YANG Module Revision
Control Examples (v2)

Scott Mansfield
Ericsson
YANGster
Continuation of Discussion

• New material starts at slide 8
YANG Module revisions

• Published modules can be updated by multiple projects at a time.
• We currently do not have a git repository setup for this (A discussion for another time)
• This presentation provides examples on how module revisions are handled until we get an IEEE git system in place

• Reference
Example in Pictures

- C = Commit (accept changes)
- C# = The lower the number the older the commit
Example using files

- C0
  - myproject-types.yang
    revision 2021-09-01 {
      description "published as part of myProject-2021-09";
      reference "2021-09-rollup";
    }

- C1
  - myproject-types.yang
    revision 2021-11-01 {
      description "draft Project 1";
      reference "project1-v1";
    }

- C2
  - myproject-types.yang
    revision 2021-12-01 {
      description "draft Project 2";
      reference "project2-v1";
    }

- C3
  - myproject-types.yang
    revision 2022-01-01 {
      description "published as part of amendment 1 to myProject-2021-09";
      reference "myProject Amd 1";
    }

- C4
  - myproject-types.yang
    revision 2022-02-01 {
      description "published as part of amendment 2 to myProject-2021-09";
      reference "myProject Amd 2";
    }

- C0 has myproject-types.yang
  - revision 2021-09-01

- C1 is based on C0
  - adds types for Project 1

- C2 is based on C0
  - adds types for Project 2

- C3 is based on C1 which is based on C0
  - other modifications needed for Project 1
  - publishes and is now the "latest" version of myproject-types.yang
  - myproject-types.yang
    - revision 2022-01-01

- In order to publish Project 2 a “merge commit” needs to be done
  - So that all the changes from Project 1 and Project 2 are merged and
    conflicts (if any) are resolved.

- C4 is a merge commit
  - myproject-types.yang
    - revision 2022-02-01

- Further amendments or roll-ups will start from “main” at C4.
Example in words

• Published Module
  • myproject-types.yang
    • contains a type called mytype

• Project 1 and Project 2 are running concurrently

• Project 1 adds a type called newtype1

• Project 2 adds a type called newtype2

• Project 1 finishes first so the new version of myproject-types.yang contains mytype and newtype1

• When Project 2 finishes, Project 2 needs to “rebase” from the newest version of myproject-types.yang so that the “main” version contains mytype, newtype1, and newtype2
Practical Example

- CBcv and CBdb are both amending CB at the same time
- CBcv is producing YANG for CB
  - Introduces ieee802-dot1cb-stream-identification-types.yang
- CBdb is adding the mask and match feature to CB
  - Needs to add types for mask and match to ieee802-dot1cb-stream-identification-types.yang
- CBcv will publish a ieee802-dot1cb-stream-identification-types.yang file with a revision date of 2021-12-08
- CBdb will publish a ieee802-dot1cb-stream-identification-types.yang file with a revision date of 2021-12-09
  - based on CBcv 2021-12-08 with the mask/match types added
- In the IEEE802 YANG repository the CBdb version will be the latest version

- If we used git with a branch per project per draft, the merging and conflict resolution would be less error prone, but would require the editors to use a common set of tooling and repositories.
  - tagging and release management TDB
  - connection to remote repositories also TDB
Where the code live?

- The IETF’s YANG Catalog github repository is on github’s website
- I create a “fork” of that github in my personal github on github’s website
- I then create a clone of my fork on my local Ubuntu VM running on my Windows 10 PC.
Git terminology

• From the perspective of my clone.
  • “origin” is the name of my remote repository
    • Points to my samans/yang repository
  • “upstream” is the name of the remote repository I forked from
    • Points to the YangModels/yang repository

• branch
  • A collection of files/modifications that are tracked together so they can be committed

• commit
  • stores the current contents

• merge
  • resolve conflicts when committing

• main (or master)
  • refers to the default branch of a repository (main is the current, master is deprecated terminology)
IEEE Perspective

• We have some choices

• We could create an IEEE repository that everyone treats as the main remote repository
  • Everyone could either fork or clone the main branch of the IEEE repository.
  • The only difference is what git commands you need to use to get your changes reviewed and merged into the main repository.

• Benefit is we could have a repository owner and set up our own validation checks and have control over when branches are merged.

• From an IEEE editor’s perspective, this would also alleviate the complexity of dealing with the IETF’s repository, the IEEE YANG repository owner could deal with that.
Options available

• If the IEEE YANG editors want to have their own environment
  • and
• There are people willing to serve as repository managers
  • We could setup a yang repository on https://opensource.ieee.org/users/sign_in
    • We would need to have a way to refer to YANG that is not stored in the IEEE repository
    • The repository manager would need to monitor and service
      • Merge requests
      • Determine when files get moved to the IETF Repository
      • Support updating IEEE yang file repository on dot1 website

• Otherwise, we could create a github repository and setup a “sub-module” like the BBF has done.
  • IEEE YANG repository manager would maintain the sub-module
  • Support when to update the IETF github repository
  • Support updating IEEE yang file repository on dot1 website