Structured Representations in IETF Documents: Notes from IETF 103

November 7, 2019

These notes are from informal discussions and surveys carried out at IETF 103 in Bangkok, Thailand, in November 2018.

1 Adoption of current representation formats

Survey participants were given a list of the structured representation formats commonly found in IETF documents[1]. While the sample size for the survey was too small to draw conclusions about broad trends, none of the participants indicated that they had seen CBOR in IETF documents. All of the other formats had been seen by at least one participant, with the all participants having seen C code, YANG models, and ASCII packet header diagrams.

When asked if these structured representations improved the utility of the documents they were included in, the broad consensus was that they did – but not significantly. The representations were good to have, but they do not replace the body of the document (i.e., the English prose). Further, there was concern that structured representations – especially code and pseudocode – were actually detrimental to readers understanding of the document. There were two reasons given for this: (i) the (pseudo)code can be overly complicated, and more difficult to understand than the English prose version; and (ii) because one is not typically generated from the other, differences can arise between the pseudo(code) and the English prose. On the positive side, participants noted that some representations – particularly diagrams, such as packet header and sequence diagrams – were useful for clearly defining the syntax of the protocol, and giving structure and context to the English prose that accompanied them.

Participants were split on whether or not the current level of adoption of structured representations ws sufficient. The learning curve involved in understanding these languages – or at least having sufficient knowledge to make good use of them in documents – limits their adoption. In addition, adding a structured representation requires repeating yourself: if some protocol feature has already been described in English, you don't want to encode the same in code.

Other miscellaneous feedback:

- A lack of tooling makes adopting structured representations more challenging in a lot of respects: correctness, repetition, lack of grammar for ad-hoc pseudocode languages
- Text-based protocols typically use ABNF, there isn't a similarly widely adopted format for binary protocols
- Would be good to write one (in English or a structured language), and then generate the other. This would avoid duplication and inconsistency.

2 Generating parsers from standards documents

The next set of questions focussed on the generation of parser code from a structured representation of the protocol's syntax, contained in the standards document that specifies the protocol.

First, participants were asked for their opinions on the existing approach. Currently, protocol syntax is specified using an ASCII packet header diagram, followed by a list of descriptions (in English), detailing the syntax and semantics of each field. When asked about the benefits of this approach, participants said that flexibility in how these diagrams and their accompanying descriptions are written/generated is useful: people have different workflows. The visual representation is also useful, especially for bit alignment, and writing implementations: knowing bit offsets is very useful. In terms of the limitations, participants highlighted issues that arise from the use of ASCII packet header diagrams. These diagrams lack expressivity, ultimately pushing the description of much of the complexity of the protocol into the English descriptions that follow the diagram. This is much more difficult to read, and is particularly problematic for those whose

[1] ABNF, XML, ASN.1, C code, JSON, CBOR, TLS presentation language, YANG, MIB, state transition diagrams, ASCII packet header diagrams, protocol sequence (ladder) diagrams

first language is not English. The current ASCII packet header diagram format makes it difficult to express certain protocol features: optional and variable length fields were highlighted as being difficult to represent.

Finally, it was explained to participants that generating parser code from standards documents would require the use of a structured representation when describing protocol data units. Participants were asked how adoption of such an approach should be encouraged. Broadly, participants suggested two approaches: (i) essentially require that a standard description format is used, or (ii) make the benefit of a standard format clear, such that people will voluntarily adopt it. It was clear that the second of these was much more suitable. It was suggested that the upcoming change to the RFC format would be a good opportunity to promote new tooling, given the need to generate SVG, for example.

Other miscellaneous feedback:

- Other languages (CDDL[2] and Google's protobufs[3]) were highlighted. These appear to have similar motivations and syntax/semantics to our intermediate representation, and should be further researched.
- Again, participants indicated that they want to specify syntax once, and use tooling to generate other formats. Specifying the same thing in multiple formats duplicates effort and allows for inconsistencies to arise.
- Being more structured is good for human readers too.
- The use of structured formats is also good where there are multiple authors: more structure makes managing differences in approach easier. For example, the required "Security Considerations" section enforces structure, and makes authors think about the content of the section.
- Whatever tooling is developed should be co-ordinated with the RFC Editor. It is already difficult to get started when writing a draft. The tooling should work with the existing workflow, and not introduce another layer.

3 Other application domains

Participants were asked about other application domains for structured representations, beyond the generation of parser code.

A number of suggestions were made:

- Validation of other serialisation formats, such as JSON, XML (against a DTD).
- Checking that paths in sequence diagrams are reachable.
- Introduction of UML, with a format that logic and analysis can be performed on.
- YANG models are increasingly common in documents. Often inconsistent in formatting. Tooling to validate/format consistently would be helpful.

[2] https://tools.ietf.org/html/draft-ietf-cbor-cddl
[3] https://developers.google.com/protocol-buffers/