Living Standards
IAB/IESG Retreat 2021

Cullen Jennings
Rob Wilton

Other Info

Warren Kumari has a bunch of work on this

“Checkpoint drafts”

https://slides.com/wkumari/evolvingdocuments/fullscreen

https://docs.google.com/document/d/1QA2Hym85OPgT_QGhnWw9Wd7_hKzV025djiKKscFWcEw/edit?usp=sharing

Tiny Changes

- Update metadata. For example the “replaced by”, “errata exist”
- Update XML non rendered data that does not significantly change the rendered output such as new defaults for old XML tags
- Provide live pointers to approved eratta

Small Changes

- Apply approved errata to the document
- Allow errata that change the normative way the document operates

Medium Changes

- Apply new backwards compatible extensions that are optional to implement

Big Changes

- Non backwards compatible changes
- New mandatory to implement functionality
- Changes to references to documents that require code to change. (For example deprecating a previously allowed cipher suite)

Reference Issues

- Standards are agreement's on what different systems will do. They have to be able to refer to an absolute version. Vendor state supported to customers. Equipment is tested to work against given versions.
- For living documents, this can be solved with dated version (or otherwise non mutable version).
- The approval status dated version tends to get very confusing.
- References inside a spec have to refer to dated or specific versions of the documents they are referencing. To say it another way, normative references have to be stable. (As a side note this is one of the many reasons the IETF errata process is currently broken)

What problem are we solving?

- The number of positive integers is too small. (Seriously some people don't want to issue a new RFC number for each dated version of a spec)
- The review process and effort to publish an updated specification for a small change is too large
- The IETF Last Call approval process is not wanted
- The IESG approval process is not wanted
- The RFC Editor process is not wanted
- Standards writers want developers to always support the latest version of the specification

What problem are we solving?

YANG + other code assets

- YANG modules are Code/APIs, not documentation, but put into RFCs + Github
- Code/APIs needs bug fixes, incremental improvements, changes (ex. Opus codec)
- Operators want consistent “sets of YANG modules@version” aka YANG packages
- YANG package revisions will change whenever any included modules change, need a lightweight process
- But still want IETF consensus
- And don’t want RFCs pointing to stale versions.

Discussion (not worried about solutions)

- Numbers ...Imagine living docs were RFC number, decimal point, then draft version number. Not arguing this is a good solution, just strawman that shows this problem seems solvable.
- Could we review only the changes since the proviso approved version, and not the whole document?
- Would we even want that? Imagine the following case: we have a BCP that says MD5 is not allowed for certain uses, and we have a protocol that uses it. We have an update to that protocol that has nothing to do with crypto algorithms. Do we want to insist the protocol is upgraded to be compliant with the current BCPs before adding any other extensions or improvements?
- To quote Mr. Peterson, “More review finds more problems” and perhaps the whole idea of cross area review by the IESG is totally useless. What level has the IESG found problems that the WG missed? (Note my position as an ex-AD is that the WG regularly send stuff to the IESG that clearly no one has read)

Anti-competitive Issues

- Hard part of living documents is "consensus" on what goes into the the specification.
- Often end up with situation where some people can add changes without consensus then remove if they there are problems, and others have to get full consensus before the changes gets added.
- Many existing living documents at W3C, WhatWG, and IETF end up with a single dominant implementations that if it is not willing to accept the changes, the change does not make it into the required to implement part of the living standards.
- In practice this ends up highly anti-competitive where one vendor or cartel of vendors has absolute veto over proposed changes.
- WhatWG tries to deal with this by having voting members be browsers of certain "stature" but in practice large parts of all qualifying browser are funded by google.

Version Fragmentation

Imagine that instead of TLS 1.1, 1.2, 1.3 we have 500 smaller versions representing these and all the intermediate steps in between them. A wide variety of clients and servers would support various versions of the 500 version. Would that be better or worse?

- It would mean things would be easier to incrementally deploy
- It would make it close to impossible to test the deployed permutations

Probably works well where the only things that matters if every product can interoperate with the a single dominant product that sets the defacto standard.

Extension Linearization

- SIP has tons of extensions
- There is no common set of ones everyone implements
- If all of the extension were part of document, it might change what vendors implement
- Or not, if all the extensions were optional