IEEE git-pyang issue

YANGsters

Overview

- Problem Statement
- Git structure in YANG GitHub Repository for IEEE
- The check.sh process
- Compiling Error
- Options
- Discussion

Problem Statement

- There is a problem with the automatic checking scripts when creating revisions of common files
- Specific issue is:
  - There are common files that are being modified by multiple draft projects at the same time
  - When a project ends and the common file is moved to published folder, the other draft projects that are modifying that common file use the common file with the newest revision date instead of the local version of the common file in the draft project directory
- There are several options to resolve this issue

Git structure used by IEEE 802.1

- https://github.com/YangModels/yang/tree/master
  - /experimental/ieee
    - no mandated sub-directories
  - /standard/ieee
    - /draft
      - /802
      - /802.1
      - /802.3
      - /802.11
      - /1588
      - /...
    - /published
      - /802
      - /802.1
      - /802.3
      - /802.11
      - /1588
      - /...

- Expanding on /standard/ieee/draft/802.1
  - There is a sub-directory for each project
- /802.1
  - ABcu
  - AEdk
  - CBcv
  - CBdb
  - Qcr
  - Qcw
  - Qcx
  - Qcz
  - x

- Expanding further for example:
- /802.1/Qcw
  - check_pyang_extra_flags
  - ieee802-dot1q-bridge.yang
  - ieee802-dot1q-preemption.yang
  - ieee802-dot1q-psfp.yang
  - ieee802-dot1q-sched.yang
  - ieee802-dot1q-types.yang
  - ieee802-types.yang

- Common files that are being modified by Qcw

Example

- Qcx is finishing
  - Qcx updates ieee802-types.yang and ieee802-dot1q-types.yang
  - So the version of ieee802-types.yang and iee802-dot1q-types.yang are given a new revision number and placed in the 802/published folder
- When the pull request is made, a check script runs to make sure everything is fine
- Qcw is a project still in drafting, and has its own version of ieee802-types.yang and ieee802-dot1q-types.yang
  - The pyang search path, searches the local Qcw directory first and the 802/published directory second, but pyang selects, imports, and uses the file with the latest revision, so pyang picks up the ieee802-types and ieee802-dot1q-types files from 802/published
- The following slide shows the current check.sh script and the pyang command used
- Following that are several options for discussion

pyang example

- pyang --verbose -p . -p /home/scott/yang/standard/ieee/../ietf/RFC/ -p /home/scott/yang/standard/ieee/draft/802 -p /home/scott/yang/standard/ieee/published/802 -p /home/scott/yang/standard/ieee/published/802.1 -p ../Qcr ieee802-dot1q-psfp.yang

- So the search path for the Qcw file ieee802-dot1q-psfp.yang looks in...
  - Local directory (ieee/draft/802.1/Qcw)
  - ietf/RFC
  - ieee/draft/802
  - ieee/published/802
  - ieee/published/802.1
  - ieee/draft/802.1/Qcr

Options

- Option A
  - Before a common file is moved to published, all draft projects that have their own version of the common file need to set their revision date to a date later than the date the published common file will have
    - Or, make it the job of a YANGsters contact to review and validate the draft common files (setting their revision dates appropriately) before the approved files are moved to published
    - check.sh will need to change so that any pyang command on files in the published directory do not include the path to any draft files
- Option B
  - Set the revision date of all common files being worked on to 2525-12-25 (or some date sufficiently far in the future that it will always be later than anything published
    - this could be a guideline and done once in bulk now to fix the issue
    - check.sh will need to change so that any pyang command on files in the published directory do not include the path to any draft files

Options (continued)

- Option C
  - Modify pyang to provide an option to select the first included file in the path search path instead of the file with the latest revision-date
- Option D
  - Create a check.sh script per project directory
    - check.sh (the main one in the IEEE directory) will have to change to call all the check.sh files in each directory
    - Note: Not sure this helps if a draft project is including files from published and locally, still have the same issue with picking the file with the latest revision date (which may not be the right one)
- Option E
  - Restructure the draft directories
    - Each draft directory would need to include a copy of all the ieee files needed for a clean compile
    - So the check.sh file would change to not include any ieee published path in a compile for drafts

Options (continued)

- Option F
  - Editors of draft projects that are modifying common files, load explicit revisions of the common files
  - When project is approved, the specific revision-date import is removed
- Option G
  - Create a set of file specific sed commands run over a file before the file is compiled
    - A sed-script-file per file could be created
    - The check.sh script would look for the existance of the sed script and then run the sed script on the file before the stream is passed to pyang (example next slide)

Example of sed

Running this…

```
sed 's/prefix ieee;/prefix ieee;\n    revision-date 2019-03-07;/; s/prefix yang;/prefix yang;\n    revision-date 2013-07-15;/' ieee802-dot1q-bridge.yang | pyang
```

- The check.sh script would run sed then pipe to pyang
- Obviously need to fix the pyang options (add paths etc.)

- NOTE: This is an example, the full set of search/replace statements depends on the common files used

```
module ieee802-dot1q-bridge {
  namespace urn:ieee:std:802.1Q:yang:ieee802-dot1q-bridge;
  prefix dot1q;
  import ieee802-types {
    prefix ieee;
  }
  import ietf-yang-types {
    prefix yang;
  } …
```

This

```
module ieee802-dot1q-bridge {
  namespace urn:ieee:std:802.1Q:yang:ieee802-dot1q-bridge;
  prefix dot1q;
  import ieee802-types {
    prefix ieee;
    revision-date 2019-03-07;
  }
  import ietf-yang-types {
    prefix yang;
    revision-date 2013-07-15;
  } …
```

Becomes this

Discussion

- My analysis
  - Option A is cleanest, but more work on editors, requires changes to revision dates on common files
  - Option B is the easiest, seems a bit hacky
  - Option C requires tool changes
  - Option D much work, not sure solves the problem
  - Option E much copying, check.sh scripts per directory
  - Option F requires editing yang files to load explicit revisions, need to remember to fix when publishing
  - Option G script geekiness, isolates the revision info to a local file