Netmod
Internet-Draft
Intended status: Standards Track
Expires: August 30, 2019

B. Lengyel
Ericsson
B. Claise
Cisco Systems, Inc.

February 26, 2019

YANG Instance Data File Format
draft-ietf-netmod-yang-instance-file-format-02

Abstract

There is a need to document data defined in YANG models when a live YANG server is not available. Data is often needed already at design or implementation time or needed by groups that do not have a live running YANG server available. This document specifies a standard file format for YANG instance data (which follows the syntax and semantic from existing YANG models, re-using the same format as the reply to a <get> operation/request) and decorates it with metadata.

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 https://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 August 30, 2019.

Table of Contents

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 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

Instance Data Set: A named set of data items decorated with metadata that can be used as instance data in a YANG data tree.

Instance Data File: A file containing an instance data set formatted according to the rules described in this document.

Target YANG Module: A YANG module for which the instance data set contains instance data, like ietf-yang-library in the examples.

YANG Instance Data, or just instance data for short, is data that could be stored in a datastore and whose syntax and semantics is defined by YANG models.

2. Introduction

There is a need to document data defined in YANG models when a live YANG server is not available. Data is often needed already at design or implementation time or needed by groups that do not have a live running YANG server available. To facilitate this off-line delivery of data this document specifies a standard format for YANG instance data sets and YANG instance data files.

The following is a list of already implemented and potential use cases.

UC1 Documentation of server capabilities

UC2 Preloading default configuration data

UC3 Documenting Factory Default Settings

UC4 Instance data used as backup

UC5 Storing the configuration of a device, e.g. for archive or audit purposes

UC6 Storing diagnostics data

UC7 Allowing YANG instance data to potentially be carried within other IPC message formats

UC8 Default instance data used as part of a templating solution

UC9 Providing data examples in RFCs or internet drafts

In Appendix C we describe the first three use cases in detail.

There are already many and varied use cases where YANG instance data could be used. We do not want to limit future uses of instance data sets, so specifying how and when to use Yang instance data is out of scope for this document. It is anticipated that other documents

outside the instance data set itself will define specific use cases. Use cases are listed here only to indicate the need for this work.

2.1. High Level Principles

The following is a list of the basic principles of the instance data format:

P1 Two standard formats are based on the XML and the JSON encoding

P2 Re-use existing formats similar to the <get> operation/request

P3 Add metadata about the instance data set

P4 A YANG instance data file shall contain only a single YANG instance data set

P5 A YANG instance data set may contain data for many target YANG modules

P6 Instance data may include configuration data, state data or a mix of the two

P7 Partial data sets are allowed

P8 YANG instance data format may be used for any data for which target YANG module(s) are defined and available to the reader, independent of whether the module is actually implemented by a YANG server

3. Instance Data File Format

A YANG instance data file MUST contain a single instance data set and no additional data.

The instance data set is placed in a top level auxiliary container named "instance-data-set". An instance data set is made up of a header part and content-data. The initial header part carries metadata for the instance data set. It is defined by the ietf-yang-instance-data YANG module. The content-data is all data inside the anydata datanode, this carries the "real data" that we want to document/provide. The syntax and semantics of content-data is defined by the target YANG modules.

Two formats are specified that can be used to represent YANG instance data based on the XML and JSON encoding. Later as other YANG encodings (e.g. CBOR) are defined further instance data formats may be specified.

The content-data part of the XML format SHALL follow the encoding rules defined in [RFC7950] for XML and [RFC7951] for JSON and MUST use UTF-8 character encoding.

It MAY include metadata as defined by [RFC7952].

It MAY include entity-tags and timestamps as defined in [RFC8040]

It MAY include an explicit tag for default values as defined in [RFC6243] and [RFC8040]

It MAY include the origin metadata as specified in [I-D.ietf-netconf-nmda-netconf] and [I-D.ietf-netconf-nmda-restconf]

It MAY include implementation specific metadata. Unknown metadata MUST be ignored by users of YANG instance data, allowing it to be used later for other purposes.

It MAY include implementation specific XML attributes. Unknown attributes MUST be ignored by users of YANG instance data, allowing them to be used later for other purposes.

The content-data part will be very similar to the result returned for a NETCONF <get-data> or for a RESTCONF get operation.

The content-data part MUST conform to the corresponding target YANG Modules. A single instance data set MAY contain data for any number of target YANG modules; if needed it MAY carry the complete configuration and state data set for a YANG server. Default values SHOULD NOT be included.

Config=true and config=false data MAY be mixed in the instance data file.

Instance data files MAY contain partial data sets. This means mandatory, min-elements or require-instance=true constrains MAY be violated.

The name of the file SHALL be of the form:

instance-data-set-name ['@' revision-date] '.filetype'

E.g. acme-router-modules@2018-01-25.xml

The revision date is optional. ".filetype" SHALL be ".json" or ".xml" according to the format used.

Metadata, information about the data set itself SHALL be included in the instance data set. This data will be children of the top level instance-data-set container as defined in the ietf-instance-data YANG module. Metadata MUST include:

- name of the instance data set

Metadata SHOULD include:

- target-ptr: A pointer to the list of target YANG modules their revision, supported features and deviations.
- An inline definition of target-modules, when the INLINE method is used for the target-ptr
- Description of the instance data set. The description SHOULD contain information whether and how the data can change during the lifetime of the YANG server.

Metadata MAY include:

- Organization responsible for the instance data set
- Contact information
- Information about the datastore associated with the instance data set e.g. the datastore from where the data was read or the datastore where the data could be loaded or the datastore which is being documented. This information is optional, as often a single datastore can not be specified.
- Revision date of the instance data set. If both this date and and the date in the instance data file name are present they MUST have the same value.
- Timestamp: The date and time when the instance data set was last modified.
- It is anticipated that different organizations will have the need to augment the metadata with various other data nodes.

3.1. Specifying the Target YANG Modules: target-ptr

To properly understand and use an instance data set the user needs to know the list of target YANG modules their revision, supported features and deviations. The metadata "target-ptr" is used to specify the YANG target module list. One of the following options SHOULD be used:

INLINE method: Include the needed information as part of instance data set as defined by e.g. ietf-yang-library

URI method: Include a URI that points to the target module set. (if you don't want to repeat the info again and again)

EXTERNAL Method: Do not include the target-ptr as the target YANG module set is already known, or the information is available through external documents.

Additional methods e.g. a YANG-package based solution may be added later.

Note, the specified target YANG modules only indicate the set of modules that were used to define this YANG instance data set. Sometimes instance data may be used for a YANG server supporting a different YANG module set e.g. for "UC2 Preloading Data" the instance data set may not be updated every time the YANG modules on the YANG server are updated, an unchanged instance data set may still be usable. Whether the instance data set is usable for a possibly different real-life target YANG module set depends on many factors including the compatibility between the specified target and the real-life target YANG module set (considering modules, revisions, features, deviations), the scope of the instance data, etc.

3.1.1. INLINE Method

One or more inline-target-spec elements SHALL be specified. The first one specifies ietf-yang-library or a similar YANG module listing target YANG modules with their name, revision-date, supported-features and deviations. Deviations or unsupported features MUST NOT remove any of the above data from the module. Using ietf-yang-library MUST be supported.

E.g. ietf-yang-library@2016-06-21.yang

As some versions of ietf-yang-library MAY contain different module-sets for different datastores, if multiple module-sets are defined, the instance data set's meta-data MUST contain the datastore information and instance data for the ietf-yang library MUST also contain information specifying the module-set for the relevant datastore.

Subsequent inline-target-spec elements MAY specify YANG modules augmenting the first module with useful data (e.g. a semantic version).

When using the inline method a 'target-modules' element MUST be present. This SHALL contain instance data corresponding to the YANG modules specified in the inline-target-spec elements specifying the set of target YANG modules for this instance-data-set.

3.1.2. URI Method

A target-uri element SHALL contain a URI that references another YANG instance data file. The current instance data file will use the same set of target YANG modules, revisions, supported features and deviations as the referenced YANG instance data file.

The referenced instance data file will usually contain data only for ietf-yang-library to specify the target YANG modules for the original instance data file.

The URI method is advantageous when the user wants to avoid the overhead of specifying the target YANG modules in the instance data file: E.g. In Use Case 6, when the system creates a diagnostic file every 10 minutes to document the state of the YANG server.

The referenced YANG instance data file might use the in-line method or might use the URI method to reference further instance data file(s). However at the end of this reference chain there MUST be an instance data file using the in-line method.

If a referenced instance data file is not available the revision data, supported features and deviations for the target YANG modules are unknown.

3.2. Examples

The following example is based on "UC1, Documenting Server Capabilities". It provides (a shortened) list of supported YANG modules and Netconf capabilities for a YANG server. It uses the inline method for the target-ptr.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-modules</name>
  <inline-target-spec>
    ietf-yang-library@2016-06-21.yang
  </inline-target-spec>
  <target-modules>
    <module-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module>
        <name>ietf-yang-library</name>
```

```
        <revision>2016-06-21</revision>
      </module>
      <module>
        <name>ietf-netconf-monitoring</name>
        <revision>2010-10-04</revision>
      </module>
    </module-state>
  </target-modules>
  <revision>
    <date>2108-01-25</date>
    <description>Initial version</description>
  </revision>
  <description>Defines the minimal set of modules that any acme-router
      will contain.</description>
  <contact>info@acme.com</contact>
  <content-data>
    <!-- The example lists only 4 modules, but it could list the
      full set of supported modules for a YANG server, potentially many
      dozens of modules -->
    <module-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module>
        <name>ietf-yang-library</name>
        <revision>2016-06-21</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-yang-library
        </namespace>
        <conformance-type>implement</conformance-type>
      </module>
      <module>
        <name>ietf-system</name>
        <revision>2014-08-06</revision>
        <namespace>urn:ietf:params:xml:ns:yang:ietf-system</namespace>
        <feature>sys:authentication</feature>
        <feature>sys:local-users</feature>
        <deviation>
          <name>acme-system-ext</name>
          <revision>2018-08-06</revision>
        </deviation>
        <conformance-type>implement</conformance-type>
      </module>
      <module>
        <name>ietf-yang-types</name>
        <revision>2013-07-15</revision>
        <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-types
          </namespace>
        <conformance-type>import</conformance-type>
      </module>
      <module>
```

```
        <name>acme-system-ext</name>
        <revision>2018-08-06</revision>
        <namespace>urn:rdns:acme.com:oammodel:acme-system-ext
          </namespace>
        <conformance-type>implement</conformance-type>
      </module>
    </module-state>
    <netconf-state>
      <capabilities>
        <capability>
          urn:ietf:params:netconf:capability:validate:1.1
        </capability>
      </capabilities>
    </netconf-state>
  </content-data>
</instance-data-set>
```

Figure 1: XML Instance Data Set - Use case 1, Documenting server capabilities

The following example is based on "UC2, Preloading Default Configuration". It provides a (shortened) default rule set for a read-only operator role. It uses the inline method for the target-ptr.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>read-only-acm-rules</name>
  <inline-target-spec>ietf-yang-library@2016-06-21.yang
    </inline-target-spec>
  <target-modules>
    <module-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module>
        <name>ietf-netconf-acm</name>
        <revision>2012-02-22</revision>
      </module>
    </module-state>
  </target-modules>
  <revision>
    <date>2018-01-25</date>
    <description>Initial version</description>
  </revision>
  <description>Access control rules for a read-only role.</description>
  <contact>info@acme.com</contact>
  <content-data>
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
      <enable-nacm>true</enable-nacm>
      <read-default>deny</read-default>
      <exec-default>deny</exec-default>
      <rule-list>
        <name>read-only-role</name>
        <group>read-only-group</group>
        <rule>
          <name>read-all</name>
          <module-name>*</module-name>
          <access-operation>read</access-operation>
          <action>permit</action>
        </rule>
      </rule-list>
    </nacm>
  </content-data>
</instance-data-set>
```

Figure 2: XML Instance Data Set - Use case 2, Preloading access control data

The following example is based on UC6 Storing diagnostics data. An instance data set is produced by the YANG server every 15 minutes that contains statistics about NETCONF. As a new set is produced periodically multiple times a day a revision-date would be useless; instead a timestamp is included.

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "acme-router-netconf-diagnostics",
    "target-uri": "file:///acme-netconf-diagnostics-yanglib.json",
    "timestamp": "2018-01-25T17:00:38Z",
    "description":
      "Netconf statistics",
    "content-data": {
      "ietf-netconf-monitoring:netconf-state": {
        "statistics": {
          "netconf-start-time ": "2018-12-05T17:45:00Z",
          "in-bad-hellos ": "32",
          "in-sessions ": "397",
          "dropped-sessions ": "87",
          "in-rpcs ": "8711",
          "in-bad-rpcs ": "408",
          "out-rpc-errors ": "408",
          "out-notifications": "39007"
        }
      }
    }
  }
}
```

Figure 3: JSON Instance Data File example - UC6 Storing diagnostics data

4. Data Life cycle

Data defined or documented in YANG instance data sets may be used for preloading a YANG server with this data, but the server may populate the data without using the actual file in which case the instance data file is only used as documentation.

While such data will usually not change, data documented by instance data sets MAY be changed by the YANG server itself or by management operations. It is out of scope for this document to specify a method to prevent this. Whether such data changes and if so, when and how, SHOULD be described either in the instance data set's description statement or in some other implementation specific manner.

YANG instance data is a snap-shot of information at a specific point of time. If the data changes afterwards this is not represented in the instance data set anymore, the valid values can be retrieved in run-time via NETCONF/RESTCONF.

Notifications about the change of data documented by instance data sets may be supplied by e.g. the Yang-Push mechanism, but it is out of scope for this document.

5. Delivery of Instance Data

Instance data sets that are produced as a result of some sort of specification or design effort SHOULD be available without the need for a live YANG server e.g. via download from the vendor's website, or in any other way product documentation is distributed.

Other instance data sets may be read from or produced by the YANG server itself e.g. UC6 documenting diagnostic data.

6. Backwards Compatibility

The concept of backwards compatibility and what changes are backwards compatible are not defined for instance data sets as it is highly dependent on the specific use case and the target YANG model. However as instance data does use the concept of managed entities identified by key values the following guidelines are provided:

- For list entries representing the same managed entity as previously key values SHOULD NOT be changed.
- The meaning of list entries, representing the same managed entity as previously, SHOULD NOT be changed e.g. redefining an alarm-type but not changing its alarm-type-id should be avoided.
- Keys for previously removed list entries SHOULD NOT be reused if they represent a different meaning.

7. YANG Model

```
<CODE BEGINS> file "ietf-yang-instance-data.yang"
module ietf-yang-instance-data {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data";
  prefix yid ;

  import ietf-yang-data-ext { prefix yd; }
  import ietf-datastores { prefix ds; }
  import ietf-inet-types { prefix inet; }
  import ietf-yang-types { prefix yang; }

  organization  "IETF NETMOD Working Group";
  contact
```

```
   "WG Web:   <https://datatracker.ietf.org/wg/netmodf/>
    WG List:  <mailto:netmod@ietf.org>

    Author:  Balazs Lengyel
       <mailto:balazs.lengyel@ericsson.com>";

  description "The module defines the structure and content of YANG
       instance data sets.";

  revision 2019-02-20 {
    description  "Initial revision.";
    reference "RFC XXXX: YANG Instance Data Format";
  }

  yd:yang-data instance-data-format {
    container instance-data-set {
      description "Auxiliary container to carry meta-data for
        the complete instance data set.";

      leaf name {
        type string;
        mandatory true;
        description "Name of the YANG instance data set.";
      }

      choice target-ptr {
        description "A pointer to the list of target YANG modules
          their revisions, supported features and deviations.";

        case inline {
          leaf-list inline-target-spec {
            type string {
              pattern '.+@\d{4}-\d{2}-\d{2}\.yang';
            }
            min-elements 1;
            ordered-by user;
            description
              "Indicates that target modules are specified inline.
              Each value MUST be a YANG Module name including the
              revision-date as defined for YANG file names in RFC7950.

              E.g. ietf-yang-library@2016-06-21.yang

              The first item is either ietf-yang-library or some other
              YANG module that contains a list of YANG modules with
              their name, revision-date, supported-features and
              deviations.
              As some versions of ietf-yang-library MAY contain
```

```
            different module-sets for different datastores, if
            multiple module-sets are defined, the instance data set's
            meta-data MUST contain the datastore information and
            instance data for the ietf-yang-library MUST also contain
            information specifying the module-set for the relevant
            datastore.

            Subsequent items MAY specify YANG modules augmenting the
            first module with useful data (e.g. a semantic version).";
        }
        anydata target-modules {
          mandatory true;
          description "Instance data corresponding to the YANG modules
            specified in the inline-target-spec nodes defining the set
            of target YANG modules for this instance-data-set.";
        }
      }

      case uri {
        leaf target-uri {
          type inet:uri;
          description
            "A reference to another YANG instance data file.
            This instance data file will use the same set of target
            YANG modules, revisions, supported features and deviations
            as the referenced YANG instance data file.";
        }
      }
    }

    leaf description { type string; }

    leaf contact {
      type string;
      description "Contact information for the person or
        organization to whom queries concerning this
        instance data set should be sent.";
    }

    leaf organization {
      type string;
      description "Organization responsible for the instance
        data set.";
    }

    leaf datastore {
      type ds:datastore-ref;
      description  "The identity of the datastore with which the
```

```
          instance data set is associated. If a single specific
          datastore can not be specified, the leaf MUST be absent.

          If this leaf is absent, then the datastore to which the
          instance data belongs is undefined.";
      }

      list revision {
        key date;
        description "Instance data sets that are produced as
          a result of some sort of specification or design effort
          SHOULD have at least one revision entry.  For every
          published editorial change, a new one SHOULD be added
          in front of the revisions sequence so that all
          revisions are in reverse chronological order.

          For instance data sets that are read from
          or produced by the YANG server or otherwise
          subject to frequent updates or changes, revision
          SHOULD NOT be present";

        leaf date {
          type string {
            pattern '\d{4}-\d{2}-\d{2}';
          }
          description "Specifies the date the instance data set
              was last modified. Formatted as YYYY-MM-DD";
        }

        leaf description { type string; }
      }

      leaf timestamp {
        type yang:date-and-time;
        description "The date and time when the instance data set
          was last modified.

          For instance data sets that are read from or produced
          by the YANG server or otherwise subject to frequent
          updates or changes, timestamp SHOULD be present";
      }

      anydata content-data {
        mandatory true;
        description "Contains the real instance data.
            The data MUST conform to the relevant YANG Modules.";
      }
    }
```

```
  }
}
<CODE ENDS>
```

8. Security Considerations

Depending on the nature of the instance data, instance data files MAY need to be handled in a secure way. The same type of handling should be applied, that would be needed for the result of a <get> operation returning the same data.

9. IANA Considerations

This document registers one URI and one YANG module.

9.1. URI Registration

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

URI: urn:ietf:params:xml:ns:yang:ietf-yang-instance-data

Registrant Contact: The IESG.

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

9.2. YANG Module Name Registration

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

```
name: ietf-yang-instance-data
namespace: urn:ietf:params:xml:ns:yang:ietf-yang-instance-data
prefix: yid
reference: RFC XXXX
```

10. Acknowledgments

For their valuable comments, discussions, and feedback, we wish to acknowledge Andy Bierman, Juergen Schoenwaelder, Rob Wilton, Joe Clark, Martin Bjorklund, Ladislav Lhotka, Qin Wu and other members of the Netmod WG.

11. References

11.1. Normative References

[I-D.ietf-netconf-nmda-netconf]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "NETCONF Extensions to Support the Network Management Datastore Architecture", draft-ietf-netconf-nmda-netconf-08 (work in progress), October 2018.

[I-D.ietf-netconf-nmda-restconf]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "RESTCONF Extensions to Support the Network Management Datastore Architecture", draft-ietf-netconf-nmda-restconf-05 (work in progress), October 2018.

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Extensions", draft-ietf-netmod-yang-data-ext-01 (work in progress), March 2018.

[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>.

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <https://www.rfc-editor.org/info/rfc6243>.

[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>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[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>.

11.2. Informative References

[I-D.ietf-ccamp-alarm-module]
Vallin, S. and M. Bjorklund, "YANG Alarm Module", draft-ietf-ccamp-alarm-module-07 (work in progress), January 2019.

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-22 (work in progress), February 2019.

[I-D.wu-netconf-restconf-factory-restore]
Wu, Q., Lengyel, B., and Y. Niu, "Factory default Setting", draft-wu-netconf-restconf-factory-restore-03 (work in progress), October 2018.

[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>.

[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>.

Appendix A. Open Issues

- Augmenting metadata must be possible. As of now it looks like yang-data-ext will solve that. If not, define instance data as regular YANG instead of yd:yang-data.

Appendix B. Changes between revisions

v01 - v02

- Removed design time from terminology

- Defined the format of the content-data part by referencing various RFCs and drafts instead of the result of the get-data and get operations.

- Changed target-ptr to a choice

- Inline target-ptr may include augmenting modules and alternatives to ietf-yang-library

- Moved list of target modules into a separate <target-modules> element.

- Added backwards compatibility considerations

v00 - v01

- Added the target-ptr metadata with 3 methods

- Added timestamp metadata

- Removed usage of dedicated .yid file extension

- Added list of use cases

- Added list of principles

- Updated examples

- Moved detailed use case descriptions to appendix

v05 - v00-netmod

- New name for the draft following Netmod workgroup adoption. No other changes

v04 - v05

- Changed title and introduction to clarify that this draft is only about the file format and documenting server capabilities is just a use case.

- Added reference to draft-wu-netconf-restconf-factory-restore

- Added new open issues.

v03 - v04

- Updated changelog for v02-v03

v02 - v03

- Updated the document according to comments received at IETF102
- Added parameter to specify datastore
- Rearranged chapters
- Added new use case: Documenting Factory Default Settings
- Added "Target YANG Module" to terminology
- Clarified that instance data is a snapshot valid at the time of creation, so it does not contain any later changes.
- Removed topics from Open Issues according to comments received at IETF102

v01 - v02

- The recommendation to document server capabilities was changed to be just the primary use-case. (Merged chapter 4 into the use case chapter.)
- Stated that RFC7950/7951 encoding must be followed which also defines (dis)allowed whitespace rules.
- Added UTF-8 encoding as it is not specified in t950 for instance data
- added XML declaration

v00 - v01

- Redefined using yang-data-ext
- Moved metadata into ordinary leafs/leaf-lists

Appendix C. Detailed Use Cases - Non-Normative

C.1. Use Cases

We present a number of use cases were YANG instance data is needed.

C.1.1. Use Case 1: Early Documentation of Server Capabilities

A YANG server has a number of server-capabilities that are defined in YANG modules and can be retrieved from the server using protocols like NETCONF or RESTCONF. YANG server capabilities include

- data defined in ietf-yang-library: YANG modules, submodules, features, deviations, schema-mounts, datastores supported ([I-D.ietf-netconf-rfc7895bis])
- alarms supported ([I-D.ietf-ccamp-alarm-module])
- data nodes, subtrees that support or do not support on-change notifications ([I-D.ietf-netconf-yang-push])
- netconf-capabilities in ietf-netconf-monitoring

While it is good practice to allow a client to query these capabilities from the live YANG server, that is often not possible.

Often when a network node is released an associated NMS (network management system) is also released with it. The NMS depends on the capabilities of the YANG server. During NMS implementation information about server capabilities is needed. If the information is not available early in some off-line document, but only as instance data from the live network node, the NMS implementation will be delayed, because it has to wait for the network node to be ready. Also assuming that all NMS implementors will have a correctly configured network node available to retrieve data from, is a very expensive proposition. (An NMS may handle dozens of node types.)

Network operators often build their own home-grown NMS systems that needs to be integrated with a vendor's network node. The operator needs to know the network node's server capabilities in order to do this. Moreover the network operator's decision to buy a vendor's product may even be influenced by the network node's OAM feature set documented as the Yang server's capabilities.

Beside NMS implementors, system integrators and many others also need the same information early. Examples could be model driven testing, generating documentation, etc.

Most server-capabilities are relatively stable and change only during upgrade or due to licensing or addition or removal of HW. They are usually defined by a vendor at design time, before the product is released. It feasible and advantageous to define/document them early e.g. in a YANG instance data File.

It is anticipated that a separate IETF document will define in detail how and which set of server capabilities should be documented.

C.1.2. Use Case 2: Preloading Data

There are parts of the configuration that must be fully configurable by the operator, however for which often a simple default configuration will be sufficient.

One example is access control groups/roles and related rules. While a sophisticated operator may define dozens of different groups often a basic (read-only operator, read-write system administrator, security-administrator) triplet will be enough. Vendors will often provide such default configuration data to make device configuration easier for an operator.

Defining Access control data is a complex task. To help the device vendor pre-defines a set of default groups (/nacm:nacm/groups) and rules for these groups to access specific parts of common models (/nacm:nacm/rule-list/rule).

YANG instance data files are used to document and/or preload the default configuration.

C.1.3. Use Case 3: Documenting Factory Default Settings

Nearly every YANG server has a factory default configuration. If the system is really badly misconfigured or if the current configuration is to be abandoned the system can be reset to this default.

In Netconf the <delete-config> operation can already be used to reset the startup datastore. There are ongoing efforts to introduce a new, more generic reset-datastore operation for the same purpose [I-D.wu-netconf-restconf-factory-restore]

The operator currently has no way to know what the default configuration actually contains. YANG instance data can be used to document the factory default configuration.

Authors' Addresses

Balazs Lengyel
Ericsson
Magyar Tudosok korutja 11
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

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

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