RFC 8949-to-be (7049bis)

AUTH48 notes

Carsten Bormann, CBOR @ IETF 109, 2020-11-19

24 questions from RFC editor

(Mostly really good; incredibly detailed)

- 1: C++20 reference, now without paywall
- 2: superscript formatting for exponentiation (problematic TXT fallback)
- 4: use more italics, 7: use more monospace (but problematic TXT fallbacks)
- 5: differentiate <artwork> and <sourcecode>, proper `type=` attribute values
- 15: can’t use »guillemets«, so stuck with >compares< for quoting
- Oopsie: s/positive bignums/unsigned bignums/g
- Please look at https://github.com/cbor-wg/CBORbis/tree/rfced-questions

Timing?

- Expect us to be finished with answering questions in a few days
- Expect us to do another full re-read over roast turkeys
  - You are invited to help
- Should be able to finish AUTH48 ~ by end of month

- We are currently impressed by RFC editor velocity — hope that stays so

CBOR Tags for OIDs

draft-ietf-cbor-tags-oid-03

Carsten Bormann, CBOR @ IETF 109, 2020-11-19

–03 changes

- Add tag TBD112 (“p”), structurally like TBD110, but understood to be relative to "1.3.6.1.4.1" (IANA Private Enterprise Number OID).
- Add suggested CDDL typenames (`oid`, `roid`, `pen`).
- Missing: CDDL control operator for stripping a prefix from an oid, turning it into a roid — inverse to `.cat`. (#3, https://github.com/cbor-wg/cbor-oid/issues/3.)

```
root-sha2 = bytes .sdnvseq [ 60 840 1 101 3 4 2 ]
root-sha256 = bytes .sdnvseq [ 60 840 1 101 3 4 2 1 ]
rel-sha256 = root-sha256 .oidrel root-sha2; bytes .sdnvseq [1]
```

- Really, this is prefix removal in byte strings, nothing specific to oids.
- Do this in cddl-control, alongside `.cat`?

Ship it?

Packed CBOR

draft-ietf-cbor-packed-00

Carsten Bormann, CBOR @ IETF 109, 2020-11-19

JSON, CBOR: Coding efficiency

- CBOR can be more efficient than JSON, in particular if the data model is specifically designed for CBOR (e.g., integer labels in maps)
- Simply encoding JSON data in CBOR reaps less gain
- Significant redundancy often remains
  - Can be removed by, e.g. DEFLATE (RFC 1951)
  - Compression requires decompression before use, though
- Alternative: Exploiting structure and prefix sharing by “**Packing**”
  - CBOR data item can be used while remaining packed

2020-10-28 interim

- Clearly separate “table setup” from “table referencing”
- 3 Tables for (whole) Item Sharing, Prefix Sharing, Suffix Sharing
- Structure tables into efficiency buckets (1+0, 1+1, 1+2, 1+4)
- Inputs for table setup: local, surrounding data item (?),
  static dictionaries (set of tables for i/p/s)
- When combining, overflow to end of next less efficient table

–00: efficient Item and Prefix references

- Item references: 16 simple values (1+0),
  one single-byte Tag ➔ 48+512+131072 (1+1, 1+2, 1+4)
- Prefix references: Reuse tag; use more tags (32+4096+268435456)
  Do the same (but not necessarily the same sizes) separately for suffix
- Total reservation: 4/7 simple values, 1 1+0 tag (1/24), 1/8 1+1, 1/16 1+2, …
- Worth it if we think this will be a widely used part of CBOR
- Could be less agressive and less efficient, but why?

How to reference dictionaries (external tables)

- Referencing (and table building!) scheme clearly separated from packing/ referencing scheme
  - Could be (partially) application-specific
  - Could employ additional tags for those application-specific parts
- URIs: Identify + locate
- Hashes: Identify only
- (IANA-)Registered dictionaries: Identify; locate if known

Now build the –01 with all that

- Reserve more tag space for suffix references; reality check
- Nail down the **referencing** part — few further changes to be expected
  - Could ship this separately, but batteries are then not included
- Do one or more **table setup** schemes
  - Could do a simple one first
  - Could attack (external) dictionaries in next step
- Split the draft?