2017-01-09: CBOR WG

• Concise Binary Object Representation
  Maintenance and Extensions

  1. Standardize CDDL as a data definition language
     (May 2018 milestone, actual: August 2018)

  2. Formal process: Take RFC 7049 to IETF STD level
     (October 2018 milestone)

  3. (Maybe define a few more CBOR tags, as needed.)
CDDL
CDDL

Draft-ietf-cbor-cddl-08 ➔ RFC 8610

2019-06-12
Nach dem Spiel ist vor dem Spiel
(=After the game is before the game)

Next steps on CDDL
Peeking post-1.0

• SUIT people tell us they’d now really like:

  • Import function (here: for COSE)

  • Namespace control (related to import)

  • At some point, a module registry may make sense

  • (For more ideas, see also IETF102 slides)
draft-bormann-cbor-cddl-freezer

- Collected items that were not done for CDDL 1.0
- Can be thawed now
- What should we pick up?
- Let’s prioritize today
Things that can be done on the side (no new CDDL needed)

- .pcre
- Big-endian .bits
- .bitfield
Alternative Representations (1)

cddl = ['cddl', +rule]
rule = ["=" / "/=" / "//=", namep, type]
namelp = ['name', id] / ['gen', id, +id]

id = text .regexp "^[A-Za-z@_$][\[-\.]\*[A-Za-z0-9@_$]]*"  

op = ".." / "..." / 
    text .regexp "\\[A-Za-z@_$][\[-\.]\*[A-Za-z0-9@_$]]*"

namea = ['name', id] / ['gen', id, +type]
type = value / namea / ['op', op, type, type] / 
    ['unwrap', namea] / ['enum', group / namea] / 
    ['prim', ?(0..7, ?uint)]

group = ['mem', null/type, type] / 
    ['rep', uint, uint/false, group] / 
    ['seq', 2*group] / ['gcho', 2*group]

value = ['number'/'text'/'bytes', text]
Alternative Representations (2)

labeled-values = {
  ? fritz: number,
  * label => value
}
labeled-values

➔
["cddl",
[="",
["name", "labeled-values"],
["map",
["seq",
["rep", 0, 1, ["mem", ["text", "fritz"], ["name", "number"]]],
["rep", 0, false, ["mem", ["name", "label"], ["name", "value"]]]]],
[="", ["name", "label"], ["name", "text"]],
[="", ["name", "value"], ["name", "number"]]]
Field = uint .bitfield Fieldbits
Fieldbits = [
    flag1: [1, bool],
    val: [4, Vals],
    flag2: [1, bool],
]
Vals = &(A: 0, B: 1, C: 2, D: 3)
2. Base Language Features

• 2.1 Cuts (e.g., for whole map members)

• 3.1 computed literals (base = 400    a = base + 4)

• 3.2 tag-oriented literals — dt'2019-07-21T19:53Z'

• 3.3 regular expression literals

• 4 Embedded ABNF
Larger projects (1)

• Co-occurrence constraints

• Predicates

• Pointers/Selectors

    session = { … timeout: uint, … }

    other-session = {

        timeout: uint .lt [somehow refer to session.timeout],

    }
Larger projects (2)

- Module superstructure
- Namespacing
- Import/Export (relating to URIs?)
- Versioning
Larger projects (2a)

- Variants
  - Particularly: CBOR and JSON variants
Larger projects (3)

- Augmentation
- Relationship to semantics, RDF, ...
- Get real default values
- Add units and other metadata
Should there be a CDDL roadmap WG document?

- Could adopt something like -freezer as WG document
- No intent to ever publish as an RFC
- But an “official” document with (at least a snapshot of) directions that are moving towards consensus
- Document priorities
CBOR (RFC 7049) bis
Concise Binary Object Representation
Carsten Bormann, 2019-03-27
TODOs left from IETF104
Levels of Errors #45

• (not) well-formed — CBOR Syntax
  • Error: Not recoverable (outside diagnostic tools)
  • See also Appendix C (pseudocode)
• (not) valid — CBOR Semantics
  • Error: Presentable to the application in principle
• (not) expected — Application Syntax and Semantics
  • This is often expressed in CDDL
To do: strict (from 104)

- A strict decoder only accepts preferred encoding
- Again, this also has an application component
- Similar: deterministic-checking decoder
- Text about security miracles already toned down
Note: strict mode ≠ validity

- Probably need better terminology here.
- Require-deterministic vs. require-valid
  - The latter is hard to do for all tags
- UTF-8 validity is mostly fine
- Map validity can only be enforced at generic decoder precision; needs application help anyway
Tag validity (1)

- At IETF104, we discussed purely structural vs. semantic validity conditions for tags.
- Decided to move some non-essential tags to a separate document to open them up for semantic validity.
- On further reflection, this sends the wrong message.
- Don’t do that, then.
Tag validity (2)

• Stick with structural tag validity

• Mention that validity, as always, is ultimately an application concept (#86)

• Encourage generic decoder implementations to present structurally invalid tags as such to the application

• Application can then always implement semantic validity, if desired
Tag validity: #92

• Some early tags **cannot** generally be processed by the application: Tag 25, Tag 29 need to know the serialization order
  • Some implementations preserve ordering even in maps, so the application can process these tags
  • Many don’t, so the generic decoder would **have** to process these tags during decoding
  • This limits interoperability to a subset of decoders
• Mention that these tags exist, discourage (SHOULD NOT) creating more of these, but don’t outlaw between consenting implementations
• Note that this is different from making applications depend on map ordering, as this **can** be implemented by the decoder
Tag validity: Embedded CBOR (Tag 24)

• Tag 24 (Embedded CBOR) does not require anything from the byte string for tag validity

• Tag 36 (Embedded MIME) does require valid MIME for tag validity

• Suggestion from the interim: make Tag 24 require wellformedness (not validity) for tag validity (#86)

• Maybe give some guidance for tag developers (#86): Don’t overdo validity requirements, but do give generic decoders a chance to do useful work
Other validity Checking

• Make map validity checking mandatory? #63

• This might be the other dimension of “strictness”
Newer Issues
JSON-to-CBOR conversion (1)

- Fish sticks ➔ aquarium

- JSON numbers are not identified as integers or floats separately; they are floats that can be integer (10, 10.0, 1e1)

- CBOR separates the worlds of integers and floats; conversion needs to make a decision

- Floating point range is greater than base CBOR integer range: Not all floats that appear as integer can be converted to 64-bit integer

- But then, in I-JSON, everything above 53 bits is inexact anyway
Recommendation:

Decide between pure JSON and I-JSON

Pure JSON: Anything that is integer in JSON data model is represented as integer in CBOR (mt 0/1, tag 2/3)

I-JSON: Anything that is integer after conversion from decimal to binary64 and is $|x| < 2^{53}$ (allowing exact representation) becomes a CBOR integer (mt 0/1); everything else stays float (mt 7 ai 25/26/27)
Major Editorial Todos

- get rid of "follows" terminology #85
- Add redundant text for:
  - Uneven number of items in a map is not-well-formed #80
- More cleanup security considerations #90
- Data item vs. encoded data item #64
Minor editorial

• #68: advice on small integer Map keys

• #67: describe options in handling unknown extension point values (Tags/Simple values)
Slides from IETF104
Tag validity

- Example: Tag 1 (POSIX time) takes int/float

- Maybe should have taken decimal as well (then we may not have needed Tag 1001)

- Similar: Tag 36 (mime message) only takes UTF-8 Should have taken byte string as well Now have 257 for that.
Reactionary Tag Validity

- Tag is defined with a certain set of substructures (structural compatibility)
- A new substructure can never accede to an existing Tag
- There is little ambiguity about Tag validity
Progressive Tag Validity

• Tag is defined with abstract semantics

• Any substructure that fulfills that abstract semantics will do

• E.g., Tag 1 could take any number in \( \mathbb{R} \)

• E.g., Tag expecting array of numbers could take typed array (Tag 64..87)
Application expectedness of Tags

• CDDL: #6.36(tstr) vs. #6.36(tstr/bstr)

• Note that standard prelude says:
  mime-message = #6.36(tstr)

• But application saying #6.36(tstr/bstr) is unambiguously using the tag
Ways forward

• Clarify the reactionary tag validity approach taken in RFC 7049 (done well by PR #18)
  
  • Much stricter
  
  • Still modulated by application expectedness
  
• Move to progressive tag validity
  
  • Much more flexible
  
• Potential interoperability surprises outside CDDL
How to specify Tag type system

• New tag definition should document
  • expectations from tagged value (e.g., $\epsilon_\mathbb{R}$)
  • Abstract “type” of the result
Other todos

• Check Strict some more

• Clean up preferred encoding; base deterministic encoding on this

• Slightly Update IANA considerations

• (We have another specification required in 1+1)

• One more round of reviews, and then WGLC?
Other CBOR housekeeping
draft-bormann-cbor-sequence

- Patterned after RFC 7464 (JSON Sequences)
- Format definition, Media type, Content-Format, ...
- But quite different:
  - CBOR is easy to concatenate
    (no ASCII RS needed)
  - No attempt at error recovery needed or possible
- People already want to put normative references to this into their documents
CBOR tag definitions

Carsten Bormann, 2018-07-17
Batteries included

- RFC 7049 predefines 18 Tags
- Time, big numbers (bigint, float, decimal), various converter helpers, URI, MIME message
- Easy to register your own CBOR Tags
- > 20 more tags: 6 for COSE; UUIDs, Sets, binary MIME, Perl support, language tagged string, compression
Status of Tags drafts

- **OID**: On charter, kitchen sink, expired. Needs work.

- **Array**: On charter, WGLC completed, waiting for write-up.

- **Time**: Off charter; solved for now by FCFS registration (3-byte tag 1001); move spec to RFC how?

- **Template**: Off charter (will likely be done with SCHC anyway)

- **“Useful tags”**: Maybe document some of the more useful registered tags in an RFC on its own (could include Time)?