2017-01-09: CBOR WG

- Concise Binary Object Representation Maintenance and Extensions

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

2. Standardize CDDL as a data definition language (May 2018 milestone)

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

CDDL

Henk Birkholz, Christoph Vigano, Carsten Bormann

draft-ietf-cbor-cddl

Changes since IETF102

- –03:
- Editorial: clarify group entry definition, clarify barewords, fix “inheritance” example, typos.
- Say that 1 is int (so does not match 1.0) and 1.0 is float (so does not match 1).
- Add security considerations.
- Add straw man for control operator registry (policy to be decided).

Changes since IETF102

- –04-in-the-making:
- Define “byte”.
- Say what target types “.size” is defined for.

Changes we missed

- (Jim's review. Ouch.)
- Need to convert into issues and act upon them.

Open Issues (1)

- For the freezer (CDDL 2.0):
  Co-occurrence constraints (#22).
  (Tool issue #5 waiting for cddl tool v2)

- Editorial: Be a bit more explicit about history and contributers.

Open Issues: Specify group matching #14

- I believe this is an editorial issue.
- Maybe include some of the clarifications and examples from the thread at https://www.ietf.org/mail-archive/web/cbor/current/msg00380.html

Open Issues:
IANA registry for control operators? #17

- Recent discussion on the mailing list
- (1) Not sure we have consensus that there should be a registry at all
- (2) What is the policy? Proposal: Specification Required, plus guidance for Designated Expert to actually look at the specification and apply some quality control (hinted at by RFC 8126 but not consistently practiced)

Open Issues: Which data model do .eq and .ne use? #18

- .lt/.ge, .gt/.le are defined for numeric types
  - Not clear that there is value trying to extend
- .eq/.ne are useful in two variants:
  - (1) Numeric equality (with no intent to solve “epsilon”):
    - `(1..5) .ne 3.0 ≡  1 / 2 / 4 / 5`
    - `number .eq 3  ≡  3 / 3.0`
  - (2) Structural equivalence (which can be used for non-numeric types, too); define semantics close to “matches”/“does not match”
- .default is like .ne, but probably the structural one

.ne/.eq naming?

- `.ne` is in use today (`.eq` is mostly there for symmetry)
- Often useful with non-numeric semantics (for general type difference), and would even be useful with more than a single value:
  - `label .ne “foo”`
    - `label .ne (“foo” / “bar”)`
    - `label .ne keyword`
  - `any .ne bytes`
  - Note that we already have `.and` for type intersection
- (1) break those specs by defining .eq/.ne like the (numeric) inequalities; add a structural difference/subtraction control
- (2) cater to those specs, introduce numeric variants for .ne/.eq (but leave .lt/.ge and .gt/.le alone)

Serialization variants

- Discussion of “representation variants” on mailing list: one data item has multiple representations

- Issue comes up on two levels: information model to data model, data model to serialization; the discussion was really about the latter

- Maybe be more specific about the latter and talk about **serialization variants**

CBOR issue: Serialization variants and Consistent Encoding

- Consistent Encoding (“c14n”) defines a preferred serialization variant for each set of serialization variants
  - Expensive (map sorting)
- Maybe there should be a “preferred encoding” as well (like consistent except where that would be expensive)
  - Define “expensive”, then

Serialization choices

- A format specification employing CBOR may
  - Disallow floating point numbers
  - Disallow 64-bit floating point numbers
    - Disallow 64-bit integers, string lengths, item counts, tags
    - Disallow indefinite lengths
    - Disallow indefinite lengths for byte/text strings
- NOT RECOMMENDED: Disallow a preferred encoding, while selecting a non-preferred encoding

Serialization choices vs. data model level

- Disallowing serialization of floating point numbers makes it useless to allow floating point in the data model

- Can handle disabling floating point in the data model — just don’t use floating point!

- Choose float32 instead of float: not a serialization choice, but a data model choice: only allow numbers that can be represented in a binary32

    - With preferred encoding, becomes a serialization choice!

Serialization variants are invisible at CDDL level

- CDDL defines data model
- Earlier drafts hinted at potential need for selection of representation variants
    - That need did not occur at the granularity of CDDL (or could be handled by making data model choices)
- Today: No hints of serialization choices in CDDL; data model only

Jim’s comments

- Editorial: Jim’s 1, 2, 9; (10 covered)

- 9:
  Reluctant to change section numbers at this point

Consistency-Checking a specification (Jim #4)

- It is possible to have elements in a specification that never match, or that take a lot of work to always match
- This may be a specifier's error, or it may be the result of composition of generic components
- ➔ "dead code" should not be a hard error
- Tool quality issue: emit warnings
- (Language issue: silence warnings ➔ freezer)

Items from Jim's review, cont

- (5) unwrap grammar is indeed a bit weird, unwrapping a map or array type yields a group, while unwrapping a tagged type yields a type
- Proposal: s/groupname/typename/, but keep in type2 production for the latter case:

  type2 = value ……
  / "~" S typename [genericarg]

Items from Jim's review

- (6) 3.10 could indeed say generics applies to groups as well as types
- (8) oops.
  Need to open a Precedence 8 with & and ~

Points of unhappiness (1)

- The regex issue
  - Solve by adding controls for additional regex types (in freezer)
- Limited reach of cuts: works well for map keys, does not cover `{ type: "foo", … }` constructs yet
  - Solve by extending cuts in the next version

Points of unhappiness (2)

- Grammar is context-insensitive
- Maps are context-sensitive, overlaid over grammar
- Cuts introduce sequence dependence into map specifications
    - Well, maybe that is the special, sweet-and-salty CDDL flavor

Then Ship it!

- Publish –04 based on this and maybe some more mailing list discussion by 2018-07-30

- Start a 2nd WGLC then to make sure no French people can read it? (Sorry about that. You have one day.)

- Check timing with AD.

CDDL:
A peek into the freezer

- (1) making CDDL as a data description language better within its envelope
  - E.g., issues about “specifying in the large” (naming, module systems)
  - Functional support for specific application domains (usually by adding controls)
- (2) adding functions beyond (case-insensitive) structural interoperability

CDDL:
A peek into the freezer (2)

- New functions:
  - Semantic augmentation (“semantic styles”)
    - Might work well with the desire for code generation
  - Going beyond context-free grammars
    - Co-Occurrence constraints
    - (Also: discussions at WISHI Hackathon about predicate-based extensions to CDDL; cf. Schematron vs. Relax-NG)

CDDL: Selectors and Semantics

- Most of the above can be done well by pairing **selectors** with semantics that is applied everywhere the selector matches
- Simplest kind of selector: CDDL rulename
- Need predicates in selectors, relative paths, …
- ➔ CBOR Path (and CBOR pointers?)
  - Don’t do another XPath, though
- Semantics could be for matching or for augmentation

CBOR (RFC 7049) bis

Concise Binary Object Representation

Carsten Bormann, 2018-07-17

Take CBOR to STD

- **Do not**: futz around
- **Do**:
  - Document interoperability
  - Make needed improvements in specification quality
    - At least fix the errata :-)
  - Check: Are all tags implemented interoperably?

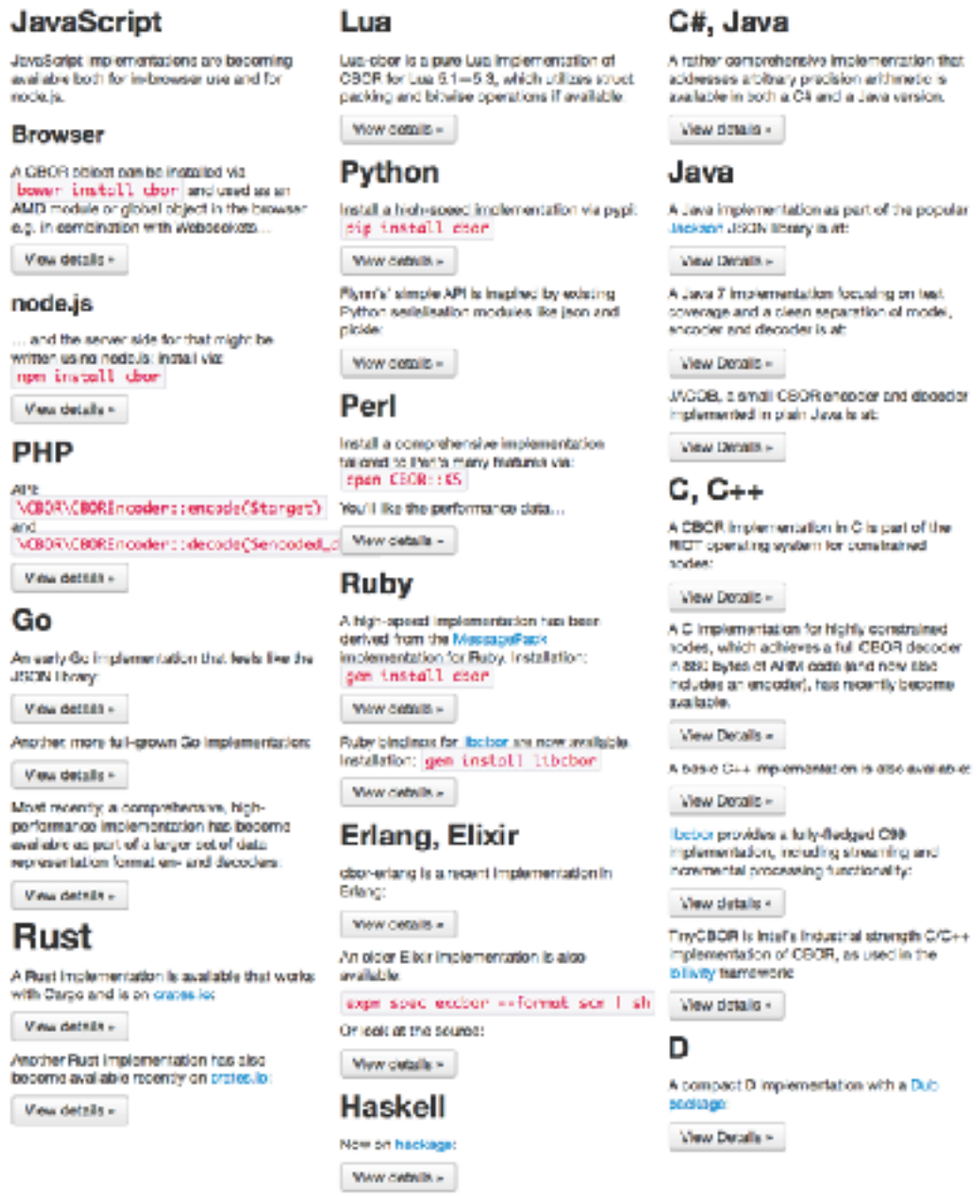
Take CBOR to STD

Process as defined by RFC 6410**:**

- independent interoperable implementations ✔
- no errata (oops) ✔ in draft
- no unused features [_]
- (if patented: licensing process) [N/A]

Implementations

- Parsing/generating CBOR easier than interfacing with application
  - Minimal implementation: 822 bytes of ARM code
- Different integration models, different languages
- > 50 implementations



http://cbor.io

7049bis has been “stable” for a while

- … while one author focused on getting up to speed again and fixing CDDL.

- To do:
  - See CBOR issues above (serialization invariants, consistent/preferred encoding, …)
  - Finish the discussion on the mailing list.
  - Fix github issues
  - Fix issues from IETF 101 minutes

Tag for ECMAscript Regex #23

- This is not the CDDL issue.
- Tag #35 “is for regular expressions in Perl Compatible Regular Expressions (PCRE) / JavaScript syntax [ECMA262]”.
- Proposal: Add a tag specifically for ECMAscript syntax.
- Sure, could do that.
- Is this a fix or a new feature?

Editorial: Make more use of (now defined) data model

- E.g., updating PR#11 (map keys)
- (This text is currently broken, a byte string is definitely *not* equivalent to a text string, and neither should be int and float; tagged items definitely differ from untagged ones)

Processing behavior on invalid input

- CBOR is careful to not require validity checks in a decoder except in strict mode
- CBOR does not require a decoder to be able to handle well-formed, but invalid input
- In effect, behavior with invalid input is not defined (but not in the C language “you are allowed to explode” kind of “undefined”)
- PR #17 proposes to define some of that behavior
- Maybe make it more explicit that it isn’t

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

Time for a “my favorite tags” document?

- Some Tags are defined in RFCs (e.g., RFC 8152 COSE, RFC 8392 CBOR Web Token (CWT) or in I-Ds that might become RFCs (draft-bormann-cbor-time-tag).

- Some are just registered, with a specification **somewhere**
  - Specifications in many places, varying forms, levels of details, etc.

- Objective: Collect definitions of “generally useful” registered tags in an RFC

- Great target date: 5 years of CBOR, October 2018 🤣