Netmod
Internet-Draft
Intended status: Standards Track
Expires: February 13, 2020

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

August 12, 2019

YANG Instance Data File Format
draft-ietf-netmod-yang-instance-file-format-04

Abstract

There is a need to document data defined in YANG models when a live 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 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 February 13, 2020.

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.

Content-schema: A set of YANG modules with their revision,suupported features and deviations for which the instance data set contains instance data

Content defining Yang module(s): YANG module(s) that make up the content-schema

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.

The term Server is used as defined in [RFC8342]

2. Introduction

There is a need to document data defined in YANG models when a live 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 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 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 will define specific use cases. Use cases are listed here only to indicate the need for this work.

2.1. 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 response to a <get> operation/request

P3 Add metadata about the instance data set (Section 3, Paragraph 9)

P4 A YANG instance data set may contain data for many YANG modules

P5 Instance data may include configuration data, state data or a mix of the two

P6 Partial data sets are allowed

P7 YANG instance data format may be used for any data for which YANG module(s) are defined and available to the reader, independent of whether the module is actually implemented by a server

3. Instance Data File Format

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

The format of the instance data set is defined by the ietf-yang-instance-data YANG module. It is made up of a header part and content-data. The header part carries metadata for the instance data set. The content-data, defined as an anydata data node, carries the "real data" that we want to document/provide. The syntax and semantics of content-data is defined by the content-schema.

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

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

metadata as defined by [RFC7952].

a default attribute as defined in [RFC6243] section 6. and in [RFC8040] section 4.8.9.

origin metadata as specified in [RFC8526] and [RFC8527]

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

in the XML format 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 content-schema. An instance data set MAY contain data for any number of YANG modules; if needed it MAY carry the complete configuration and state data set for a 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, require-instance=true, must and when constrains MAY be violated.

The name of the instance data file SHOULD take one of the following two forms:

If revision information inside the data set is present

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

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

If the leaf name is present in the instance data header this MUST be used. Revision-date MUST be set to the latest revision date inside the instance data set.

If timestamp information inside the data set is present

* instance-data-set-name ['@' timestamp] '.filetype'

* E.g. acme-router-modules@2018-01-25T15_06_34_3+01_00.json

If the leaf name is present in the instance data header this MUST be used. If the leaf timestamp is present in the instance data header this MUST be used; the semicolons and the decimal point if present shall be replaced by underscores.

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

Metadata, information about the data set itself SHOULD be included in the instance data set. Some metadata items are defined in the YANG module ietf-yang-instance-data, but other items MAY also be used. Metadata SHOULD include:

o Name of the data set

o Content schema specification

o Description of the instance data set. The description SHOULD contain information whether and how the data can change during the lifetime of the server.

3.1. Specifying the Content Schema

To properly understand and use an instance data set the user needs to know the content-schema. One of the following methods SHOULD be used:

Inline method: Include the needed information as part of the instance data set.

Simplified-Inline method: Include the needed information as part of the instance data set; short specification.

URI method: Include a URI that references another YANG instance data file. This instance data file will use the same content-schema as the referenced YANG instance data file. (if you don't want to repeat the info again and again)

EXTERNAL Method: Do not include the content-schema as it 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 content-schema only indicates the set of modules that were used to define this YANG instance data set. Sometimes instance data may be used for a 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 server are updated) Whether the instance data set is usable for a possibly different real-life YANG module set depends on many factors including the compatibility between the specified and the real-life 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 define YANG module(s) used to specify the content defining YANG modules.

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

The anydata inline-content-schema carries instance data (conforming to the inline-target-spec modules) that actually specifies the content defining YANG modules including revision, supported features, deviations and any relevant additional data (e.g. version labels)

3.1.2. Simplified-Inline Method

The instance data set contains a list of content defining YANG modules including the revision date for each. Usage of this method implies that the modules are used without any deviations and with all features supported.

3.1.3. URI Method

A schema-uri leaf SHALL contain a URI that references another YANG instance data file. The current instance data file will use the same content schema as the referenced file.

The referenced instance data file MAY have no content-data if it is used solely for specifying the content-schema. The referenced YANG instance data file might use the INLINE 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 INLINE method.

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

The URI method is advantageous when the user wants to avoid the overhead of specifying the content-schema in each instance data file: E.g. In Use Case 6, when the system creates a diagnostic file every minute to document the state of the server.

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 server. It uses the inline method to specify the content-schema.

```
<?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-spec>
    ietf-yang-library@2016-06-21.yang
  </inline-spec>
  <inline-content-schema>
    <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>
  </inline-content-schema>
  <revision>
    <date>1956-10-23</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 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 specifying the content-schema.

```
<?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-spec>ietf-yang-library@2019-01-04.yang</inline-spec>
  <inline-content-schema>
    <yang-library xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module-set>
        <name>all</name>
        <module>
          <name>ietf-netconf-acm</name>
          <revision>2012-02-22</revision>
        </module>
      </module-set>
    </yang-library>
  </inline-content-schema>
  <revision>
    <date>1776-07-04</date>
    <description>Initial version</description>
  </revision>
  <description>Access control rules for a read-only role.</description>
  <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 server every 15 minutes that contains statistics about NETCONF. As a new set is produced periodically many 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",
    "schema-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

In UC2 "Preloading default configuration data" the loaded data may be changed later e.g. by management operations. In UC6 "Storing Diagnostics data" the diagnostics values may change on device every second.

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 or received e.g. in Yang-Push notifications.

Whether the instance 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.

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 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 content-schema.

For instance data that is the result of a design or specification activity some changes that may be good to avoid are listed. YANG uses the concept of managed entities identified by key values; if the connection between the represented entity and the key value is not preserved during an update this may lead to problems.

- If the key value of a list entry that represents the same managed entity as before is changed, the user may mistakenly identify the list entry as new.

- If the meaning of a list entry is changed, but the key values are not (e.g. redefining an alarm-type but not changing its alarm-type-id) the change may not be noticed.

- If the key value of a previously removed list entry is reused for a different entity, the change may be mis-interpreted as reintroducing the previous entity.

7. Yang Instance Data Model

7.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model.

```
   module: ietf-yang-instance-data
     structure instance-data-set:
        +--rw name?                          string
        +--rw (content-schema-spec)?
        |  +--:(simplified-inline)
        |     +--rw module*                  string
        |  +--:(inline)
        |  |  +--rw inline-spec*             string
        |  |  +--rw inline-content-schema    <anydata>
        |  +--:(uri)
        |     +--rw schema-uri?            inet:uri
        +--rw description?          string
        +--rw contact?              string
        +--rw organization?         string
        +--rw datastore?            ds:datastore-ref
        +--rw revision* [date]
        |  +--rw date           string
        |  +--rw description?   string
        +--rw timestamp?            yang:date-and-time
        +--rw content-data?         <anydata>
```

7.2. YANG Model

```
  <CODE BEGINS> file "ietf-yang-instance-data@2019-07-04.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-structure-ext { prefix sx; }
    import ietf-datastores { prefix ds; }
    import ietf-inet-types { prefix inet; }
    import ietf-yang-types { prefix yang; }
    import ietf-yang-metadata { prefix "md"; }

    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.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
```

```
       'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
       'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
       are to be interpreted as described in BCP 14 (RFC 2119)
       (RFC 8174) when, and only when, they appear in all
       capitals, as shown here.

       Copyright (c) 2019 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

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

    revision 2019-07-04 {
      description  "Initial revision.";
      reference "RFC XXXX: YANG Instance Data Format";
    }

    sx:structure instance-data-set {
      description "A data structure to define a format for a
        YANG instance data set.Consists of meta-data about
        the instance data set and the real content-data.";

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

      choice content-schema-spec {
        description "Specification of the content-schema";

        case simplified-inline {
          leaf-list module  {
            type string {
              pattern '.+@\d{4}-\d{2}-\d{2}\.yang';
             }
             description "The list of content defining YANG
               modules including the revision date for each.
               Usage of this leaf-list implies the modules are
               used without any deviations and with all features
               supported.";
          }
```

```
        }
        case inline {
          leaf-list inline-spec {
            type string {
              pattern '.+@\d{4}-\d{2}-\d{2}\.yang';
            }
            min-elements 1;
            ordered-by user;
            description
              "Indicates that content defining Yang 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 included, 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 version label).";
          }
          anydata inline-content-schema {
            mandatory true;
            description "Instance data corresponding to the YANG modules
              specified in the inline-spec nodes defining the set
              of content defining Yang YANG modules for this
              instance-data-set.";
          }
        }

        case uri {
          leaf schema-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-list description {
      type string;
      description "Description of the instance data set.";
    }

    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 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.
        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 a 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;
          description
            "Description of this revision of the instance data set.";
        }
      }

      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 a server or otherwise subject to frequent
          updates or changes, timestamp SHOULD be present";
      }

      anydata content-data {
        description "Contains the real instance data.
          The data MUST conform to the relevant YANG Modules specified
          either in the content-schema-spec or in some other
          implementation specific manner.";
      }
    }
  }
  <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 Clarke, Kent Watsen Martin Bjorklund, Ladislav Lhotka, Qin Wu and other members of the Netmod WG.

11. References

11.1. Normative References

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Structure Extensions", draft-ietf-netmod-yang-data-ext-04 (work in progress), July 2019.

[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>.

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

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

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

[RFC8527] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "RESTCONF Extensions to Support the Network Management Datastore Architecture", RFC 8527, DOI 10.17487/RFC8527, March 2019, <https://www.rfc-editor.org/info/rfc8527>.

11.2. Informative References

[I-D.ietf-ccamp-alarm-module] Vallin, S. and M. Bjorklund, "YANG Alarm Module", draft-ietf-ccamp-alarm-module-09 (work in progress), April 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. and E. Voit, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-25 (work in progress), May 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

- -

Appendix B. Changes between revisions

v03 - v04

- removed entity-tag and last-modified timestamp
- Added simplified-inline method of content-schema specification

v02 - v03

- target renamed to "content-schema" and "content defining Yang module(s)"
- Made name of instance data set optional
- Updated according to draft-ietf-netmod-yang-data-ext-03
- Clarified that entity-tag and last-modified timestamp are encoded as metadata. While they contain useful data, the HTTP-header based encoding from Restconf is not suitable.

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

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 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. 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 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 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 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 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