Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

J. Clarke
B. Claise
Cisco Systems, Inc.

July 3, 2017

YANG module for yangcatalog.org
draft-clacla-netmod-model-catalog-00

Abstract

This document specifies a YANG module that contains metadata related to YANG modules and vendor implementations of those 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 January 4, 2018.

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] [RFC7950] became the standard data modeling language of choice. Not only is it used by the IETF for specifying models, but also in many Standard Development Organizations (SDOs), consortia, and open-source projects: the IEEE, the Broadband Forum (BFF), DMTF, MEF, ITU, OpenDaylight, Open ROADM, Openconfig, sysrepo, and more.

With the rise of data model-driven management and the success of YANG as a key piece comes a challenge: the entire industry develops YANG models. In order for operators to automate coherent services, the industry must ensure the following:

1. Data models must work together

2. There exists a toolchain to help one search and understand models

3. Metadata is present to further describe model attributes

The site <yangcatalog.org>(and the YANG catalog that it provides) is an attempt to address these key tenants. From a high level point of view, the goal of this catalog is to become a reference for all YANG modules available in the industry, for both YANG developers (to search on what exists already) and for operators (to discover the more mature YANG models to automate services). This YANG catalog should not only contain pointers to the YANG modules themselves, but also contains metadata related to those YANG modules: What is the module type (service model or not?); what is the maturity level? (e.g., for the IETF: is this an RFC, a working group document or an individual draft?); is this module implemented?; who is the contact?; is there open-source code available? And we expect much more in the future. The industry starts to understand that the metadata related to YANG models become equally important as the YANG models themselves.

This document defines a YANG [RFC6020] module called yang-catalog.yang that contains the metadata definitions, that must be kept next to the YANG module specification. The YANG module design is based on experience and real code. As such, it’s expected that this YANG module will be a living document. Furthermore, new use cases, which require new metadata in this YANG module, are discovered on a regular basis.

1.1. Status of Work and Open Issues

The top open issues are:

1. Add a leaf to indicate the tree type relative to [I-D.ietf-netmod-revised-datastores]

2. Obtain feedback from vendors and SDOs

3. Socialize module at the IETF and incorporate feedback

2. Learning from Experience

While implementing the catalog and tools at yangcatalog.org, we initially looked at the "Catalog and registry for YANG models" [I-D.openconfig-netmod-model-catalog] as a starting point but we quickly realized that the objectives are different. As a consequence, even if some of the information is similar, this YANG module started to diverge. [TO BE EXPANDED ON THE DIFFERENT OBJECTIVES.] Below are our observations and the justifications for the divergence.

2.1. YANG Module Library

In order for the YANG catalog to become a complete inventory of which models are supported on the different platforms, content such as the support of the YANG module/deviation/feature/etc. should be easy to import and update. An easy way to populate this information is to have a similar structure as the YANG Module Library [RFC7895]. That way, querying the YANG Module Library from a platform provides, directly in the right format, the input for the YANG catalog inventory.

There are some similar entries between the YANG Module Library and the Openconfig catalog. For example, the Openconfig catalog model defines a "uri" leaf which is similar to "schema" from [RFC7895]). And this adds to the overall confusion.

2.2. YANG Catalog Data Model

The structure of the yang-catalog.yang module described in this document is found below:

```
module: yang-catalog
    +--rw catalog
       +--rw modules
       |  +--rw module* [name revision]
       |     +--rw name                  yang:yang-identifier
       |     +--rw revision              union
       |     +--rw datastore*            identityref
       |     +--rw schema?               inet:uri
       |     +--rw namespace             inet:uri
       |     +--rw feature*              yang:yang-identifier
       |     +--rw deviation* [name revision]
       |     |  +--rw name        yang:yang-identifier
       |     |  +--rw revision    union
       |     +--rw conformance-type      enumeration
       |     +--rw submodule* [name revision]
       |     |  +--rw name        yang:yang-identifier
       |     |  +--rw revision    union
       |     |  +--rw schema?     inet:uri
       |     +--rw document-name?        string
       |     +--rw author-email?         string
       |     +--rw compilation-status?   enumeration
       |     +--rw compilation-result?   string
       |     +--rw reference?            string
       |     +--rw prefix?               string
       |     +--rw yang-version?         string
       |     +--rw organization?         string
       |     +--rw description?          string
```

```
        |     +--rw contact?              string
        |     +--rw maturity-level?       enumeration
        |     +--rw implementations
        |        +--rw implementation* [vendor platform software-version softwar
e-flavor]
        |           +--rw vendor              string
        |           +--rw platform            string
        |           +--rw software-version    string
        |           +--rw software-flavor     string
        |           +--rw os-version?         string
        |           +--rw feature-set?        string
        |           +--rw os-type?            string
        +--rw vendors
          +--rw vendor* [name]
              +--rw name         string
              +--rw platforms
                +--rw platform* [name]
                    +--rw name                 string
                    +--rw software-versions
                      +--rw software-version* [name]
                          +--rw name                string
                          +--rw software-flavors
                            +--rw software-flavor* [name]
                                +--rw name         string
                                +--rw protocols
                                |  +--rw protocol* [name]
                                |     +--rw name                enumeration
                                |     +--rw protocol-version?   string
                                |     +--rw capabilities*       string
                                +--rw modules
                                  +--rw module* [name revision]
                                      +--rw name        -> ../../../../../../../
../../../../modules/module/name
                                      +--rw revision    -> ../../../../../../../
../../../../modules/module/revision
```

Various elements of this module tree will be discussed in the subsequent sections.

2.3. Module Sub-Tree

Each module in the YANG Catalog is enumerated by its metadata and by various vendor implementations. Within the "module" sub-tree, each module is listed using its YANG Module Library [RFC7895] "module-list" grouping information. The yang-catalog module then augments the grouping to add metadata elements that will aid module developers and module consumers alike in understanding the relative maturity, compilation status, and the support contact(s) of each YANG module.

```
+--rw module* [name revision]
    +--rw name                  yang:yang-identifier
    +--rw revision              union
    +--rw schema?               inet:uri
    +--rw namespace             inet:uri
    +--rw feature*              yang:yang-identifier
    +--rw deviation* [name revision]
    |  +--rw name        yang:yang-identifier
    |  +--rw revision    union
    +--rw conformance-type      enumeration
    +--rw submodule* [name revision]
    |  +--rw name        yang:yang-identifier
    |  +--rw revision    union
    |  +--rw schema?     inet:uri
    +--rw document-name?        string
    +--rw author-email?         string
    +--rw compilation-status?   enumeration
    +--rw compilation-result?   string
    +--rw reference?            string
    +--rw prefix?               string
    +--rw yang-version?         string
    +--rw organization?         string
    +--rw description?          string
    +--rw contact?              string
    +--rw module-type?          enumeration
    +--rw maturity-level?       enumeration
    +--rw implementations
      +--rw implementation* [vendor platform software-version software-flavor]
          +--rw vendor              string
          +--rw platform            string
          +--rw software-version    string
          +--rw software-flavor     string
          +--rw os-version?         string
          +--rw feature-set?        string
          +--rw os-type?            string
```

Many of these additional metadata fields are self-explanatory, especially given their descriptions in the module itself. However, those requiring additional explanation or context as to why they are needed are described in the subsequent sections.

2.4. Compilation Information

For the inventory to be complete, YANG modules at different stage of maturity should be taken into account, including YANG modules that are clearly work-in-progress, i.e. that do not validate correctly, either because of a faulty YANG constructs, because of a faulty imported YANG module, or simply because of a warnings. Note that

some of those warnings are not always show-stoppers from a code generation point of view. Nonetheless, the compilation or validation status, along with the compilation output, provide a clear indication on the YANG module development phase and stability.

```
      leaf compilation-status {
        description
          "Status of the module, whether it was possible to compile this YANG mo
dule or
           there are still some errors/warnings.";
        type enumeration {
          enum PASSED {
            value 0;
            description
              "In case that all compilers were able to compile this YANG module
without
               any error/warning.";
          }
          enum PASSED-WITH-WARNINGS {
            value 1;
            description
              "In case that all compilers were able to compile this YANG module
without
               any error, but at least one of them caught some warning.";
          }
          enum FAILED {
            value 2;
            description
              "In case that at least one of compilers found some error while
              compiling this YANG module.";
          }
          enum MISSING {
            value 3;
            description
              "In case that there is not sufficient information about compilatio
n status.";
          }
        }
      }
      leaf compilation-result {
        type string;
        description
          "Result of the compilation explaining specifically what error or warni
ng occurred.
           This is not existing if compilation status is PASSED.";
      }
```

Figure 1: Compilation Status and Compilation Result

The compilation status and result have been added as two extra leaves, for each YANG module.

2.5. Maturity Level

Models also have inherent maturity levels from their respective Standards Development Organizations (SDOs). These maturity levels will help model consumers understand how complete, tested, etc. a model is.

```
    leaf maturity-level {
      description
        "The current maturity of the module with respect to the body that crea
ted it.
         This allows one to understand where the module is in its overall life
 cycle.";
      type enumeration {
        enum ratified {
          value 0;
          description
            "Maturity of a module that is fully approved (e.g., a standard).";
        }
        enum working-group {
          value 1;
          description
            "Maturity of a module that is actively being developed by a organi
zation towards ratification.";
        }
        enum individual {
          value 2;
          description
            "Maturity of a module that has been initially created, but has no
official
             organization-level status.";
        }
      }
    }
```

This enumeration mapping has been implemented for the YANG modules from IETF and BBF. With respect to vendor-specific modules, this same enumeration should be used and mapped to the internal vendor release or development names. Once a module has been completed, fully tested, and is stable, its maturity level should be "ratified".

2.6. Implementation

As of version 02 of openconfig-model-catalog.yang [I-D.openconfig-netmod-model-catalog] it is not possible to identify the implementations of one specific module. Instead modules are grouped into feature-bundle, and feature-bundles are implemented by devices. Because of this, we added our own implementation sub-tree under each module to yang-catalog.yang. Our implementation sub-tree is:

```
+--rw implementation* [vendor platform software-version software-flavor]
  +--rw vendor              string
  +--rw platform            string
  +--rw software-version    string
  +--rw software-flavor     string
  +--rw os-version?         string
  +--rw feature-set?        string
  +--rw os-type?            string
```

The keys in this sub-tree can be used in the "vendor" sub-tree defined below to walk through each vendor, platform, and software release to get a full list of supported YANG modules for that release.

The "software-flavor" key leaf identifies a variation of a specific version where YANG model support may be different. Depending on the vendor, this could be a license, additional software component, or a feature set.

The other non-key leaves in the implementation sub-tree represent optional elements of a software release that some vendors may choose to use for informational purposes.

2.7. Vendor Sub-Tree

The vendor sub-tree provides a way, especially for module consumers, to walk through a specific device and software release to find a list of modules supported therein. This sub-tree turns the "implementation" sub-tree on its head to provide an optimized index for one wanting to go from a platform to a full list of modules.

In addition to the module list, the vendor sub-tree lists the YANG-based protocols (e.g., NETCONF or RESTCONF) that the platforms support.

```
+--rw vendor* [name]
    +--rw name         string
    +--rw platforms
    +--rw platform* [name]
          +--rw name                string
          +--rw software-versions
              +--rw software-version* [name]
                +--rw name               string
                +--rw software-flavors
                    +--rw software-flavor* [name]
                      +--rw name         string
                      +--rw protocols
                      |  +--rw protocol* [name]
                      |     +--rw name                enumeration
                      |     +--rw protocol-version?   string
                      |     +--rw capabilities*       string
                      +--rw modules
                          +--rw module* [name revision]
                            +--rw name        -> ../../../../../../../../../../.
./modules/module/name
                            +--rw revision    -> ../../../../../../../../../../.
./modules/module/revision
```

This sub-tree structure also enables one to look for YANG modules for a class of platforms (e.g., list of modules for Cisco, or list of modules for Cisco ASR9000 routers) instead of only being able to look for YANG modules for a specific platform and software release.

2.8. Regex Expression Differences

Another challenge with using [I-D.openconfig-netmod-model-catalog] as the canonical catalog is the regular expression syntax it uses. The Openconfig module uses a POSIX-compliant regular expression syntax whereas YANG-based protocol implementations like ConfD [1] expect the IETF-chosen W3C syntax. In order to load the Openconfig catalog in such engines, changes to the regular expression syntax had to be done, and these one-off changes are not supportable.

3. YANG Catalog Use Cases

The YANG Catalog module is currently targeted to address the following use cases.

3.1. YANG Search Metadata

The yangcatalog.org toolchain provides a service for searching [2] for YANG modules based on keywords. The resulting search data currently stores the module and node metadata in a proprietary format along with the search index data. By populating the yang-catalog module, this search service can instead pull the metadata from the

implementation of the module. Populating this instance of the yang-catalog module will be using an API that is still under development, but will ultimately allow SDOs and vendors to provide metadata and ensure the search service has the most up-to-date data for all available modules.

3.2. YANG In Images

By organizing the yang-catalog module so that one can either find all implementations for a given module, or find all modules supported by a vendor platform and software release, the catalog will provide a straight-forward way for one to understand the extent of YANG module support in participating vendors' software images. Eventually a web-based graphical interface will be connected to this on yangcatalog.org to make it easier for consumers to leverage the instance of the yang-catalog module for this use case.

4. YANG Catalog YANG module

The structure of the model defined in this document is described by the YANG module below.

```
<CODE BEGINS> file "yang-catalog@2017-07-03.yang"
  module yang-catalog {
   namespace "urn:ietf:params:xml:ns:yang:yang-catalog";
   prefix yc;

   import ietf-yang-library {
     prefix yanglib;
   }

   organization
     "yangcatalog.org";
   contact
     "Benoit Claise <bclaise@cisco.com>

      Joe Clarke <jclarke@cisco.com>";
   description
     "This module contains metadata pertinent to each YANG module, as
      well as a list of vendor implementations for each module.  The
      structure is laid out in such a way as to make it possible to
      locate metadata and vendor implementation on a per-module basis
      as well as obtain a list of available modules for a given
      vendor's platform and specific software release.";

   revision 2017-07-03 {
     description
       "Initial revision.";
```

```
    reference "
      YANG Catalog <https://yangcatalog.org>";
  }

  container catalog {
    description
      "Root container of yang-catalog holding two main branches -
       modules and vendors. The modules sub-tree contains all the modules in
       the catalog and all of their metadata with their implementations.
       The vendor sub-tree holds modules for specific vendors, platforms,
       software-versions, and software-flavors. It contains reference to a
       name and revision of the module in order to reference the module's full
       set of metadata.";
    container modules {
      description
        "Container holding the list of modules";
      uses yanglib:module-list;
    } // end of modules

    container vendors {
      description
        "Container holding lists of organizations that publish YANG modules.";
      list vendor {
        key name;
        description
          "List of organizations publishing YANG modules.";
        leaf name {
          type string;
          description
            "Name of the maintaining organization -- the name should be
             supplied in the official format used by the organization.
             Standards Body examples:
               IETF, IEEE, MEF, ONF, etc.
             Commercial entity examples:
               AT&T, Facebook, <Vendor>
             Name of industry forum examples:
             OpenConfig, OpenDaylight, ON.Lab";
        }
        container platforms {
          description "Container holding list of platforms.";
          list platform {
            key name;
            description
              "List of platforms under specific vendor";
            leaf name {
              type string;
              description
                "Name of the platform";
```

```
            }
            container software-versions {
              description "Container holding list of versions of software versi
ons.";
              list software-version {
                key name;
                description
                  "List of version of software versions under specific vendor,
platform.";
                leaf name {
                  type string;
                  description
                    "Name of the version of software.  With respect to most net
work device appliances,
                     this will be the operating system version.  But for other
YANG module
                     implementation, this would be a version of appliance softw
are.  Ultimately,
                     this should correspond to a version string that will be re
cognizable by
                     the consumers of the platform.";
                }
                container software-flavors {
                  description "Container holding list of software flavors.";
                  list software-flavor {
                    key name;
                    description
                      "List of software flavors under specific vendor, platform
, software-version.";
                    leaf name {
                      type string;
                      description
                        "A variation of a specific version where
                         YANG model support may be different.  Depending on the
 vendor, this could
                         be a license, additional software component, or a feat
ure set.";
                    }
                    container protocols {
                      description
                        "List of the protocols";
                      list protocol {
                        key name;
                        description
                          "YANG-based protocol that is used on the device.  Thi
s enumeration will
                           is expected to be augmented to list other protocol n
ames.";
                        leaf name {
                          type enumeration {
                            enum netconf {
                              description
                                "NETCONF protocol described in RFC 6241";
                            }
                            enum restconf {
                              description
                                "RESTCONF protocol described in RFC 8040";
                            }
                          }
```

```
                          description
                            "Name of the YANG-based protocol that is supported.
";
                        } // end of name
                        leaf protocol-version {
                          type string;
                          description
                            "Version of the specific protocol.";
                        }
                        leaf-list capabilities {
                          type string;
                          description
                            "Listed name of capabilities that are
                             supported by the specific device.";
                        }
                      } // end of protocol
                    } // end of protocols
                    container modules {
                      description
                        "Container holding list of modules.";
                      list module {
                        key "name revision";
                        description
                         "List of references to YANG modules under specific ven
dor, platform, software-version,
                          software-flavor.  Using these references, the complet
e set of metadata can be
                          retrieved for each module.";
                        leaf name {
                          type leafref {
                            path "../../../../../../../../../../../modules/modu
le/name";
                          }
                          description
                            "Reference to a name of the module that is containe
d in specific vendor, platform,
                             software-version, software-flavor.";
                        }
                        leaf revision {
                          type leafref {
                            path "../../../../../../../../../../../modules/modu
le/revision";
                          }
                          description
                            "Reference to a revision of the module that is cont
ained in specific vendor,
                             platform, software-version, software-flavor.";
                        }
                      } // end of list module
                    } // end of container modules
                  } // end of software-flavor
                } // end of software-flavors
              } // end of software-version
            } // end of software-versions
          } // end of platform
```

```
        } // end of platforms
      } // end of vendor
    } // end of vendors
  } //end of catalog

  augment "/catalog/modules/module" {
    uses module-data;
    container implementations {
      description
        "Container holding lists of per-module implementation details.";
      list implementation{
        key "vendor platform software-version software-flavor";
        description
          "List of module implementations.";
        leaf vendor {
          type string;
          description
            "Organization that created this module.";
        }
        leaf platform {
          type string;
          description
            "Platform on which this module is implemented.";
        }
        leaf software-version {
          type string;
          description
            "Name of the version of software.  With respect to most network dev
ice appliances,
             this will be the operating system version.  But for other YANG mod
ule
             implementation, this would be a version of appliance software.  Ul
timately,
             this should correspond to a version string that will be recognizab
le by
             the consumers of the platform.";
        }
        leaf software-flavor {
          type string;
          description
            "A variation of a specific version where
             YANG model support may be different.  Depending on the vendor, thi
s could
             be a license, additional software component, or a feature set.";
        }
        leaf os-version {
          type string;
          description
            "Version of the operating system using this module.  This is primar
ily useful if
             the software implementing the module is an application that requir
es a specific
             operating system.";
        }
        leaf feature-set {
```

```
           type string;
           description
             "An optional feature of the software that is required in order to i
mplement this
              module.  Some form of this must be incorporated in software-versio
n or
              software-flavor, but can be broken out here for additional clarity
.";
         }
         leaf os-type {
           type string;
           description
             "Type of the operating system using this module.  This is primarily
 useful if
              the software implementing the module is an application that requir
es a
              specific operating system.";
         }
       }
     }
     description
       "This table augments the per-module metadata set and provides details abo
ut
        vendor implementations for each module.";
   }

   grouping module-data {
     leaf document-name {
       type string;
       description
         "The name of the document from which the module was extracted or taken;
          or that provides additional context about the module.";
     }
     leaf author-email {
       type string;
       description
        "Contact email of the author who created this module.";
     }
     leaf compilation-status {
       type enumeration {
         enum PASSED {
           value 0;
           description
             "In case that all compilers were able to compile this YANG module w
ithout
              any error/warning.";
         }
         enum PASSED-WITH-WARNINGS {
           value 1;
           description
             "In case that all compilers were able to compile this YANG module w
ithout
              any error, but at least one of them caught some warning.";
         }
         enum FAILED {
           value 2;
```

```
           description
             "In case that at least one of compilers found some error while
             compiling this YANG module.";
         }
         enum MISSING {
           value 3;
           description
             "In case that there is not sufficient information about compilation
 status.";
         }
       }
       description
         "Status of the module, whether it was possible to compile this YANG mod
ule or
          there are still some errors/warnings.";
     }
     leaf compilation-result {
       type string;
       description
         "Result of the compilation explaining specifically what error or warnin
g occurred.
          This is not existing if compilation status is PASSED.";
     }
     leaf reference {
       type string;
       description
         "A string that is used to specify a textual cross-reference to an exter
nal document, either
          another module that defines related management information, or a docum
ent that provides
          additional information relevant to this definition.";
     }
     leaf prefix {
       type string;
       description
         "Statement of yang that is used to define the prefix associated with
          the module and its namespace. The prefix statement's argument is
          the prefix string that is used as a prefix to access a module. The
          prefix string MAY be used to refer to definitions contained in the
          module, e.g., if:ifName.";
     }
     leaf yang-version {
       type string;
       default "1.0";
       description
         "The optional yang-version statement specifies which version of the
          YANG language was used in developing the module. The statement's
          argument is a string. If present, it MUST contain the value 1,
          which is the current YANG version and the default value.";
     }
     leaf organization {
       type string;
       description
```

```
          "This statement defines the party responsible for this
           module.  The argument is a string that is used to specify a textual
           description of the organization(s) under whose auspices this module
           was developed.";
      }
      leaf description {
        type string;
        description
          "This statement takes as an argument a string that
           contains a human-readable textual description of this definition.
           The text is provided in a language (or languages) chosen by the
           module developer; for the sake of interoperability, it is RECOMMENDED
           to choose a language that is widely understood among the community of
           network administrators who will use the module.";
      }
      leaf contact {
        type string;
        description
          "This statement provides contact information for the module.
           The argument is a string that is used to specify contact information
           for the person or persons to whom technical queries concerning this
           module should be sent, such as their name, postal address, telephone
           number, and electronic mail address.";
      }
      leaf module-type {
        type enumeration {
          enum module {
            value 0;
            description "If YANG file contains module.";
          }
          enum submodule {
            value 1;
            description "If YANG file contains sub-module.";
          }
        }
        description "Whether a file contains a YANG module or sub-module.";
      }
      leaf maturity-level {
        type enumeration {
          enum ratified {
            value 0;
            description
              "Maturity of a module that is fully approved (e.g., a standard).";
          }
          enum working-group {
            value 1;
            description
              "Maturity of a module that is actively being developed by a organiz
ation towards ratification.";
```

```
         }
         enum individual {
           value 2;
           description
             "Maturity of a module that has been initially created, but has no o
fficial
              organization-level status.";
         }
       }
       description
         "The current maturity of the module with respect to the body that creat
ed it.
          This allows one to understand where the module is in its overall life
cycle.";
     }
     description
       "Grouping of YANG module metadata that extends the common list defined in
 the YANG
        Module Library (RFC 7895).";
  }
 }
<CODE ENDS>
```

5. Security Considerations

The goal of the YANG Catalog module and yangcatalog.org is to document a large library of YANG modules and their implementations. Already, we have seen some SDOs hestitant to provide modules that have not reached a "ratified" maturity level because of intellectual property leakage concerns or simply organization process that mandates only fully ratified modules can be published. Care must be paid that through private automated testing and validation of such modules that their metadata does not leak before the publishing organization approves the release of such data.

Similarly, from a vendor implementation standpoint, data that is exposed to the catalog before the vendor has fully vetted it could cause confusion amongst that vendor’s customers or reveal product releases to the market before they have been officially announced.

Ultimately, there is a balance to be struck with respect to providing a rich library of YANG module metadata, and doing so at the right time to avoid information leakage.

6. IANA Considerations

No action.

7. References

7.1. Normative References

[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>.

[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>.

7.2. Informative 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-02 (work in progress), May 2017.

[I-D.openconfig-netmod-model-catalog] Shaikh, A., Shakir, R., and K. D'Souza, "Catalog and registry for YANG models", draft-openconfig-netmod-model-catalog-02 (work in progress), March 2017.

7.3. URIs

[1] https://developer.cisco.com/site/confD/index.gsp

[2] https://yangcatalog.org/yang-search

Appendix A. Acknowledgments

The authors would like to thanks Miroslav Kovac for this help on this YANG module and the yangcatalog.org implementation.

The RFC text was produced using Marshall Rose's xml2rfc tool.

Appendix B. Contributors

Contributors' Addresses

TBD

Authors' Addresses

Joe Clarke
Cisco Systems, Inc.
7200-12 Kit Creek Rd
Research Triangle Park, North Carolina
United States of America

Phone: +1-919-392-2867
Email: jclarke@cisco.com

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com