

# 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 IETF100

- Introduce cuts in maps so a matching key can be “reserved” even if its value does not match
- Move from PCRE to XSD regular expressions (and add discussion on why this may be not so great)
- Define matching rules in Appendix C

# Changes since IETF100

- Editorial:
  - Be more careful about “instances”
  - Fixes around examples
  - Get rid of some cobwebs

# draft-bormann-cbor-cddl- freezer-00

- Freezes issues that do not go in to CDDL 1.0:
  - “Cuts” beyond the simple “map validation” usage
  - Literal notation improvements (computed, tagged, regular expression, kitchen sink)
  - .pcre
  - Embedded ABNF
  - Module superstructure

# Lots of good editorial comments

- 4 Github issues
- Jim's review: 1, 2, 9; 6; 10
- Some comments encourage reverting previous improvements; need to find good balance

# Map matching

- Maps and arrays are described by **groups**
- Groups are grammars of **types**
- Grammars describe **linear** languages
- Maps are **unordered!**
- Array matching: Match **next** element
- Map matching: Match **any** member (i.e., drive parser from grammar!)

# “Map validation” issue

- CDDL semantics are generative (production system)
- All elements of a group in a map are equal
- Wildcard match (for extensibility) can enable what was not intended to be enabled 

```
{ ? 4=>text,  
  * uint=>any }
```
- How to create priority for “more specific”?



# cuts (better error messages)

```
a = ant / cat / elk  
ant = ["ant", ^ uint]  
cat = ["cat", ^ text]  
elk = ["elk", ^ float]
```

```
["ant", 47.11]
```

- Tool will not just tell you "can't match a", but "can't match rest of ant"
- Worth adding?

# Solution: Use cuts for map keys (only, for now)

- A cut after recognizing a map key cuts off any alternative matches

```
{ ? 4 ^ =>text,  
  * uint=>any }
```

- Make existing ":" a shortcut for "^ =>"

```
{ ? 4: text,  
  * uint=>any }
```

- Just that subset now in -02

# Map matching: To do?

- Are the remaining comments on map matching editorial?  
(I.e., text is not explaining this enough)
- Or is there a need for technical changes?

# Operator precedence

- Operator precedence is quite logical when considering **groups vs. types**
- But can surprise (e.g., Jim's 3 and 7). Regardless of precedence, ignoring group vs. type leads to syntax errors:  
e.g.,  $( (+a) / b )$  (can't do a type choice on a group)
- $(+ a / b)$  can be confusing, but is natural in, say,  $(? \text{foo} : \text{int} / \text{text})$
- uncomfortable with making sweeping late technical changes here
- Further editorially improve section 3.11 and some other examples
- Encourage a style that produces readable and immediately understandable grammars

# Items from Jim's review

- (4) this is more a comment on tool quality, but “dead code” should not be a hard error (and cuts that aren't matched don't do anything)
- (6) 3.10 could indeed say generics applies to groups as well as types
- (8) oops.  
Maybe open a Precedence 8 with & and ~

# 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/type name/, but keep in type2 production for the latter case:

type2 = value .....  
/ "~" S type name [genericarg]

# Terminology

- Need distinguishable terms
  - for the CBOR instance
  - for the CDDL grammar
- e.g., member (of a CBOR map)/element (array) vs. entry (of a CDDL group)
- But entry can be a composite group expression, too
- Maybe make clearer which terms are on which side

# CBOR (RFC 7049) bis

## Concise Binary Object Representation

Carsten Bormann, 2018-03-20



# 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
- > 45 implementations

The screenshot displays a grid of implementation cards for various programming languages. Each card includes a title, a brief description, and a 'View details' button. The languages listed are JavaScript, Lua, C#, Java, Python, Perl, Ruby, Erlang, Elixir, Haskell, Go, PHP, and D. The cards are arranged in three columns. The first column contains JavaScript, Lua, C#, Java, Python, Perl, Ruby, Erlang, Elixir, and Haskell. The second column contains Go, PHP, and D. The third column contains C#, Java, and D. The cards are arranged in three columns. The first column contains JavaScript, Lua, C#, Java, Python, Perl, Ruby, Erlang, Elixir, and Haskell. The second column contains Go, PHP, and D. The third column contains C#, Java, and D.

<http://cbor.io>

# draft-ietf-cbor-7049bis-01

- –00 had already fixed errata
- –01: 2017-10-14
- Amplification of chosen Simple encoding (1-byte only for false/true/null etc.)
- Add a changes section
  - Maybe sort this into fixes and new information?
- New: Section 2.5 CBOR Data Models

# CBOR data models

- Biggest failing of JSON: Data model now entirely implicit
- Observant reader could infer CBOR data model from RFC 7049
- Now more explicit: “generic data model” (as opposed to any specific data model realized in CBOR)
  - Unextended (basic) data model
  - Extension points: Simple, Tags
    - Pre-extension by false/true/null/undefined, 18 pre-defined tags
    - Further extension by Simple/Tag definitions (IANA)

# Why is a generic data model important?

- Generic data model enables the implementation of generic encoders and decoders
- An ecosystem of generic encoders and decoders
  - makes interoperability so much more likely
  - guides definition of specific data models

# “Expectations”

- “Batteries included”: not always appropriate
- But some of the pre-extensions are really basic
  - Which ones?
- Section 2.5 states **false/true/null** are *expected* to be provided in a generic encoder/decoder
- Anything else (Simple: **undefined**, 18 tags) is “truly optional and a matter of implementation quality”.

# New in -02

- Accidentally duplicated the data model text :-/
- Make more use of the fact that we now have data model terminology
- Separate integers and floating point values some more
- Clarify map key equivalence rules
  - To do: Needs to maintain separation of byte string and text string and of tagged values



# C14n

- OMG.
- Make sure it is clear that these are recommendations for an application to choose their c14n rules.

# C14n vs. generic serialization

- C14n may be application dependent
- Still want to offer c14n in a generic encoder (and possibly check for it in a decoder)
- How flexible can a generic canonicalizer be?

# C14n changes

- (Moved to recommendation for byte-wise lexicographic ordering; kept the old recommendation in, too, as historic.)  
Need to specify this more unambiguously?
- 3 variants for float c14n.  
Should we express preferences?
  - Proposal:  
prefer “shortest encoding”, as in other cases.
  - Same for bignums (i.e., canonicalize into int).

## Implementation matrix

fpalombini edited this page 14 days ago · 7 revisions

D = Decode E = Encode

Feature	TinyCBOR	node-cbor	cbor-ruby	impl4	PeterO.Cbor
Major type 0 (uint)	DE	DE			DE
Major type 1 (nint)	DE	DE			DE
Major type 2 (bstr)	DE	DE			DE
Major type 3 (tstr)	DE	DE			DE
Major type 4 (array)	DE	DE			DE
Major type 5 (map)	DE	DE			DE
Major type 6 (tag)	DE	DE			DE
Major type 7 (simple)	DE	DE			DE
Float16	DE	DE			D
Float32	DE	DE			DE
Float64	DE	DE			DE
Indefinite length array/map	DE	D			D
Indefinite length string	D	D			D[1]
Canonical CBOR	DE[2]	DE			D
Tag 0	DE[2]	D			DE
Tag 1	DE[2]	DE			DE
Tag 2	DE[2]	DE			DE
Tag 3	DE[2]	DE			DE
Tag 4	DE[2]	DE			DE
Tag 5	DE[2]	D			DE
Tag 21	DE[2]				
Tag 22	DE[2]				
Tag 23	DE[2]				
Tag 24	DE[2]				
Tag 32	DE[2]	DE			DE
Tag 33	DE[2]				
Tag 34	DE[2]				
Tag 35	DE[2]	DE			DE
Tag 36	DE[2]				
Tag 55799	DE[2]				

# Continuing work on implementation matrix

- <https://github.com/cbor-wg/CBORbis/wiki/Implementation-matrix>
- Need to fill in more columns
  - Certainly not for all 45 implementations :-)
- Who?

# CBOR tag definitions

Carsten Bormann, 2018-03-20

# 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

# CWT: CBOR Web Token

- JWT: JSON Web Token (RFC 7519)
  - Package **Claim Set** into JSON
  - Apply JOSE for Signing and Encryption
- CWT: Use CBOR and COSE instead of JSON and JOSE
- CWT can replace unstructured misuse of certificates for Claim Sets
- CBOR Tag 61 assigned;  
[draft-ietf-ace-cbor-web-token-15](#) now in RFC editor queue

# Status of Tags drafts

- **OID**: On charter, kitchen sink, expired.  
Needs work.
- **Array**: On charter, ready for adoption
- **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)?



# draft-jroatch-cbor-tags-07

- Provide tags for homogeneous arrays represented in byte strings

uint	sint	float
uint8	sint8	binary16
uint16	sint16	binary32
uint32	sint32	binary64
uint64	sint64	binary128

- Inspired by JavaScript
- 12×2: Both LSB and MSB first
- Reserves 24 contiguous tags
- Provides a tag for other homogeneous arrays
- Provides a tag for multidimensional arrays

# Array tags: 2-byte space?

- 2-byte Tags: Tags 24 to 255
- 2017: ~ 20 taken of 232; be careful with the space
- This is taking out 24 more — would this be a waste of 2-byte space?
  - **Yes**; arrays can be large; fine with 3-byte tags
  - **No**; arrays can also be small (e.g., RGB)
- Could partition 2 vs. 3 by size of basic type; ugly
- -07 does not take a position

# Reviews

- Paul: Need more MUSTs around endianness (last para of 2.1???)
- Jim: (1) would like type in extra byte and not tag [ceterum...]
- (2) need example for multi-dimensional out of non-TypedArray
- (3) multi-dimensional: do we need column major?
- (4) homogeneity is in the eye of the beholder (more examples)
- (5) what about the reserved Tag in the middle?
- (6) security considerations: dealing with large items

# Another proposal for array tags

- There is a registration request pending at IANA for what is pretty much the same thing (a bit less well-cooked)
  - Used (1+2)-byte tags for ease of registration
- Trying to contact author — maybe he wants to collaborate on finishing this?
- Go through with the registration very soon **now!**