YANG Pretty Printer

Introduction

Johannes Specht
Overview

What it is
• Command Line Tool to Format YANG files
• Written in JAVA

Requirements
• JAVA Capable OS (e.g., Windows, Linux)
• Up-to-date Oracle JRE
• May work with other JDKs/JREs

Library Dependencies (JAR-in-JAR)
• AntLR, StringTemplate, Args4j, Apache Commons, diff-match-patch

Why?
• Tedious Tasks: Indentations, Line-wrapping (e.g. YANG description strings), etc.
• Syntax Highlighting in IEEE 802 Stds drafts for easier review
• Idea: Give IEEE 802 Stds YANG files a common design/”Look & Feel”
Basic Functions

Tab replacement

Indentation fixing

“Light” Syntax Highlighting (FrameMaker)
Descriptions

"An IPV can be either of the following:

1) The null value. For a frame that passes through the gate, the priority value associated with the frame is used to determine the frame's traffic class, using the Traffic Class Table as specified in 8.6.6.

2) An internal priority value. For a frame that passes through the gate, the IPV is used, in place of the priority value associated with the frame, to determine the frame's traffic class, using the Traffic Class Table as specified in 8.6.6."

reference
"IEEE Std 802.1QQR, 8.6.5.2";
Break long paths at “reasonable” positions (readability)

Known Machine Grammars:
- YANG
  (IETF "RFC7950")
- Xpath
  (partial)
- Regular Expressions
How to Use It/How I Use It

1. Execute

   ```
   java -jar [pretty printer jar path/]yang802tool.jar
           "<input file path>"
           -W 76 -m -D
           -o "<output directory path>"
   ```

2. Copy&Paste generated Framemaker output file contents into Stds Draft

3. Generate Stds Draft PDF

4. Attach generated .YANG file to Stds Draft PDF
“diff” (new!): Overview

Character Accuracy

Googles/Neil Fraser’s Algorithm, a.k.a. “diff-match-patch”¹

One- or Two-Level Highlighting

- New project’s YANG code vs. old published YANG code (revision) – Amendments and corrigenda (cmp. “2014 IEEE-SA Standards Style Manual”, §18.2)
- Last draft vs. new draft
- Combined (next slide)

How?

- **Underscore, Strikethrough**
- Changebars, conservatively placed (even font changes, e.g. regular normal to Strikethrough)
- FrameMaker’s tracked text edits
  - Previews like “Final” and “Old” previews possible
  - System username – should I implement an override option?
- Configurable colors, style (tracked text edits, Changebars, both)

¹: Eugene Myer’s Algorithm (1986) plus post-processing for human readability – produced the best results in my experiments
“diff” (new!): Simplified Flow

- `<new-draft>.yang`
  - Normalize
  - Format

- `<last-rev>.yang`
  - Normalize

- `<old-draft>.yang`
  - Normalize

1. `new draft vs. last rev. underscore and strikethrough`
2. `old draft vs. last rev. underscore and strikethrough`

3. (new draft vs. last rev.) vs. (old draft vs. last rev.)
   - Tracked text edits, |Changebars, or both
"diff" (new!): Two-Level Change Bars and Tracked Text Edits

```
revision 1980-01-01 {
    description  "Updates from an anonymous project.";
    reference    "N/A";
}
revision 1970-01-01 {
    description  "Initial revision.";
    reference    "N/A";
}
```

```
typedef bridge-ref-type {
    type leafref {
        path
        '/dot1q:bridges'+
        '/dot1q:bridge'+
    }
    /dot1q:component'+
    }
    description  "A broken bridge ref description"
    }
    description  "A working bridge reference with an exhaustive explanation to the previous draft. The reason for this explanation is"
```
Caveats

Outdated IEEE Stds Reference Parser
- Understands “old” Format in YANG files, ignores “new” Format
- Worked well in earlier stages of .1Qcr, .1Qcp, etc.
- Can be bypassed (omit command line switch “-R”)

“Tolerant” Maximum Line Lengths (command line switch “-W”)
- Expect a few characters more/less for string boundary, concatenation, and termination characters (”,+;)

Long Words in Descriptions
- Formatting “description” strings does not break words. Why? Think backwards – implementing breaking is easy, implementing re-merging on text modification is tough/impossible (too much semantic information needed, including [but not limited to] human languages).

“Dual-Diff-Font-Glitch”
- FrameMaker format limitation: No dual layer font overrides. In other words, switching to “Preview Old” may show text with fonts from “Final”. Consider this a temporary glitch, usually not visible in drafts (always “Final”)

Bugs
- Limited testing samples so far → there will be bugs. Please help finding them!
Proposed Next Steps

1. Provide executable JAR to IEEE 802.1 YANG editors for testing
   • Github: [https://github.com/JohannesSpecht/yang802tool-pre](https://github.com/JohannesSpecht/yang802tool-pre)
   • Private Repository - If you are an editor and want to test it, please contact me!

2. Fix issues
   • Based on Editors feedback

3. Collect feature and change requests for enhancements
   • Feasibility
   • Relevance
   • ...

4. Publish
   1. Github?
   2. IEEE Server private/public?