Internet Engineering Task Force (IETF)
Request for Comments: 8525
Obsoletes: 7895
Category: Standards Track
ISSN: 2070-1721

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University
K. Watsen
Watsen Networks
R. Wilton
Cisco Systems

March 2019

YANG Library

Abstract

This document describes a YANG library that provides information about the YANG modules, datastores, and datastore schemas used by a network management server. Simple caching mechanisms are provided to allow clients to minimize retrieval of this information. This version of the YANG library supports the Network Management Datastore Architecture (NMDA) by listing all datastores supported by a network management server and the schema that is used by each of these datastores.

This document obsoletes RFC 7895.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8525.

Table of Contents

1. Introduction

There is a need for a standard mechanism to expose which YANG modules [RFC7950], datastores [RFC8342], and datastore schemas [RFC8342] are in use by a network management server.

This document defines the YANG module "ietf-yang-library" that provides this information. This version of the YANG library is compatible with the Network Management Datastore Architecture (NMDA) [RFC8342]. The previous version of the YANG library, defined in [RFC7895], is not compatible with the NMDA since it assumes that all datastores have exactly the same schema. This is not necessarily true in the NMDA since dynamic configuration datastores may have their own datastore schema. Furthermore, the operational state datastore may support non-configurable YANG modules in addition to the YANG modules supported by conventional configuration datastores.

The old YANG library definitions have been retained (for backwards-compatibility reasons), but the definitions have been marked as deprecated. For backwards compatibility, an NMDA-supporting server SHOULD populate the deprecated "/modules-state" tree in a backwards-compatible manner. The new "/yang-library" tree will be ignored by legacy clients but will provide all the data needed for NMDA-aware clients (which will ignore the "/modules-state" tree). The recommended approach to populate "/modules-state" is to report the YANG modules with "config true" data nodes that are configurable via conventional configuration datastores and the YANG modules with "config false" data nodes that are returned via a Network Configuration Protocol (NETCONF) <get> operation or equivalent.

The YANG library information can be different on every server, and it can change at runtime or across a server reboot. If a server implements multiple network management protocols to access the server’s datastores, then each such protocol may have its own conceptual instantiation of the YANG library.

If a large number of YANG modules are utilized by a server, then the YANG library contents can be relatively large. Since the YANG library contents change very infrequently, it is important that clients be able to cache the YANG library contents and easily identify whether their cache is out of date.

1.1. Terminology

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.

The following terms are defined in [RFC7950]:

- module
- submodule
- data node

This document uses the phrase "implement a module" as defined in Section 5.6.5 of [RFC7950].

The following terms are defined in [RFC8342]:

- datastore
- datastore schema
- configuration
- conventional configuration datastore
- operational state
- operational state datastore
- dynamic configuration datastore
- client
- server

The following terms are used within this document:

- YANG library: A collection of YANG modules, submodules, datastores, and datastore schemas used by a server.
- YANG library content identifier: A server-generated identifier of the contents of the YANG library.

Tree diagrams in this document use the notation defined in [RFC8340].

2. Objectives

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

- name: The name of the YANG module.
- revision: If defined in the YANG module or submodule, the revision is derived from the most recent revision statement within the module or submodule.
- submodule list: The name and (if defined) revision of each submodule used by the module must be identified.
- feature list: The name of each YANG feature supported by the server must be identified.
- deviation list: The name of each YANG module with deviation statements affecting a given YANG module must be identified.

In addition, the following information is needed by a client application for each datastore supported by a server:

- identity: The YANG identity for the datastore.
- schema: The schema (i.e., the set of modules) implemented by the datastore.

In order to select one out of several possible designs for the YANG library data model, the following criteria were used:

1. The information must be efficient for a client to consume. Since the size of the YANG library can be quite large, it should be possible for clients to cache the YANG library information.
2. A dynamic configuration datastore must be able to implement a module or feature that is not implemented in the conventional configuration datastores.
3. It must be possible to not implement a module or feature in <operational>, even if it is implemented in some other datastore. This is required for transition purposes; a server that wants to implement <operational> should not have to implement all modules at once.

4. A given module can only be implemented in one revision in all datastores. If a module is implemented in more than one datastore, the same revision is implemented in all these datastores.

5. Multiple revisions can be used for import, if import-by revision is used.

6. It must be possible to use the YANG library by schema mount [RFC8528].

3. YANG Library Data Model

The "ietf-yang-library" YANG module provides information about the modules, submodules, datastores, and datastore schemas supported by a server. All data nodes in the "ietf-yang-library" module are "config false" and thus only accessible in the operational state datastore.

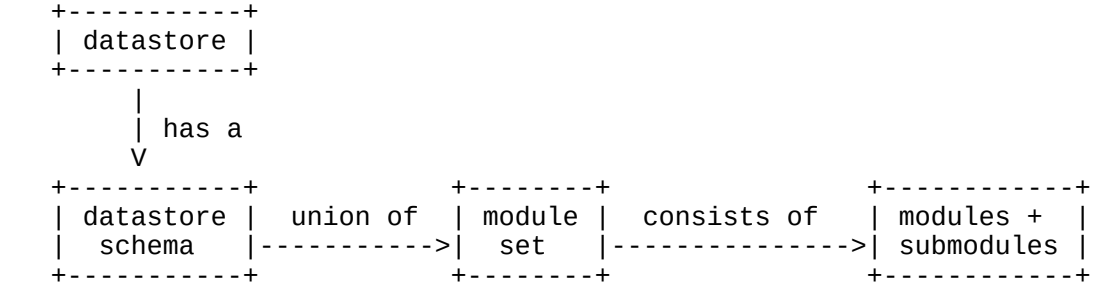

Figure 1

The conceptual model of the YANG library is depicted in Figure 1. Following the NMDA, every datastore has an associated datastore schema. A datastore schema is a union of module sets, and every module set is a collection of modules and submodules, including the modules and submodules used for imports. Note that multiple datastores may refer to the same datastore schema. Furthermore, it is possible for individual datastore schemas to share module sets. A common use case is the operational state datastore schema, which is a superset of the schema used by conventional configuration datastores.

Below is the YANG tree diagram for the "ietf-yang-library" module, excluding the deprecated "/modules-state" tree:

```
module: ietf-yang-library
  +--ro yang-library
     +--ro module-set* [name]
     |  +--ro name                  string
     |  +--ro module* [name]
     |  |  +--ro name         yang:yang-identifier
     |  |  +--ro revision?    revision-identifier
     |  |  +--ro namespace    inet:uri
     |  |  +--ro location*    inet:uri
     |  |  +--ro submodule* [name]
     |  |  |  +--ro name        yang:yang-identifier
     |  |  |  +--ro revision?   revision-identifier
     |  |  |  +--ro location*   inet:uri
     |  |  +--ro feature*     yang:yang-identifier
     |  |  +--ro deviation*   -> ../../module/name
     |  +--ro import-only-module* [name revision]
     |     +--ro name         yang:yang-identifier
     |     +--ro revision     union
     |     +--ro namespace    inet:uri
     |     +--ro location*    inet:uri
     |     +--ro submodule* [name]
     |        +--ro name        yang:yang-identifier
     |        +--ro revision?   revision-identifier
     |        +--ro location*   inet:uri
     +--ro schema* [name]
     |  +--ro name          string
     |  +--ro module-set*   -> ../../module-set/name
     +--ro datastore* [name]
     |  +--ro name      ds:datastore-ref
     |  +--ro schema    -> ../../schema/name
     +--ro content-id    string

  notifications:
    +---n yang-library-update
       +--ro content-id    -> /yang-library/content-id
```

The "/yang-library" container holds the entire YANG library. The container has the following child nodes:

- The "/yang-library/module-set" contains entries representing module sets. The list "/yang-library/module-set/module" enumerates the modules that belong to the module set. A module is listed together with its submodules (if any), a set of features, and any deviation modules. The list "/yang-library/module-set/import-only-module" lists all modules (and their submodules) used only for imports. The assignment of a module to a module set is

at the server's discretion. This revision of the YANG library attaches no semantics as to which module set a module is listed in.

- The "/yang-library/schema" list contains an entry for each datastore schema supported by the server. All conventional configuration datastores use the same "schema" list entry. A dynamic configuration datastore may use a different datastore schema from the conventional configuration datastores and hence may require a separate "schema" entry. A "schema" entry has a leaf-list of references to entries in the "module-set" list. The schema consists of the union of all modules in all referenced module sets.

- The "/yang-library/datastore" list contains one entry for each datastore supported by the server, and it identifies the datastore schema associated with a datastore via a reference to an entry in the "schema" list. Each supported conventional configuration datastore has a separate entry, pointing to the same "schema" list element.

- The "/yang-library/content-id" leaf contains the YANG library content identifier, which is an implementation-specific identifier representing the current information in the YANG library on a specific server. The value of this leaf MUST change whenever the information in the YANG library changes. There is no requirement that the same information always result in the same "content-id" value. This leaf allows a client to fetch all schema information 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-update" notification.

Note that for a NETCONF server implementing the NETCONF extensions to support the NMDA [RFC8526], a change of the YANG library content identifier results in a new value for the :yang-library:1.1 capability defined in [RFC8526]. Thus, if such a server implements NETCONF notifications [RFC5277] and the "netconf-capability-change" notification [RFC6470], a "netconf-capability-change" notification is generated whenever the YANG library content identifier changes.

4. YANG Library YANG Module

The "ietf-yang-library" YANG module imports definitions from the "ietf-yang-types" and "ietf-inet-types" modules defined in [RFC6991] and from the "ietf-datastores" module defined in [RFC8342]. While the YANG module is defined using YANG version 1.1, the YANG library supports YANG modules written in any version of YANG.

```
<CODE BEGINS> file "ietf-yang-library@2019-01-04.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
      "RFC 8342: Network Management Datastore Architecture
                 (NMDA)";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <https://datatracker.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:   Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>

     Author:   Kent Watsen
               <mailto:kent+ietf@watsen.net>

     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>";
  description
    "This module provides information about the YANG modules,
     datastores, and datastore schemas used by a network
     management server.
```

```
      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.

      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
      (https://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC 8525; see
      the RFC itself for full legal notices.";

    revision 2019-01-04 {
      description
        "Added support for multiple datastores according to the
         Network Management Datastore Architecture (NMDA).";
      reference
        "RFC 8525: 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 module-identification-leafs {
    description
      "Parameters for identifying YANG modules and submodules.";
    leaf name {
      type yang:yang-identifier;
      mandatory true;
      description
        "The YANG module or submodule name.";
    }
    leaf revision {
      type revision-identifier;
      description
        "The YANG module or submodule revision date.  If no revision
         statement is present in the YANG module or submodule, this
         leaf is not instantiated.";
    }
  }

  grouping location-leaf-list {
    description
      "Common leaf-list parameter for the locations of modules and
       submodules.";
    leaf-list location {
      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.";
    }
  }

  grouping module-implementation-parameters {
    description
      "Parameters for describing the implementation of a module.";
    leaf-list feature {
      type yang:yang-identifier;
      description
        "List of all YANG feature names from this module that are
         supported by the server, regardless whether they are defined
         in the module or any included submodule.";
    }
    leaf-list deviation {
      type leafref {
        path "../../module/name";
      }
```

```
        description
          "List of all YANG deviation modules 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.

           This reference MUST NOT (directly or indirectly)
           refer to the module being deviated.

           Robust clients may want to make sure that they handle a
           situation where a module deviates itself (directly or
           indirectly) gracefully.";
      }
    }

    grouping module-set-parameters {
      description
        "A set of parameters that describe a module set.";
      leaf name {
        type string;
        description
          "An arbitrary name of the module set.";
      }
      list module {
        key "name";
        description
          "An entry in this list represents a module implemented by the
           server, as per Section 5.6.5 of RFC 7950, with a particular
           set of supported features and deviations.";
        reference
          "RFC 7950: The YANG 1.1 Data Modeling Language";
        uses module-identification-leafs;
        leaf namespace {
          type inet:uri;
          mandatory true;
          description
            "The XML namespace identifier for this module.";
        }
        uses location-leaf-list;
        list submodule {
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";
          uses module-identification-leafs;
          uses location-leaf-list;
        }
```

```
        uses module-implementation-parameters;
      }
      list import-only-module {
        key "name revision";
        description
          "An entry in this list 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 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.";
        leaf name {
          type yang:yang-identifier;
          description
            "The YANG module name.";
        }
        leaf revision {
          type union {
            type revision-identifier;
            type string {
              length "0";
            }
          }
          description
            "The YANG module revision date.
             A zero-length string is used if no revision statement
             is present in the YANG module.";
        }
        leaf namespace {
          type inet:uri;
          mandatory true;
          description
            "The XML namespace identifier for this module.";
        }
        uses location-leaf-list;
        list submodule {
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";
          uses module-identification-leafs;
          uses location-leaf-list;
        }
      }
    }
```

```
     grouping yang-library-parameters {
       description
         "The YANG library data structure is represented as a grouping
          so it can be reused in configuration or another monitoring
          data structure.";
       list module-set {
         key "name";
         description
           "A set of modules that may be used by one or more schemas.

            A module set does not have to be referentially complete,
            i.e., it may define modules that contain import statements
            for other modules not included in the module set.";
         uses module-set-parameters;
       }
       list schema {
         key "name";
         description
           "A datastore schema that may be used by one or more
            datastores.

            The schema must be valid and referentially complete, i.e.,
            it must contain modules to satisfy all used import
            statements for all modules specified in the schema.";
         leaf name {
           type string;
           description
             "An arbitrary name of the schema.";
         }
         leaf-list module-set {
           type leafref {
             path "../../module-set/name";
           }
           description
             "A set of module-sets that are included in this schema.
              If a non-import-only module appears in multiple module
              sets, then the module revision and the associated features
              and deviations must be identical.";
         }
       }
       list datastore {
         key "name";
         description
           "A datastore supported by this server.

            Each datastore indicates which schema it supports.
```

```
           The server MUST instantiate one entry in this list per
           specific datastore it supports.
           Each datastore entry with the same datastore schema SHOULD
           reference the same schema.";
        leaf name {
          type ds:datastore-ref;
          description
            "The identity of the datastore.";
        }
        leaf schema {
          type leafref {
            path "../../schema/name";
          }
          mandatory true;
          description
            "A reference to the schema supported by this datastore.
             All non-import-only modules of the schema are implemented
             with their associated features and deviations.";
        }
      }
    }

    /*
     * Top-level container
     */

    container yang-library {
      config false;
      description
        "Container holding the entire YANG library of this server.";
      uses yang-library-parameters;
      leaf content-id {
        type string;
        mandatory true;
        description
          "A server-generated identifier of the contents of the
           '/yang-library' tree.  The server MUST change the value of
           this leaf if the information represented by the
           '/yang-library' tree, except '/yang-library/content-id', has
           changed.";
      }
    }

    /*
     * Notifications
     */

    notification yang-library-update {
```

```
      description
        "Generated when any YANG library information on the
         server has changed.";
      leaf content-id {
        type leafref {
          path "/yanglib:yang-library/yanglib:content-id";
        }
        mandatory true;
        description
          "Contains the YANG library content identifier for the updated
           YANG library at the time the notification is generated.";
      }
    }

    /*
     * Legacy groupings
     */

    grouping module-list {
      status deprecated;
      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 {
        status deprecated;
        description
          "Common parameters for YANG modules and submodules.";
        leaf name {
          type yang:yang-identifier;
          status deprecated;
          description
            "The YANG module or submodule name.";
        }
        leaf revision {
          type union {
            type revision-identifier;
            type string {
              length "0";
            }
          }
          status deprecated;
          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 {
        status deprecated;
        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";
        status deprecated;
        description
          "Each entry represents one revision of one module
           currently supported by the server.";
        uses common-leafs {
          status deprecated;
        }
        uses schema-leaf {
          status deprecated;
        }
        leaf namespace {
          type inet:uri;
          mandatory true;
          status deprecated;
          description
            "The XML namespace identifier for this module.";
        }
        leaf-list feature {
          type yang:yang-identifier;
          status deprecated;
          description
            "List of YANG feature names from this module that are
             supported by the server, regardless of whether they are
             defined in the module or any included submodule.";
        }
        list deviation {
          key "name revision";
          status deprecated;
```

```
      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 {
        status deprecated;
      }
    }
    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;
```

```
          status deprecated;
          description
            "Indicates the type of conformance the server is claiming
             for the YANG module identified by this entry.";
        }
        list submodule {
          key "name revision";
          status deprecated;
          description
            "Each entry represents one submodule within the
             parent module.";
          uses common-leafs {
            status deprecated;
          }
          uses schema-leaf {
            status deprecated;
          }
        }
      }
    }

    /*
     * 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;
        status deprecated;
        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 {
        status deprecated;
      }
    }

    /*
     * Legacy notifications
```

```
     */

    notification yang-library-change {
      status deprecated;
      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;
        status deprecated;
        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>

5. IANA Considerations

RFC 7895 previously registered one URI in the "IETF XML Registry" [RFC3688]. This document takes over this registration entry made by RFC 7895 and changes the Registrant Contact to the IESG according to Section 4 of [RFC3688].

URI: urn:ietf:params:xml:ns:yang:ietf-yang-library

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-library
namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-library
prefix:       yanglib
reference:    RFC 8525
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF 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:

The "/yang-library" subtree of the YANG library may help an attacker identify the server capabilities and server implementations with known bugs since the set of YANG modules supported by a server may reveal the kind of device and the manufacturer of the device. Although some of this information may be available to all NETCONF 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. 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.

7. References

7.1. Normative References

[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>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

[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>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

7.2. Informative References

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

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

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

[RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <https://www.rfc-editor.org/info/rfc8348>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8526] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "NETCONF Extensions to Support the Network Management Datastore Architecture", RFC 8526, DOI 10.17487/RFC8526, March 2019, <https://www.rfc-editor.org/info/rfc8526>.

[RFC8528] Bjorklund, M. and L. Lhotka, "YANG Schema Mount", RFC 8528, DOI 10.17487/RFC8528, March 2019, <https://www.rfc-editor.org/info/rfc8528>.

Appendix A. Summary of Changes from RFC 7895

This document changes [RFC7895] in the following ways:

- Renamed document title from "YANG Module Library" to "YANG Library".
- Added a new top-level "/yang-library" container to hold the entire YANG library providing information about module sets, schemas, and datastores.
- Refactored the "/modules-state" container into a new "/yang-library/module-set" list.
- Added a new "/yang-library/schema" list and a new "/yang-library/datastore" list.
- Added a set of new groupings as replacements for the deprecated groupings.
- Added a "yang-library-update" notification as a replacement for the deprecated "yang-library-change" notification.
- Deprecated the "/modules-state" tree.
- Deprecated the "/module-list" grouping.
- Deprecated the "/yang-library-change" notification.

Appendix B. Example YANG Library Instance for a Basic Server

The following example shows the YANG library of a basic server implementing the "ietf-interfaces" [RFC8343] and "ietf-ip" [RFC8344] modules in the <running>, <startup>, and <operational> datastores and the "ietf-hardware" [RFC8348] module in the <operational> datastore.

Newline characters in leaf values are added for formatting reasons.

```
<yang-library
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">

  <module-set>
    <name>config-modules</name>
    <module>
      <name>ietf-interfaces</name>
      <revision>2018-02-20</revision>
      <namespace>
```

```
          urn:ietf:params:xml:ns:yang:ietf-interfaces
        </namespace>
      </module>
      <module>
        <name>ietf-ip</name>
        <revision>2018-02-22</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-ip
        </namespace>
      </module>
      <import-only-module>
        <name>ietf-yang-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-yang-types
        </namespace>
      </import-only-module>
      <import-only-module>
        <name>ietf-inet-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-inet-types
        </namespace>
      </import-only-module>
    </module-set>

    <module-set>
      <name>state-modules</name>
      <module>
        <name>ietf-hardware</name>
        <revision>2018-03-13</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-hardware
        </namespace>
      </module>
      <import-only-module>
        <name>ietf-inet-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-inet-types
        </namespace>
      </import-only-module>
      <import-only-module>
        <name>ietf-yang-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-yang-types
        </namespace>
```

```
        </import-only-module>
        <import-only-module>
          <name>iana-hardware</name>
          <revision>2018-03-13</revision>
          <namespace>
            urn:ietf:params:xml:ns:yang:iana-hardware
          </namespace>
        </import-only-module>
      </module-set>

      <schema>
        <name>config-schema</name>
        <module-set>config-modules</module-set>
      </schema>
      <schema>
        <name>state-schema</name>
        <module-set>config-modules</module-set>
        <module-set>state-modules</module-set>
      </schema>

      <datastore>
        <name>ds:startup</name>
        <schema>config-schema</schema>
      </datastore>
      <datastore>
        <name>ds:running</name>
        <schema>config-schema</schema>
      </datastore>
      <datastore>
        <name>ds:operational</name>
        <schema>state-schema</schema>
      </datastore>

      <content-id>75a43df9bd56b92aacc156a2958fbe12312fb285</content-id>
    </yang-library>
```

Appendix C. Example YANG Library Instance for an Advanced Server

The following example extends the example in Appendix B by using modules from [RFC8345] and [RFC8349] to illustrate a slightly more advanced server that:

- Has a module with features only enabled in <operational>; the "ietf-routing" module is supported in <running>, <startup>, and <operational>, but the "multiple-ribs" and "router-id" features are only enabled in <operational>. Hence, the "router-id" leaf may be read but not configured.

- Supports a dynamic configuration datastore "example-ds-ephemeral", with only the "ietf-network" and "ietf-network-topology" modules configurable via a notional dynamic configuration protocol.

- Shows an example of datastore-specific deviations. The "example-vendor-hardware-deviations" module is included in the schema for <operational> to remove data nodes that cannot be supported by the server.

- Shows how module-sets can be used to organize related modules together.

Newline characters in leaf values are added for formatting reasons.

```
<yang-library
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores"
    xmlns:ex-ds-eph="urn:example:ds-ephemeral">

  <module-set>
    <name>config-state-modules</name>
    <module>
      <name>ietf-interfaces</name>
      <revision>2018-02-20</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-interfaces
      </namespace>
    </module>
    <module>
      <name>ietf-ip</name>
      <revision>2018-02-22</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-ip
      </namespace>
    </module>
    <module>
      <name>ietf-routing</name>
      <revision>2018-03-13</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-routing
      </namespace>
    </module>
    <import-only-module>
      <name>ietf-yang-types</name>
      <revision>2013-07-15</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-yang-types
      </namespace>
```

```
      </import-only-module>
      <import-only-module>
        <name>ietf-inet-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-inet-types
        </namespace>
      </import-only-module>
    </module-set>

    <module-set>
      <name>config-only-modules</name>
      <module>
        <name>ietf-routing</name>
        <revision>2018-03-13</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-routing
        </namespace>
      </module>
    </module-set>

    <module-set>
      <name>dynamic-config-state-modules</name>
      <module>
        <name>ietf-network</name>
        <revision>2018-02-26</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-network
        </namespace>
      </module>
      <module>
        <name>ietf-network-topology</name>
        <revision>2018-02-26</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-network-topology
        </namespace>
      </module>
      <import-only-module>
        <name>ietf-inet-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-inet-types
        </namespace>
      </import-only-module>
    </module-set>

    <module-set>
      <name>state-only-modules</name>
```

```
        <module>
          <name>ietf-hardware</name>
          <revision>2018-03-13</revision>
          <namespace>
            urn:ietf:params:xml:ns:yang:ietf-hardware
          </namespace>
          <deviation>example-vendor-hardware-deviations</deviation>
        </module>
        <module>
          <name>ietf-routing</name>
          <revision>2018-03-13</revision>
          <namespace>
            urn:ietf:params:xml:ns:yang:ietf-routing
          </namespace>
          <feature>multiple-ribs</feature>
          <feature>router-id</feature>
        </module>
        <module>
          <name>example-vendor-hardware-deviations</name>
          <revision>2018-01-31</revision>
          <namespace>
            urn:example:example-vendor-hardware-deviations
          </namespace>
        </module>
        <import-only-module>
          <name>ietf-inet-types</name>
          <revision>2013-07-15</revision>
          <namespace>
            urn:ietf:params:xml:ns:yang:ietf-inet-types
          </namespace>
        </import-only-module>
        <import-only-module>
          <name>ietf-yang-types</name>
          <revision>2013-07-15</revision>
          <namespace>
            urn:ietf:params:xml:ns:yang:ietf-yang-types
          </namespace>
        </import-only-module>
        <import-only-module>
          <name>iana-hardware</name>
          <revision>2018-03-13</revision>
          <namespace>
            urn:ietf:params:xml:ns:yang:iana-hardware
          </namespace>
        </import-only-module>
      </module-set>

      <schema>
```

```
      <name>config-schema</name>
      <module-set>config-state-modules</module-set>
      <module-set>config-only-modules</module-set>
    </schema>
    <schema>
      <name>dynamic-config-schema</name>
      <module-set>dynamic-config-state-modules</module-set>
    </schema>
    <schema>
      <name>state-schema</name>
      <module-set>config-state-modules</module-set>
      <module-set>dynamic-config-state-modules</module-set>
      <module-set>state-only-modules</module-set>
    </schema>

    <datastore>
      <name>ds:startup</name>
      <schema>config-schema</schema>
    </datastore>
    <datastore>
      <name>ds:running</name>
      <schema>config-schema</schema>
    </datastore>
    <datastore>
      <name>ex-ds-eph:ds-ephemeral</name>
      <schema>dynamic-config-schema</schema>
    </datastore>
    <datastore>
      <name>ds:operational</name>
      <schema>state-schema</schema>
    </datastore>

    <content-id>14782ab9bd56b92aacc156a2958fbe12312fb285</content-id>
  </yang-library>
```

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Robert Wilton
Cisco Systems

Email: rwilton@cisco.com