YANG Validation Examples

Scott Mansfield
Ericsson

23 September 2021

802.1 September 2021 Interim

v2

YANG Validation

- pyang
  - https://1.ieee802.org/yangsters/yangsters-guidelines/yangsters-faq/#Tool_pyang
- yanglint
  - https://1.ieee802.org/yangsters/yangsters-guidelines/yangsters-faq/#Tool_yanglint_libyang
- on-line
  - https://1.ieee802.org/yangsters/yangsters-guidelines/yangsters-faq/#On-line_YANG_Validation_Tool

- This presentation has a couple of examples of using yanglint running on linux

Basic workflow for yanglint

- Setup your environment (personal preference regarding OS)
  - Linux OS
  - git (repository)
  - expect (scripting)
  - libyang (includes yanglint)
- Create a directory and copy in all yang files (including all imported yang files) locally
- Configure your expect input file (I call them “.in” files)
- Configure your input instance information in the format of your choice (xml or json are popular)
- Run the expect script
- Admire your results

yanglint

- I use pyang for syntax checking
- yanglint does even more syntax and validation checking even if you don’t use the instance data
  - So if you just run “yanglint *.yang” in a directory with all the yang files you care about, extensive checking is done
  - If you want to exercise the XPATH/must statements then you need to create some instance data to test

Simple Example

- A small contrived example to show a successful run and an unsuccessful run

```
YANG file: minerals.yang
Here is the tree:
module: minerals
  +--rw mineral-db
  |  +--rw mineral* [mineral-name]
  |     +--rw mineral-name          string
  |     +--rw physical-properties?  string
  |     +--rw hardness?             decimal64
  +--rw samples
    +--rw sample* [sample-id]
       +--rw sample-id      string
       +--rw streak?        string
       +--rw cleavage?      string
       +--rw mineral-ref?   -> /mineral-db/mineral/mineral-name
```

Successful Run

```
scott@Cosima: ~/rwsl/validate/small
scott@Cosima:~/rwsl/validate/small$ expect minera
spawn /usr/local/bin/yanglint
> load minerals
> data -t config -f json minerals.xml
{
  "minerals:mineral-db": {
    "mineral": [
      {
        "mineral-name": "Talc",
        "physical-properties": "White, Green, Gra
        "hardness": "1.0"
      },
      {
        "mineral-name": "Gypsum",
        "physical-properties": "Colorless or Whit
        "hardness": "2.0"
      }
    ]
  },
  "minerals:samples": {
    "sample": [
      {
        "sample-id": "X001",
        "streak": "White",
        "cleavage": "one good",
        "mineral-ref": "Gypsum"
      }
    ]
  }
}
> quit
scott@Cosima:~/rwsl/validate/small$
```

```
<!--A simple example in Yanglint -->
<mineral-db
 xmlns="urn:test:minerals">
  <mineral>
    <mineral-name>Talc</mineral-name>
    <physical-properties>White, Green, Gray, or Black. Greasy</physical-properties>
    <hardness>1.0</hardness>
  </mineral>
  <mineral>
    <mineral-name>Gypsum</mineral-name>
    <physical-properties>Colorless or White. Glassy</physical-properties>
    <hardness>2.0</hardness>
  </mineral>
</mineral-db>
<samples
 xmlns="urn:test:minerals">
  <sample>
    <sample-id>X001</sample-id>
    <streak>White</streak>
    <cleavage>one good</cleavage>
    <mineral-ref>Gypsum</mineral-ref>
  </sample>
</samples>
```

Unsuccessful Run

```
scott@Cosima:~/rwsl/validate/small$ expect mineral-error.in
spawn /usr/local/bin/yanglint
> load minerals
> data -t config -f json mineral-error.xml
libyang[0]: Invalid leafref value "Gypsumm" - no target instance "/mineral-db/mi
neral/mineral-name" with the same value. (path: Schema location /minerals:sample
s/sample/mineral-ref, data location /minerals:samples/sample[sample-id='X001']/m
ineral-ref.)
YANGLINT[E]: Failed to parse input data file "mineral-error.xml".
> quit
scott@Cosima:~/rwsl/validate/small$
```

```
<!--A simple example in Yanglint -->
<mineral-db
 xmlns="urn:test:minerals">
  <mineral>
  <sample>
    <sample-id>X001</sample-id>
    <streak>White</streak>
    <cleavage>one good</cleavage>
    <mineral-ref>Gypsumm</mineral-ref>
  </sample>
</samples>
```

Output

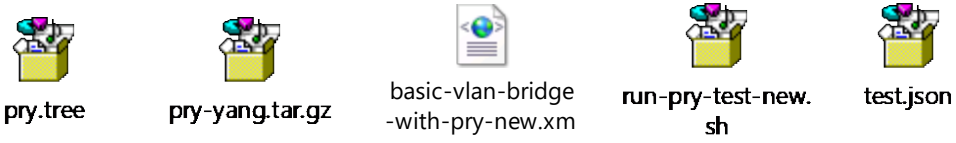
- When the validation succeeds, output is created based on the format chosen in the yanglint “data” command.
- The example command for the simple example is:
  - “data –t config –f json minerals.xml\r”

```
yanglint [Options] [-f { xml | json }] <schema>... <file> ...
    Validates the YANG modeled data in <file> according to the <schema>.
```

- The TYPE used in my validation are either “data” which indicates a datastore along with status data or “config” which indicates data used for a config (no status data)
- The FORMAT tells yanglint to convert the input data into another format. For data, xml and json are available.

Practical Example

- There must be 50 ways to count your Privacy frames!
- Attached is a
  - Tar file with all needed yang
  - Tree file
  - XML file with instance data for input
  - Expect script
  - JSON file that was created upon successful run of yanglint



Other Examples

- https://1.ieee802.org/yangsters/yang-instance-examples/

- Work in progress on creating regression tests for the Features found in Qrev and other subsequent amendments.