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

- –05:
- Move Appendix H (Examples) to Appendix A (fill remaining gap)
- Align some terminology
- s/can produce/matches/g
- Explain the non-deterministic order of the map matching rules (3.5.3)
- Other editorial (some clarifications), fix typos

Comments after WGLC

- Kevin Braun: Representation Variants not covered (2018-08-30) — CDDL anticipates CBORbis a bit
  ➔ address with more technical clarification to 2.2.3. Representation Types?
- Kevin Braun: (Question about map matching), asks about matching algorithm
  But that is in 3.5.3. Non-deterministic order
  ➔ ? (maybe just clarify more)

IANA questions

- (There is one IANA registry: control operators)
  Clearly a new registry page.
  New category, or under “Concise Binary Object Representation (CBOR)”?
- Should the registry be ordered by name, alphabetically?
  - Pro: easier to find whether name is taken
    (search functions in browsers are hard to use)
  - Con: Can’t have grouping of related operators, e.g., .ne .eq .ge .lt .le .gt, which would naturally result from chronological order
- Bikeshed!

SECDIR review

Chris Lonvick:

- Reference RFC 3552 while talking about security considerations of protocols using CDDL in their specifications
- Do that as a normative reference (does not hurt)
- Add a normative reference to COSE because COSE references CDDL (?, the present presenter thinks that would be wrong)

GENART review

- Ines Robles: (many good editorial comments, including one quite embarrassing one, and:)

- (3) Should 10..0 have a meaning (maybe 0..10)?

1.0 Plan

- Submit –06 on 2018-11-13
- IESG Telechat on 2018-11-21
- React to IESG comments over Thanksgiving
- Declare it’s “May 2018”.

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)

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?

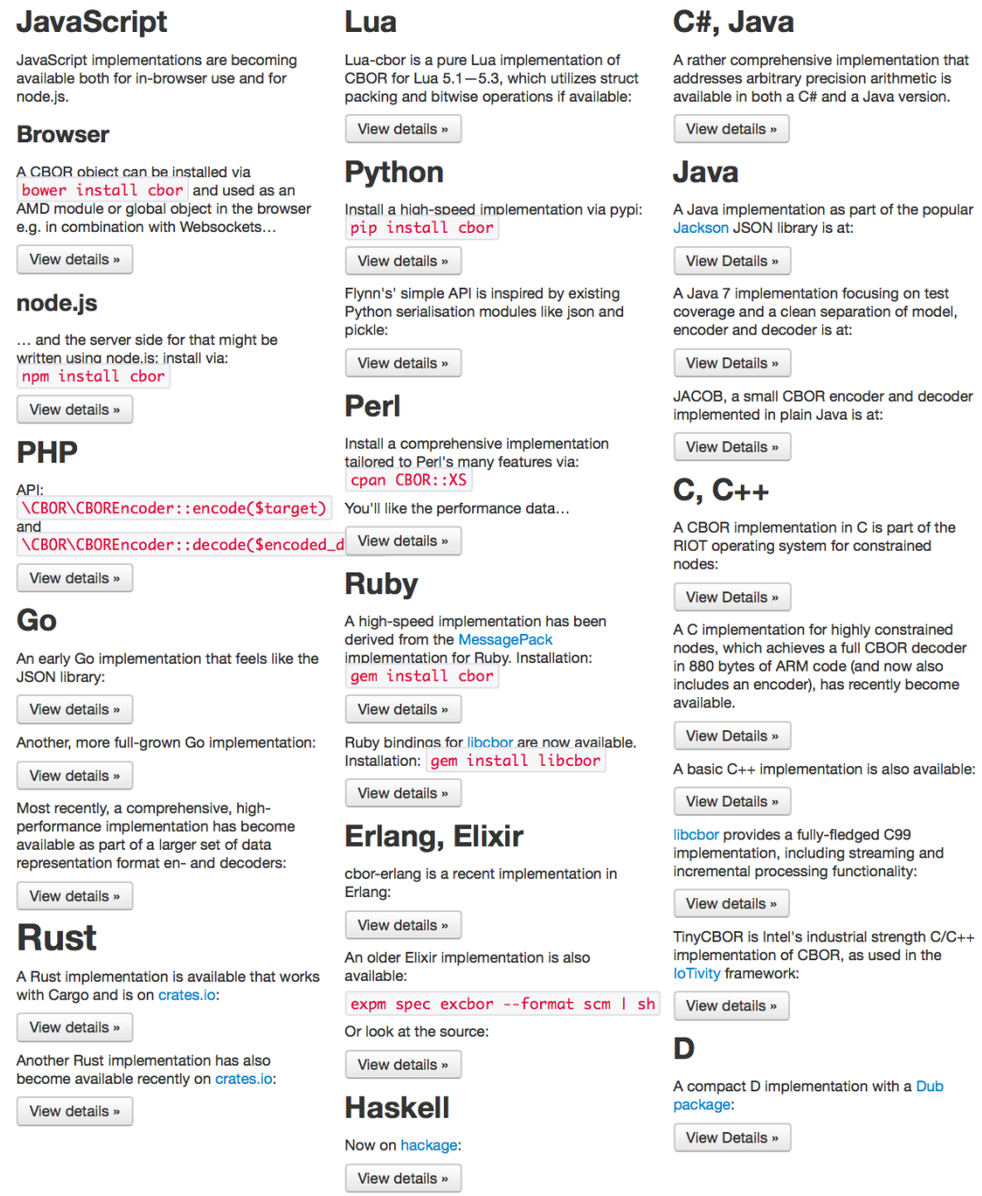
Take CBOR to STD

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



JavaScript

JavaScript implementations are becoming available both for in-browser use and for node.js.

Browser

A CBOR object can be installed via `bower install cbor` and used as an AMD module or global object in the browser e.g. in combination with Websockets…

View details »

node.js

… and the server side for that might be written using node.js: install via: `npm install cbor`

View details »

PHP

API: `\CBOR\CBOREncoder::encode($target)` and `\CBOR\CBOREncoder::decode($encoded_d`

View details »

Go

An early Go implementation that feels like the JSON library:

View details »

Another, more full-grown Go implementation:

View details »

Most recently, a comprehensive, high-performance implementation has become available as part of a larger set of data representation format en- and decoders:

View details »

Rust

A Rust implementation is available that works with Cargo and is on crates.io:

View details »

Another Rust implementation has also become available recently on crates.io:

View details »

Lua

Lua-cbor is a pure Lua implementation of CBOR for Lua 5.1—5.3, which utilizes struct packing and bitwise operations if available:

View details »

Python

Install a high-speed implementation via pypi: `pip install cbor`

View details »

Flynn's' simple API is inspired by existing Python serialisation modules like json and pickle:

View details »

Perl

Install a comprehensive implementation tailored to Perl's many features via: `cpan CBOR::XS`

You'll like the performance data…

View details »

Ruby

A high-speed implementation has been derived from the MessagePack implementation for Ruby. Installation: `gem install cbor`

View details »

Ruby bindings for libcbor are now available. Installation: `gem install libcbor`

View details »

Erlang, Elixir

cbor-erlang is a recent implementation in Erlang:

View details »

An older Elixir implementation is also available:

`expm spec excbor --format scm | sh`

Or look at the source:

View details »

Haskell

Now on hackage:

View details »

C#, Java

A rather comprehensive implementation that addresses arbitrary precision arithmetic is available in both a C# and a Java version.

View details »

Java

A Java implementation as part of the popular Jackson JSON library is at:

View Details »

A Java 7 implementation focusing on test coverage and a clean separation of model, encoder and decoder is at:

View Details »

JACOB, a small CBOR encoder and decoder implemented in plain Java is at:

View Details »

C, C++

A CBOR implementation in C is part of the RIOT operating system for constrained nodes:

View Details »

A C implementation for highly constrained nodes, which achieves a full CBOR decoder in 880 bytes of ARM code (and now also includes an encoder), has recently become available.

View Details »

A basic C++ implementation is also available:

View Details »

libcbor provides a fully-fledged C99 implementation, including streaming and incremental processing functionality:

View details »

TinyCBOR is Intel's industrial strength C/C++ implementation of CBOR, as used in the IoTivity framework:

View details »

D

A compact D implementation with a Dub package:

View Details »

http://cbor.io

Changes in -03

- Editorial: Use “argument” for the value resulting from additional information + 1, 2, 4, 8 bytes

- (Many other editorial, e.g., remove “data model” duplication)

- MUST NOT rely on ordering of items in map

Changes in -04 (1)

- Explain 0b/0x notation for byte strings some more
- Reference IEEE 754 (duuh)
- Remove UBJSON from Appendix E
  (has completely changed, no need to track this here, and it likely will change again)
- Explain that representation variants are not visible at data model level
- Be more specific for Tag 1 (Thanks, Laurence),
  but there is still continuing discussion on issue #35
- Specify preferred serialization, specifically for floating point

–04: map key equivalence

- Make it clear that map key equivalence is up to the application
- Define a base equivalence at the basic generic data model level
  - Application definitions can only be more restrictive, not less!
- Minimal restrictive definition mostly obvious, except:
  0.0 and –0.0 are equivalent
  (while NaN and -NaN depend on significand)

#37: Section 4 vs. MUST NOT

- Section 4 is intended as explanatory: How do you write protocol specifications that employ CBOR
- Map ordering MUST NOT doesn't quite fit in
- Move where? Could do it right in definition of MT5

IANA considerations

- RFC 7049: Very friendly to Specification required, friendly to FCFS

- May want to place some more conservation on 1+1-byte spaces (Simple, Tags) —
  Specification Required + some good reason?
  IETF review?
  Standards Action?

- May want to put 1+2 (Tags) under Specification Required

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, recently adopted

- **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-ietf-cbor-array-tags-00 (was draft-jroatch-cbor-tags)

- Provide tags for homogeneous arrays represented in byte strings
  - Inspired by JavaScript
  - 12×2: Both LSB and MSB first
  - Reserves 24 contiguous tags

```
+--------+--------+------------+
| uint   | sint   | float      |
+--------+--------+------------+
| uint8  | sint8  | binary16   |
| uint16 | sint16 | binary32   |
| uint32 | sint32 | binary64   |
| uint64 | sint64 | binary128  |
+--------+--------+------------+
```

- Provides a tag for other homogeneous arrays
- Provides a tag for multidimensional arrays

Array tags: 1+1-byte space?

- 1+1-byte Tags: Tags 24 to 255
- 2017: ~ 20 taken of 232, 2018: ~ 22;
  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 1+2-byte tags
  - **No**; arrays can also be small (e.g., RGB)
- Could partition 1+1 vs. 1+2 by size of basic type; ugly
- ietf…–00 does not take a position

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**!

Are we ready for 1+1-byte tags yet?