Internet Engineering Task Force	A. Bierman
Internet-Draft	Netconf Central
Intended status: BCP	January 24, 2009
Expires: July 28, 2009	

TOC

Guidelines for Authors and Reviewers of YANG Data Model Documents
draft-bierman-netmod-yang-usage-00

Status of this Memo

Copyright Notice

Abstract

This memo provides guidelines for authors and reviewers of standards track specifications containing YANG data model modules. Applicable portions may be used as a basis for reviews of other YANG data model documents. Recommendations and procedures are defined, which are intended to increase interoperability and usability of NETCONF implementations which utilize YANG data model modules.

Table of Contents

1. Introduction

TOC

The standardization of network configuration interfaces for use with the NETCONF (Enns, R., "NETCONF Configuration Protocol," December 2006.) [RFC4741] protocol requires a modular set of data models, which can be reused and extended over time.

This document defines a set of usage guidelines for standards track documents containing YANG (Bjorklund, M., “YANG - A data modeling language for NETCONF,” April 2010.) [I-D.ietf-netmod-yang] data models. It is similar to the MIB usage guidelines specification [RFC4181] (Heard, C., “Guidelines for Authors and Reviewers of MIB Documents,” September 2005.) in intent and structure.

Many YANG constructs are defined as optional to use, such as the description clause. However, in order to maximize interoperability of NETCONF implementations utilizing YANG data models, it is desirable to define a set of usage guidelines which may require a higher level of compliance than the minimum level defined in the YANG specification.

A new IANA registry is needed to support YANG. This registry will allow YANG module namespace and other definitions to be centrally located, minimizing name collisions, and providing an authoritative status of each YANG module.

The YANG Module Registry will support YANG modules, as well as YANG submodules which utilize a 'virtual' module definition. A virtual module contains only the module header, submodule include statements, and meta statements. The Submodule Registration procedure [ed: IANA procedure TBD] is used to publish specifications containing YANG submodules which extend a virtual module. This procedure allows the main module revision statement and include statement to be updated, without requiring publication or a separate RFC to contain the main module. Refer to Section 5 (YANG Module Registry) for more details.

The NETCONF stack can be conceptually partitioned into four layers.

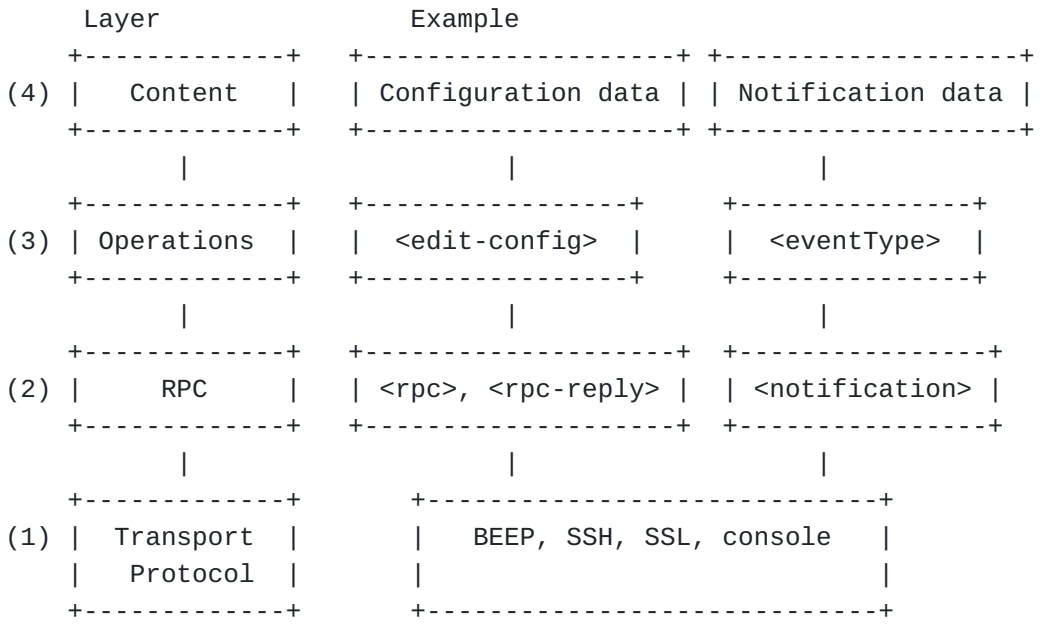

Figure 1

This document defines usage guidelines related to the NETCONF operations layer (3), and NETCONF content layer (4).
It also contains a definition for a registry for YANG Modules, which can be used to locate documents which contain standards-track modules or submodules.

---

2. Terminology

TOC

2.1. Requirements Notation

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.).

2.2. NETCONF Terms

TOC

The following terms are defined in [RFC4741] (Enns, R., "NETCONF Configuration Protocol," December 2006.) and are not redefined here:

- *agent
- *application
- *capabilities
- *manager
- *operation
- *RPC

2.3. YANG Terms

TOC

The following terms are defined in [I-D.ietf-netmod-yang] (Bjorklund, M., "YANG - A data modeling language for NETCONF," April 2010.) and are not redefined here:

- *data node
- *module
- *submodule
- *namespace
- *version

2.4. Terms

TOC

The following terms are used throughout this document:

- *YAM: Shorthand term for a YANG data model module or submodule, used for properties which apply to both modules and submodules.

When describing properties which are specific to modules, the term 'YANG module', or simply 'module', is used instead. When describing properties which are specific to submodules, the term 'YANG submodule', or simply 'module' is used instead.

*Published Document: A stable release of a YAM, usually contained in an RFC.

*Unpublished Document: An unstable release of a YAM, usually contained in an Internet Draft.

*Virtual Module: A YANG module which does not contain any body statements, and is maintained in a registry. The body statements are defined in submodules, in one or more documents, and 'included' in the main module via a registry entry for the main module.

---

3. General Documentation Guidelines

TOC

YANG data model modules (YAMs) under review are likely to be contained in Internet Drafts. All guidelines for Internet Draft authors MUST be followed. These guidelines are available online at:
http://www.isi.edu/in-notes/rfc-editor/instructions2authors.txt
The following sections MUST be present in an Internet Draft containing a YAM:

*YANG data model boilerplate section

*Narrative sections

*Definitions section

*Security Considerations section

*IANA Considerations section

*References section

3.1. YANG Data Model Boilerplate Section

TOC

This section MUST contain a verbatim copy of the latest approved Internet-Standard Management Framework boilerplate, which is available on-line at [ed: URL TBD].

3.2. Narrative Sections

TOC

The narrative part MUST include an overview section that describes the scope and field of application of the YAM(s) defined by the specification and that specifies the relationship (if any) of these YAMs to other standards, particularly to standards containing other YAM modules. The narrative part SHOULD include one or more sections to briefly describe the structure of the YAMs defined in the specification.
If the YAM(s) defined by the specification import definitions from other YAMs (except for those defined in the YANG (Bjorklund, M., "YANG - A data modeling language for NETCONF," April 2010.) [I-D.ietf-netmod-yang] or YANG Types (Schoenwaelder, J., "Common YANG Data Types," April 2010.) [I-D.ietf-netmod-yang-types] documents) or are always implemented in conjunction with other YAMs, then those facts MUST be noted in the overview section, as MUST any special interpretations of objects in other YAMs.

3.3. Definitions Section

TOC

This section contains the YAM(s) defined by the specification. These modules MUST be written in YANG [I-D.ietf-netmod-yang] (Bjorklund, M., "YANG - A data modeling language for NETCONF," April 2010.).
See Section 4 (YANG Usage Guidelines) for guidelines on YANG usage.

3.4. Security Considerations Section

TOC

Each specification that defines one or more YAMs MUST contain a section that discusses security considerations relevant to those modules. This section MUST be patterned after the latest approved template (available at [ed: URL TBD]).
In particular, writable YAM objects that could be especially disruptive if abused MUST be explicitly listed by name and the associated security risks MUST be spelled out; similarly, readable YAM objects that contain especially sensitive information or that raise significant privacy concerns MUST be explicitly listed by name and the reasons for the sensitivity/privacy concerns MUST be explained.

3.5. IANA Considerations Section

TOC

In order to comply with IESG policy as set forth in http://www.ietf.org/ID-Checklist.html, every Internet-Draft that is submitted to the IESG for publication MUST contain an IANA Considerations section. The requirements for this section vary depending what actions are required of the IANA.
Refer to [TBD] for details on the structure of the YANG registries maintained by the IANA.

3.5.1. Documents that Create a New Name Space

TOC

If an Internet-Draft defines a new name space that is to be administered by the IANA, then the document MUST include an IANA Considerations section, specifies how the name space is to be administered.
Specifically, if any YANG module namespace statement value contained in the document is not already registered with IANA, then a new YANG Namespace registry entry must be requested from the IANA [ed: procedure TBD].

3.5.2. Documents that Extend an Existing Name Space

TOC

If an Internet-Draft defines any extensions to a YANG Namespace already administered by the IANA, then the document MUST include an IANA Considerations section, specifies how the name space extension is to be administered.
Specifically, if any YANG submodule belongs-to value contained in the document is associated with a module that contains a namespace statement value equal to a YANG Namespace already administered by the IANA, then a new YANG Module registry entry and YANG Namespace Update Procedure must be requested from the IANA [ed: procedure TBD].

3.6. Reference Sections

TOC

[ed: 2223bis text TBD]
For every import or include statement which appears in a YAM contained in the specification, which identifies a YAM in a separate document, a corresponding normative reference to that document MUST appear in the

Normative References section. The reference MUST correspond to the specific YAM version actually used within the specification.
If any YANG submodule contained in the specification contains a 'belongs-to' statement value which identifies a 'virtual' YANG module maintained in the IANA YANG Module Registry, then a corresponding normative reference to the registry identifier MUST appear in the Normative References section. The registry entry MUST be properly updated, using the appropriate procedures [ed: IANA procedures TBD].

3.7. Copyright Notices

TOC

The proper copyright notices MUST be present in the module description statement. [ed.: See RFC 4181, 3.7. Exact text for insertion is TBD.]

3.8. Intellectual Property Section

TOC

The proper IPR statements MUST be present in the document, according to the most current Internet Draft boilerplate. [ed.: actual IETF IPR text reference TBD]

---

4. YANG Usage Guidelines

TOC

In general, YAMs in IETF standards-track specifications MUST comply with all syntactic and semantic requirements of YANG.
[I-D.ietf-netmod-yang] (Bjorklund, M., “YANG - A data modeling language for NETCONF,” April 2010.). The guidelines in this section are intended to supplement the YANG specification, which is intended to define a minimum set of conformance requirements.
In order to promote interoperability and establish a set of practices based on previous experience, the following sections establish usage guidelines for specific YANG constructs.
Only guidelines which clarify or restrict the minimum conformance requirements are included here.

4.1. Identifiers

TOC

Identifiers for modules, submodules, typedefs, groupings, data objects, rpcs, and notifications MUST be between 1 and 63 characters in length.

4.2. Defaults

TOC

In general, it is suggested that sub-statements containing default values SHOULD NOT be present. For example, 'status current;', 'config true;', 'mandatory false;', and 'max-elements unbounded;' are common defaults which would make the YAM difficult to read if used everywhere they are allowed.
Instead, it is suggested that common statements SHOULD only be used when being set to a value other than the default value.

4.3. Conditional Statements

TOC

A YAM may be conceptually partitioned in several ways, using the 'if-feature' and/or 'when' statements. In addition, NETCONF capabilities are designed to identify optional functionality.
Data model designers need to carefully consider all modularity aspects, including the use of YANG conditional statements.
Objects SHOULD NOT directly reference NETCONF capabilities, in order to specify optional behavior. Instead, a 'feature' statement SHOULD be defined to represent the NETCONF capability, and the 'if-feature' statement SHOULD be used within the object definition.
If the condition associated with the desired semantics is not dependent on any particular instance value within the database, then an 'if-feature' statement SHOULD be used instead of a 'when' statement.
All 'must' and 'when' statements MUST contain valid XPath. If any name tests are present, they MUST contain valid module prefixes and/or data node names.
The 'attribute', 'namespace', 'preceding', 'preceding-sibling', 'following', and 'following-sibling' axis SHOULD NOT be used.
The 'position' and 'last' functions SHOULD NOT be used.
Implicit 'position' function calls within predicates SHOULD NOT be used. (e.g., //chapter[42]).
Data nodes which use the 'int64' and 'uint64' built-in type SHOULD NOT be used within relational expressions.
Data modelers need to be careful not to confuse the YANG value space and the XPath value space. The data types are not the same in both, and conversion between YANG and XPath data types SHOULD be considered carefully.
Explicit XPath data type conversions SHOULD be used (e.g., 'string', 'boolean', or 'number' functions), instead of implicit XPath data type conversions.

TOC

4.4. Header Contents

*The namespace MUST be a globally unique URI, usually assigned by the IANA.

*Until a URI is assigned by the IANA, a temporary namespace string SHOULD be selected which is not likely to collide with other YANG namespaces, such as the filename of the Internet Draft containing the YAM.

*The organization statement MUST be present.

*The contact statement MUST be present.

*The description statement MUST be present.

*If the YAM represents a model defined in one or more external documents, then a reference statement SHOULD be present.

*A revision statement MUST be present for each published version of the YAM.

*Each new revision MUST include a revision date which is higher than any other revision date in the YAM.

*It is acceptable to reuse the same revision statement within unpublished versions (i.e., Internet Drafts), but the revision date MUST be updated to a higher value each time the Internet Draft is re-published.

4.5. Data Types

TOC

*Selection of an appropriate data type (i.e., built-in type, existing derived type, or new derived type) is very subjective and therefore few requirements can be specified on that subject.

*Data model designers SHOULD use the most appropriate built-in data type for the particular application.

*If extensibility of enumerated values is required, then the identityref data type SHOULD be used instead of an enumeration or other built-in type.

*If an appropriate derived type exists in any standard YAM, such as [I-D.ietf-netmod-yang-types] (Schoenwaelder, J., “Common YANG

Data Types," April 2010.), then it SHOULD be used instead of defining a new derived type.

*The description statement MUST be present.

*If the type semantics are defined in an external document, then a reference statement SHOULD be present.

*For string data types, if a machine-readable pattern can be defined for the desired semantics, then one or more pattern statements SHOULD be present.

*For string data types, if the length of the string is not unbounded in all implementations, then a length statement SHOULD be present.

*For numeric data types, if the values allowed by the intended semantics are different than those allowed by the unbounded intrinsic data type (e.g., int32), then a range statement SHOULD be present.

*The signed numeric data types (i.e., 'int8', 'int16', 'int32', and 'int64') SHOULD NOT be used unless negative values are allowed for the desired semantics.

*The 'float32' and 'float64' data types SHOULD only be used if the other numeric data types do not fully represent the desired semantics.

*For enumeration or bits data types, the semantics for each enum or bit SHOULD be documented. A separate description statement (within each enum or bit statement) SHOULD be used, instead of the description statement for the type itself.

*If an appropriate units identifier can be associated with the desired semantics, then a units statement SHOULD be present.

*If an appropriate default value can be associated with the desired semantics, then a default statement SHOULD be present.

*If a significant number of derived types are defined, and it is anticipated that these data types will be reused by multiple YAMs, then these derived types SHOULD be contained in a separate module or submodule, to allow easier reuse without unnecessary coupling.

TOC

4.6. Object Definitions

*The description statement MUST be present.

*If the object semantics are defined in an external document, then a reference statement SHOULD be present.

*The 'anyxml' construct MUST NOT be used within configuration data.

*If there are referential integrity constraints associated with the desired semantics that can be represented with XPath, then one or more must statements SHOULD be present.

*For list and leaf-list objects, if the number of possible instances for all implementations is not unbounded, then the min-elements and/or max-elements statements SHOULD be present.

4.7. RPC Definitions

TOC

*The description statement MUST be present.

*If the RPC method semantics are defined in an external document, then a reference statement SHOULD be present.

*If the RPC method impacts system behavior in some way, it SHOULD be mentioned in the description statement.

*If the RPC method is potentially harmful to system behavior in some way, it MUST be mentioned in the Security Considerations section of the document.

4.8. Notification Definitions

TOC

*The description statement MUST be present.

*If the notification semantics are defined in an external document, then a reference statement SHOULD be present.

TOC

5. YANG Module Registry

This section contains a YANG module registry specification, which can be used to document each release of a module. It can also be used to maintain virtual modules, in which all the body statements are contained in submodules specified in the registry, not in a YANG module within a published RFC or Internet Draft.
In order for YANG submodules to be used effectively within standards track documents, it is desirable to avoid re-publishing an RFC containing the 'main' module, each time a submodule is added or changed.
The use of submodules can effectively reduce the number of XML namespaces used within NETCONF PDUs, but their primary use is to allow flexible partitioning of a single XML namespace into multiple, independent documents, which can be easily extended over time.
The YANG Module Registry is an XML instance document which contains minimal information about the modules represented in the registry.
Each registry has a unique ID, called the 'registry-id'. There are also zero or more 'module' entries.
Each 'module' entry contains the module name, XML namespace, and optional 'url' field to identify its location. If this is a virtual module, then the 'virtual' field will be present.
Within each module entry, there are one or more 'release' entries.
Each 'release' entry contains the publication date of the release. It also contains zero or more 'submodule' entries.
For each submodule included by the main module represented by each 'module' entry, a 'submodule' entry SHOULD be present. Each entry provides the name, release date and an optional 'url' if the submodule is available online.
It is expected that the IANA will maintain the official YANG Module Registry for YAMs contained in published standards-track documents.
It is also expected that procedures for adding a new YANG module, and adding a new release of an existing module, will also be specified.
[ed: A YANG data model and example XML instance document are provided below to demonstrate how such a registry might work. This work is very preliminary and subject to change.]

5.1. YANG Registry Data Model

TOC

This section contains a YANG module definition which represents the information stored in the IANA YANG Module Registry. It is provided for informational purposes only. The actual definition is [TBD].

```
module yang-registry {

    namespace "yang-registry-TBD";

    prefix "yr";

    // for the uri data type
    import yang-types { prefix "yang"; }

    organization "IETF";

    contact
      "Send comments to <andy@netconfcentral.com>.";

    description
       "YANG Module Registry Data Structure";

    revision "2009-01-22" {
        description
          "Initial version.";
    }

    container registry {

        leaf registry-id {
            description
               "Contains the identity of this registry.";
            type yang:uri;
            mandatory true;
        }

        list module {
            key "name";

            unique "namespace";

            leaf name {
               description "YANG module name.";
               // TBD: imported name string type
               type string { length "1..63"; }
            }

            leaf namespace {
                description "YANG module namespace.";
                type yang:uri;
                mandatory true;
            }
```

```
leaf url {
    description
       "URL for this YANG module, if one
        is available.";
    type yang:uri;
}

leaf virtual {
    description
      "If present, then this registry entry
       represents a virtual YANG module,
       which is a YANG module which does not
       contain any body statements.  Instead,
       submodules are used to contain all
       body statements.

       Each release entry within this entry
       is expected to contain all
       the submodule content information for
       this virtual module.";
    type empty;
}

list release {
    description
      "Describes the contents of a specific
       release of a YANG module.  At least
       one entry MUST exist for the most
       current version of the module.";

    min-elements 1;

    key version;

    leaf version {
       description "YANG module release date.";
       // TBD: imported date string type
       // YYYY-MM-DD
       type string { length "10"; }
    }

    list submodule {
        key "name";

        leaf name {
            description "YANG submodule name.";
            // TBD: imported name string type
            type string { length "1..63"; }
        }
```

```
                    leaf version {
                        description "YANG module revision date.";
                        // TBD: imported date string type
                        // YYYY-MM-DD
                       type string { length "10"; }
                       mandatory true;
                    }

                    leaf url {
                        description
                          "URL for this YANG submodule, if one
                           is available.";
                        type yang:uri;
                    }
                  }  // list submodule
              }  // list release
          } // list module
      }  // container registry
  } // module yang-registry
```

Figure 2

5.2. Examples

TOC

This section contains some example registry entries, demonstrating the basic use cases.

```
<?xml version="1.0" encoding="UTF-8"?>
<registry xmlns="yang-registry-TBD">
  <registry-id>
    http://example.com/yang-registry
  </registry-id>
  <module>
    <name>notification</name>
    <namespace>
      urn:ietf:params:xml:ns:netconf:notification:1.0
    </namespace>
    <url>
      ftp://ftp.rfc-editor.org/in-notes/rfc5277.txt
    </url>
    <release>
      <version>2008-07-01</version>
    </release>
  </module>
  <module>
    <name>notification-content</name>
    <namespace>
      urn:ietf:params:xml:ns:netmod:notification
    </namespace>
    <url>
      ftp://ftp.rfc-editor.org/in-notes/rfc5277.txt
    </url>
    <release>
      <version>2008-07-01</version>
    </release>
  </module>
  <module>
    <name>services</name>
    <namespace>
      http://example.com/yang/services
    </namespace>
    <url>
      http://example.com/yang/monitor-tools.yang
    </url>
    <virtual/>
    <release>
      <version>2009-01-23</version>
      <submodule>
        <name>common-types</name>
        <version>2008-11-14</version>
        <url>
          http://example.com/yang/common-types.yang
        </url>
```

```
      </submodule>
      <submodule>
        <name>ping</name>
        <version>2008-11-14</version>
        <url>
          http://example.com/yang/ping.yang
        </url>
      </submodule>
      <submodule>
        <name>traceroute</name>
        <version>2009-01-23</version>
        <url>
          http://example.com/yang/traceroute.yang
        </url>
      </submodule>
    </release>
    <release>
      <version>2008-11-14</version>
      <submodule>
        <name>common-types</name>
        <version>2008-11-14</version>
        <url>
          http://example.com/yang/common-types.yang
        </url>
      </submodule>
      <submodule>
        <name>ping</name>
        <version>2008-11-14</version>
        <url>
          http://example.com/yang/ping.yang
        </url>
      </submodule>
    </release>
  </module>
</registry>
```

Figure 3

TOC

6. IANA Considerations

There are no actions requested of IANA at this time. [ed.: If the YANG Registry approach is pursued, then details for those procedures will need to be defined.]

---

7. Security Considerations

TOC

This document defines documentation guidelines for NETCONF content defined with the YANG data modeling language. It does not introduce any new or increased security risks into the management system. [ed: RFC 4181 style security section TBD]

---

8. Acknowledgments

TOC

The structure and contents of this document are adapted from Guidelines for MIB Documents (Heard, C., “Guidelines for Authors and Reviewers of MIB Documents,” September 2005.) [RFC4181], by C. M. Heard.

---

9. References

TOC

9.1. Normative References

TOC

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC4741]	Enns, R., “NETCONF Configuration Protocol,” RFC 4741, December 2006 (TXT).
[I-D.ietf-netmod-yang]	Bjorklund, M., “YANG - A data modeling language for NETCONF,” draft-ietf-netmod-yang-12 (work in progress), April 2010 (TXT).
[I-D.ietf-netmod-yang-types]	Schoenwaelder, J., “Common YANG Data Types,” draft-ietf-netmod-yang-types-09 (work in progress), April 2010 (TXT).

9.2. Informative References

TOC

[RFC4181]	Heard, C., “Guidelines for Authors and Reviewers of MIB Documents,” BCP 111, RFC 4181, September 2005 (TXT).

Author's Address

TOC

	Andy Bierman
	Netconf Central
	Simi Valley, CA
	USA
Email:	andy@netconfcentral.com