pyang Tutorial

Ladislav Lhotka
{lhotka@nic.cz}

1 November 2015
Agenda

- Required software,
- Editing YANG modules,
- pyang plugins,
- Preparing a sample instance document,
- DSDL-based validation of instance documents,
- Converting XML instance documents to JSON.

An extended version of this tutorial is available at https://github.com/mbj4668/pyang/wiki/Tutorial
Required Software

- **pyang**
  
  [Link](https://github.com/mbj4668/pyang)

- Libxml2 tools (**xmlint**, **xsltproc**). Packages available for most operating systems and distributions.
  
  [Link](http://www.xmlsoft.org/)

Optional:

- **Jing** and **Trang**
  
  [Link](https://code.google.com/p/jing-trang/)

- **GNU Emacs (Aquamacs on OS X)**
About `pyang`

Command-line tool written in Python, XSLT and sh/bash.
Extensible via plugins.
Under active development, some plugins and bugfixes only available on GitHub.
Last stable version: 1.6 (2015-10-06), installable via `pip`.

**RTFM:** Unix man pages
- `pyang` (1)
- `yang2dsdl` (1)
Plugins

Conversions to various formats, activated with -f.
Most plugins have specific command-line switches and arguments.

- `yin`, `yang` – YIN and YANG syntax,
- `dsdl` – DSDL hybrid schema (RFC 6110),
- `xsd` – W3C XML Schema (incomplete, deprecated),
- `tree` – schema tree (ASCII art),
- `xmi`, `uml` – UML diagrams,
- `omni` – input to OmniGraffle (OS X and iOS only),
- `jstree` – HTML/JavaScript YANG browser,
- `hypertree` – Hyperbolic YANG browser, to be used with Trebolic,
- `jsonxsl`, `jtox` – XML↔JSON instance document conversion,
- `sample-xml-skeleton` – skeleton of a sample instance document.
Editing YANG Modules

Commercial editors and development environments exist but standard editors mostly suffice.

Special support for popular editors:

http://www.yang-central.org/twiki/bin/view/Main/YangTools

- **Emacs** – yang-mode
- **Vim** syntax file

With Emacs and nXML mode, it is also quite effective to use YIN syntax as the source format, see

https://gitlab.labs.nic.cz/labs/yang-tools/wikis/editing_yang
Example Module: Turing Machine

Available in pyang distribution: doc/tutorial/examples

Radek Krejčí wrote a Turing Machine simulator that can be managed via NETCONF:

https://github.com/CESNET/netopeer/tree/master/transAPI/turing
Essential Steps

- check module correctness
  
  $ pyang turing-machine.yang

- stricter check, mandatory for all IETF modules (RFC 6087 rules)
  
  $ pyang --ietf turing-machine.yang

- generate tree diagram
  
  $ pyang -f tree turing-machine.yang

Help on tree symbols:

$ pyang --tree-help
$ pyang -f tree turing-machine.yang
module: turing-machine
  +--rw turing-machine
    +--ro state state-index
    +--ro head-position cell-index
    +--ro tape
      |  +--ro cell* [coord]
      |     +--ro coord cell-index
      |     +--ro symbol? tape-symbol
    +--rw transition-function
      +--rw delta* [label]
      |  +--rw label string
      +--rw input
        |  +--rw state state-index
        |  +--rw symbol tape-symbol
      +--rw output
        +--rw state? state-index
        +--rw symbol? tape-symbol
        +--rw head-move? head-dir
rpcs:
  +--x initialize
    |  +--w input
    |     +--w tape-content? string
  +--x run
notifications:
  +--n halted
    +--ro state state-index
DSDL Schemas

DSDL = Document Schema Definition Languages


RFC 6110 defines the mapping of YANG data models to three schemas of the DSDL family:

- RELAX NG – schema (grammar) and types
- Schematron – semantic constraints
- DSRL (Document Schema Renaming Language) – defaults

```
$ yang2dsdl -t config turing-machine.yang
== Generating RELAX NG schema './turing-machine-config.rng'
Done.
== Generating Schematron schema './turing-machine-config.sch'
Done.
== Generating DSRL schema './turing-machine-config.dsrl'
Done.
```
Preparing Sample XML Configuration

In an I-D describing a data model, it is often useful to include a sample document showing instance data such as the contents of a configuration datastore.

1. Generate a skeleton document:

   $ pyang -f sample-xml-skeleton turing-machine.yang \ 
   > --sample-skeleton-annotations --sample-skeletondoctype=config | \ 
   > xmlllint -o turing-machine-config.xml --format -

   The skeleton document has to be edited!

2. Convert the RELAX NG schema to the compact syntax:

   $ trang -I rng -O rnc turing-machine-config.rng turing-machine-config.rnc

3. Load turing-machine-config.xml into Emacs.
Schema-based Validation

use pre-generated schemas  use jing  schema name base  XML instance to validate

$ yang2dsdl -s -j -t config -b turing-machine -v turing-machine-config.xml
== Using pre-generated schemas
== Validating grammar and datatypes ...
turing-machine-config.xml validates.
== Adding default values... done.
== Validating semantic constraints ...
No errors found.

Without -j, xmllint is used by default for RELAX NG validation – it works, too, but often gives inferior/wrong error messages.
DSDL Validation Procedure

XML document

RELAX NG schema

DSRL schema

Schematron schema

XML document with defaults

grammar, data types

filling in defaults

semantic constraints
Other Targets (Document Types)

```bash
$ cat turing-machine-notification.xml
<?xml version="1.0" encoding="utf-8"?>
<notification
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2015-07-08T00:01:00Z</eventTime>
    <halted xmlns="http://example.net/turing-machine">
        <state>4</state>
    </halted>
</notification>
```

```bash
$ yang2dsdl -j -t notification -v turing-machine-notification.xml > turing-machine.yang
== Generating RELAX NG schema './turing-machine-notification.rng'
Done.
== Generating Schematron schema './turing-machine-notification.sch'
Done.
== Generating DSRL schema './turing-machine-notification.dsrl'
Done.
== Validating grammar and datatypes ...
turing-machine-notification.xml validates.
== Adding default values... done.
== Validating semantic constraints ...
No errors found.
```
Converting XML Instances to JSON

XML↔JSON mapping is defined in draft-ietf-netmod-yang-json.

JSON is an optional media type in RESTCONF: http://tools.ietf.org/html/draft-ietf-netconf-restconf

1. Generate XSLT 1.0 stylesheet with jsonxsl plugin:

   $ pyang -f jsonxsl -o tmjson.xsl turing-machine.yang

2. Apply the stylesheet to a valid XML instance document:

   $ xsltproc tmjson.xsl turing-machine-config.xml

   The same stylesheets works for all document types.

The jtox plugin performs the opposite conversion.
Further Information

1. NETMOD WG:
   http://datatracker.ietf.org/wg/netmod/documents/

2. NETCONF Central
   http://www.netconfcentral.org/

3. pyang wiki
   https://github.com/mbj4668/pyang/wiki

4. YANG Central
   http://www.yang-central.org/twiki/bin/view/Main/WebHome