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 (included)
- AntLR, Args4j, Apache Commons

Why?
- Tedious Tasks: Indentations, Line-wrapping (e.g. YANG description strings)
- Syntax Highlighting in IEEE 802Stdbs drafts for easier review
- Idea: Give IEEE 802 Stdbs YANG files a common design
Basic Functions

```.yang
feature closed-gate-state {
  description
  "The bridge component supports gate state closed.";
  reference
  "IEEE 802.1Qcr";
}

/* Types and groupings */
typedef priority-spec-type {
  enum zero { value 0; description "Priority 0"; }
  enum one { value 1; description "Priority 1"; }
  enum two { value 2; description "Priority 2"; }
  enum three {
    value 3;
    description
    "Priority 3";
  }
  /* Types and groupings */
typedef priority-spec-type {
  type enumeration {
    enum zero {
      value 0;
      description
      "Priority 0";
    }
    enum one {
      value 1;
      description
      "Priority 1";
    }
    enum two {
      value 2;
      description
      "Priority 2";
    }
    enum three {
      value 3;
      description
      "Priority 3";
    }
  }
```
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.
Machine Readable Strings

typedef stream-gate-ref {
  type leafref {
    path
  }
}

typedef stream-gate-ref {
  type leafref {
    path
     '/dot1q:bridges/*
     '/dot1q:bridge/*
     '/dot1q:component/*
     '/sfg:stream-gates/*
     '/sfg:stream-gate-instance-table/*
     '/sfg:stream-gate-instance-id';
  }
  description
   "This type is used to refer to a stream gate instance.";
}

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

1. Execute

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

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

3. Generate Stds Draft PDF

4. Attach generated .YANG file to Stds Draft PDF
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 (“,’,+,;)

Bugs

• Limited testing samples so far \( \rightarrow \) there will be bugs. Please help finding them!
Current Development: Diff Indications

```
  feature closed-gate-state {
    description
      "The bridge component supports gate state closed."
    reference
      "IEEE Std 802.1Qaz";

  }

*/ Types and groupings */

typedef priority-spec-type {
  type enumeration {
    enum zero {
      value 0;
      description
        "Priority 0";
```
Proposed Next Steps

1. Provide executable JAR to IEEE 802.1 YANG editors for testing
   - Github?
   - IEEE Server private?
   - E-Mail?

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?