CoAP Problem Details

draft-fossati-core-coap-problem-02

Summary

- Standardise an error reporting format for CoAP APIs – RFC 7807-like
- -00 published Nov 2019; 2 iterations since then
- Got airtime in Singapore @ CoRE APPs side-meeting
- Got some quality (on- and off-line) discussion
- Time seems ripe to discuss next steps with the wider working group

Quick recap

- Structure of Problem
  - Global block
    - Error identification: `ns` and `type`
    - Common fields: `title`, `details`, CoAP `response_code`, `instance` URI
  - Local block
    - Per namespace extensions: API developers can define their (ANY DEFINED BY `ns`) stuff
    - The keys defined here (TODO, in a separate map) have scoped meaning
- Name-spacing
  - `ns` codepoints can be private (<0) or public (>0)
  - When / if API goes public, renumbering happens by grabbing a public `ns`, the rest (types and per-`ns` extensions) stays the same

Issue #19 - Localisation

- Is there anything we can do to help here?
- Should we recommend a default language?
- Should we add language tags a la CoRAL?

Issue #14: “X dash”

- Context: RFC 6648, in particular the analysis in Appendix B
- The problem is if the producer never updates to the public format, consumers – not just CoAP clients but the whole logging pipeline – need to cope for an indefinite amount of time
- Unfortunately, consumers don’t seem to have much leverage
- We define a private-to-public migration plan from the onset
  - To what extent is that effective in preventing the problem?
  - Provide discussion on strategies for minimising the risk of “eternal pollution” (e.g., use an automated software update mechanism)

Open questions

- Jim suggests subsuming the diagnostic payload under the problem structure:
  - Add another optional `diagnostic` key in the “Global” map
  - Christian: *“APIs that need something similar could add their own extension”*
- I think the question is: is this going to be common enough that is worth factoring it out proactively?
- Is there an appetite for that?

CoRALization?

- PRO: technically superior:
  - absorbs encoding, compression, transport variability
- CON: depends on the CoRAL machinery
  - Q: how strong is the dependency? Can it exist with a minimalist implementation that has comparable complexity with the current spec?
  - How long will it take to get it out?
    - Which is really a question about CoRAL stability – when can we expect CoRAL’s moving parts (at least those that would have an impact here) to become fully stable?

Discussion Points

- Is standardization needed here?
- Is this ready for adoption?