

# Validating anydata in YANG Library context

draft-aelhassany-anydata-validation

ahmed.elhassany@swisscom.com

21. March 2024

# Context

- RFC 7950: The YANG 1.1 Data Modeling Language

The "anydata" statement is used to represent an unknown set of nodes that can be modeled with YANG, except anyxml, but for which **the data model is not known at module design time**. It is possible, though not required, for the data model for anydata content to become **known through protocol signaling or other means that are outside the scope of this document**.

# Where anydata is currently used?

incomplete list

- RFC 8342: `ietf-netconf-nmda`
- RFC 9144: `ietf-nmda-compare`
- RFC 8040: `ietf-restconf`
- RFC 8639: `ietf-subscribed-notifications`
- RFC 9195: `ietf-yang-instance-data`
- RFC 8072: `ietf-yang-patch`
- RFC 8072: `ietf-yang-push`
- RFC 8532: `ietf-connectionless-oam` (uses yang mount)
- RFC 8791: any YANG data structure is encoded the same way as anydata node.

# Problem statement

- How can we validate the subtree of anydata node?

```
notifications:
  +---n push-update
  |   +--ro id?          sn:subscription-id
  |   +--ro datastore-contents? <anydata>
```

```
{
  "ietf-yang-push:push-update": {
    "id": 89,
    "datastore-contents": {
      "ietf-interfaces:interfaces": {
        "interface": [
          {
            "name": "eth0",
            "oper-status": "down"
          }
        ]
      }
    }
  }
}
```

# Assumptions

1. Is a subtree of anydata node is always specified by a YANG model?
  - The spirit of RFC7950 assumes yes, but it's not explicit in a "MUST" statement.
2. Is a subtree of anydata is complete subtree of a valid YANG model?
  - Not always! subtree and xpath filters gives powerful semantic to only select certain nodes of a subtree in a datastore.

# YANG Data encodings

YANG Data JSON & XML encodings always contain full namespace information to identify the relevant model.

```
{
  "ietf-yang-push:push-update": {
    "id": 89,
    "datastore-contents": {
      "ietf-interfaces:interfaces": {
        "interface": [
          {
            "name": "eth0",
            "oper-status": "down"
          }
        ]
      }
    }
  }
}
```

```
<push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
  <id>89</id>
  <datastore-contents>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">eth0</name>
        <oper-status>down</oper-status>
      </interface>
    </interfaces>
  </datastore-contents>
</push-update>
```

# RFC 8525 YANG Library

YANG Library provides information about the YANG modules, datastores, and datastore schemas used by a network management server.

```
{
  "ietf-yang-library:yang-library": {
    "module-set": [
      {
        "name": "complete",
        "module": [
          {
            "name": "ietf-interfaces",
            "revision": "2018-02-20",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-interfaces",
            "location": ["file://ietf-interfaces@2018-02-20.yang"],
            "feature": [
              "arbitrary-names",
              "pre-provisioning",
              "if-mib"
            ]
          }
        ]
      },
      ...
    ]
  }
}
```

# YANG Library look up

- The namespace of the encoded data nodes under anydata can be looked up in a YANG Library context.

```
{
  "ietf-yang-library:yang-library": {
    "module-set": [
      {
        "name": "complete",
        "module": [
          {
            "name": "yang",
            "revision": "2022-06-16",
            "namespace": "urn:ietf:params:xml:ns:yang:1"
          },
          {
            "name": "ietf-interfaces",
            "revision": "2018-02-20",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-interfaces",
            "location": ["file://ietf-interfaces@2018-02-20.yang"],
            "feature": [
              "arbitrary-names",
              "pre-provisioning",
              "if-mib"
            ]
          }
        ]
      },
      ...
    ]
  }
}
```



# Implementation

- Current libyang implementation disables strict parsing while in anydata subtree. Implementing this draft would require to change this behavior with an optional flag and use strict validation always.

```
diff --git a/src/parser_xml.c b/src/parser_xml.c
index 5d97c8e49..6938d3712 100644
--- a/src/parser_xml.c
+++ b/src/parser_xml.c
@@ -931,7 +931,7 @@ lydxml_subtree_any(struct lyd_xml_ctx *lydctx, const struct lysc_node
 *snode, co
     LY_CHECK_ERR_GOTO(r, rc = r, cleanup);

     /* update options so that generic data can be parsed */
-    lydctx->parse_opts &= ~LYD_PARSE_STRICT;
+    //lydctx->parse_opts &= ~LYD_PARSE_STRICT;
     lydctx->parse_opts |= LYD_PARSE_OPAQ | (ext ? LYD_PARSE_ONLY : 0);
     lydctx->int_opts |= LYD_INTOPT_ANY | LYD_INTOPT_WITH_SIBLINGS;
```

# Open Issues

- Complete vs incomplete data tree?
  - Values in anydata could be generated by XPath or subtree filters. The resulting subtree could be incomplete. How should that data be validated
- YANG Library could contain multiple module-sets, which one to choose?