CDDL:
Additional Control Operators
draft-bormann-cbor-cddl-control-01
Additional Control Operators for CDDL

draft-bormann-cbor-cddl-control-01

• .cat .plus
• .abnf .abnfb
• .feature [implemented]

• Register?
• Adopt?
CBORbis issue #204: Diagnostic notation should be surjective (empty strings, NaNs)

https://github.com/cbor-wg/CBORbis/issues/204
CBOR diagnostic Notation

• CBOR extended diagnostic notation (RFC 7049 Section 6 + RFC 8610 Appendix G), EDN, provides a human readable form of a CBOR data item at the data model level

• Almost all CBOR data items can be expressed uniquely in EDN

• Exception 1: (_ ) can be an empty byte string (5FFF) or text string (7FFF): No obvious proposal, (_b ) and (_t ) crutch maybe

• Exception 2: NaN payloads are lost: Probably just provide the whole number in hex, e.g. NaN_1_x7E00?
Next steps?

- Don’t try to shoehorn this into 7049bis
- Maybe new document that
  - Points to RFC 7049 Section 6 (DN) + RFC 8610 Appendix G (EDN)
  - Makes these small additions: ( _ ) and NaN
  - Provides more examples for DN/EDN
  - Maybe provides an ABNF (gasp!)?
CBOR Tags for OID

draft-bormann-cbor-tags-oid-07
CBOR Tags for ASN.1 Object IDs
draft-bormann-cbor-tags-oid-07

- Draft was started in October 2014, with Sean Leonard, 2½ years … -06
  - Was accreting more functionality on the way than we maybe really needed
- Use cases in RATS and related now create some urgency
- -07 reduces content to what is really needed
- Adoption call ended yesterday — chairs’ evaluation?
- Beyond editorial issues, the Tag Factoring functionality is at risk
  - Could solve this while this is a WG document
Editor/Contributor question

- I’m currently unable to reach Sean Leonard.

- (With RFC 8746, we handled a similar issue by moving an author to the contributor list.)

- The chairs can decide this now, or at any time [RFC 2418].
Packed CBOR
draft-bormann-cbor-packed-01
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
Structure Sharing

- Many data items nested in a larger data item repeat
  - E.g., strings used for labels or enums
- Idea: Provide one copy of repeated item and share it
- Item is put into a sharing array, referenced in the places where a copy is needed
Prefix Sharing

• data items often share a **prefix**
  • E.g., initial parts of URIs are often similar

• Idea: Provide one copy of repeated prefix and **share** it

• Common is put into a prefix array, referenced in the places where a copy is needed
Structure of packed CBOR

- Packed data item is an array tagged with tag 6:
  0 1 2 3 4 5 6
  6(                                                             )

  - Rump can reference shared items; shared items can, too (yes, needs loop detection)
  - Items can use a prefix (identified by a tag) plus a supplied suffix
Experiment

wot-thing-description/test-bed/data/plugfest/2017-05-osaka/MyLED_f.jsonld

— JSON file: 3116

— **JSON no whitespace: 1447**
  — deflate: 323, lz4: 415, lz4hc: 411

— **CBOR: 1210**
  — deflate: 325, lz4: 416, lz4hc: 404

— CBOR packed (semantic sharing only): 793

— CBOR packed (prefix compression, too): 564
Conclusion

— Packing (exploiting structural sharing)
— maintains processability
— saves ~ 1/3 (implementation not yet complete)
— Prefix sharing helps with URLs, another 20 %
— but reduces processability
— Could further improve by adding static dictionary
— In the example: 119 bytes of mostly static data:

```
[
  "name",
  "@type",
  "links",
  "application/json",
  "outputData",
  "mediaType",
  "href",
  {
    "valueType": {
      "type": "number"
    }
  },
  {
    "Property": {
      "writable": "valueType",
      "type": "application/json"
    }
  }
]"
Both Item and Prefix references need to be efficient

- Item references: 16 simple values (1+0), one single-byte Tag $\rightarrow$ 48+512+131072 (1+1, 1+2, 1+4)
- Prefix references: Reuse tag; use more tags (32+4096+268435456)
- 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 aggressive and less efficient, but why?
Standard defines unpacking

- As usual for compression — define decompression, enable diversity in compression effort
- Packing can be done at different levels of complexity
  - Just find shared strings
  - More generally, find shared items (and nest)
  - Add common prefix detection
- Algorithm left as an exercise to the reader
  - May need a reference algorithm (TBD)
Can we go ahead with packed CBOR?

- Interesting development in W3C: “CBOR-LD” proposal
  - Does a form of packing specific to JSON-LD
  - Proposes to use external dictionaries for efficiency
- Could add external dictionaries to packed CBOR, too
- Could add “prefixes” for maps (sets of key/value pairs)
- These could be done for a WG document
Notable CBOR Tags

draft-bormann-cbor-notable-tags-02
Notable CBOR Tags

draft-bormann-cbor-notable-tags-02

• Collect definitions of registered tags that are widely dispersed
  • Give the growing field a structure and some additional explanations
  • As far as possible, collect and preserve defining text for tags

• During development of 7049bis, served as repository for some tags we found we needed

• This can live as an individual document for quite a while
  • It would still be useful to have some feedback