description
                "Indicates that the server imports reusable definitions
                 from the specified revision of the module, but does
                 not implement any protocol accessible objects from
                 this revision.

                 Multiple module entries for the same module name MAY
                 exist. This can occur if multiple modules import the
```



```
                 same module, but specify different revision-dates in
                 the import statements.";
            }
          }
          mandatory true;
          description
            "Indicates the type of conformance the server is claiming
             for the YANG module identified by this entry.";
        }
        list submodule {
          key "name revision";
          description
            "Each entry represents one submodule within the
             parent module.";
          uses common-leafs;
          uses schema-leaf;
        }
      }
    }

    /*
     * Legacy operational state data nodes
     */

    container modules-state {
      config false;
```

```
    status deprecated;
    description
      "Contains YANG module monitoring information.";

    leaf module-set-id {
      type string;
      mandatory true;
      description
        "Contains a server-specific identifier representing
         the current set of modules and submodules.  The
         server MUST change the value of this leaf if the
         information represented by the 'module' list instances
         has changed.";
    }

    uses module-list;
  }

  /*
   * Notifications
   */
```



```
  notification yang-library-change {
    description
      "Generated when the set of modules and submodules supported
       by the server has changed.";
    leaf module-set-id {
      type leafref {
        path "/yanglib:modules-state/yanglib:module-set-id";
      }
      mandatory true;
      description
        "Contains the module-set-id value representing the
         set of modules and submodules supported at the server at
         the time the notification is generated.";
    }
  }

}
```

<CODE ENDS>

3. IANA Considerations

3.1. YANG Module Registry

RFC 7895 previously registered one URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration was made:

```
URI: urn:ietf:params:xml:ns:yang:ietf-yang-library
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

This document takes over this registration entry made by RFC 7895.

RFC 7895 previously registered one YANG module in the "YANG Module Names" registry [RFC6020] as follows:

```
name:         ietf-yang-library
namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-library
prefix:       yanglib
reference:    RFC 7895
```

This document takes over this registration entry made by RFC 7895.

4. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus

important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /modules-state/module: The module list used in a server implementation may help an attacker identify the server capabilities and server implementations with known bugs. Although some of this information may be available to all users via the NETCONF <hello> message (or similar messages in other management protocols), this YANG module potentially exposes additional details that could be of some assistance to an attacker. Server vulnerabilities may be specific to particular modules, module revisions, module features, or even module deviations. This information is included in each module entry. For example, if a particular operation on a particular data node is known to cause a server to crash or significantly degrade device performance, then the module list information will help an attacker identify server implementations with such a defect, in order to launch a denial-of-service attack on the device.

5. Acknowledgements

Contributions to this material by Andy Bierman are based upon work supported by the The Space & Terrestrial Communications Directorate (S&TCD) under Contract No. W15P7T-13-C-A616. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of The Space & Terrestrial Communications Directorate (S&TCD).

6. References

6.1. Normative References

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-03 (work in progress), July 2017.

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

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

[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>.

[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>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://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, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <http://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <http://www.rfc-editor.org/info/rfc8040>.

6.2. Informative References

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <http://www.rfc-editor.org/info/rfc5277>.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <http://www.rfc-editor.org/info/rfc6470>.

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net