CBOR

10 documents in 52 minutes

Recently completed

- RFC9090: was draft-ietf-cbor-tags-oid
- draft-ietf-cbor-cddl-control:
  - Completed WGLC
    - made `.det` both-handed (tool updated)
  - Carsten: Process CA's remaining comments ➔ –04
  - Christian: Shepherd report, submit [~ next week]

WGLC ends today (MCR)

- draft-ietf-cbor-file-magic:
  - WGLC will need **more feedback** to complete
  - 55800 registered
  - need to go forward with content-format tags registration
- draft-ietf-cbor-network-addresses:
  - WGLC ending here, shepherd writeup ready
  - deflect "zone-id" comment (RFC 4007 `[CAFE:BEEF::1]%en0`)
    - would make it too easy to allow zone by mistake (YANG!)
    - zone id, zone name, zone number? + fluid 6847bis...
    - this instead belongs in draft-ietf-core-href

Recently adopted

- draft-ietf-cbor-time-tag
  - Just adopted recently
  - General direction:
    Do we want to add more or ship as a v1?
    - no rush: tags registered, in use in implementations
  - Next steps either way: Reviews needed.

Packed

Needs more independent implementation work!

Main issue: table building
Also: are the prefix/suffix/shared modes all we need?
(E.g., compare Kris Zyp's Records,
draft-bormann-lpwan-cbor-template)

Next steps CBOR-packed use cases

Plan: use the implementations to try out CBOR-packed on various use cases

On the bench: COSE C509 certificates (from John Mattsson)
(Unfortunately, found a bug in cbor-packed gem.)

Also: Try use cases with nested dictionaries

Document split?

We could go forward with the referencing tags (share, prefix, suffix)

— Without a "basic" table setup
— With a simple basic table setup (~ current document)

Or we could wait until we understand the whole picture of

— static dictionaries combining with in-instance ones
— combining CBOR out of separate packed instances

draft-bormann-cbor-notable-tags

Status: Individual, skeleton of a draft

- Background: Tags registry is an unsorted bucket
- NT to serve as an overview over important tags
- Can be more opinionated than registry?
  - WG opinions?
- Might help to "clean up" registry
  - No "recommended" columns etc. needed?

Freezer (CDDL evolution)

Most CDDL work since 8610 was done in control operators.
When do we touch the language itself again?

More control operators?

⊖ .pcre, .es2018re; iregexp?

⊖ .bits-bigendian, .bitfield, …

➔ Maybe later… (Can always add when needed)

Beyond control operators?

◎ Cuts
⊖ Literal syntax (including regexp literals)
👍 Clarifications
◎ More co-occurrence constraints
🙌 namespaces
🙌 cross-universe references

Some are rather independent of CDDL itself:

👍 alternative representations

➔ Work on these on independent timelines

How might name spaces look like (strawman syntax)

RFC8610.int ⟷ int (OK, that is already being imported)

RFC9090.oid ⟷ oid

```
;# export oid, roid, pen as RFC9090
oid = #6.111(bstr)
roid = #6.110(bstr)
pen = #6.112(bstr)
```

```
; unadorned, just import?
a = [RFC9090.oid]

;# import oid from RFC9090
a = [oid]
```

Add retroactive exporting to RFCs

Existing RFCs with CDDL in it:
presume an `export ...all... as RFCnnnn`
(Possibly also per-section exports RFC8610.D?)

Make those exports available in a focal space

New specs (including RFCs) can "include" from that
(And maybe "export" in a more considered way?)

Operations

"export":

(1) prefix:
add a namespace name
to "local" rulenames:

`oid` ➔ `RFC9090.oid`

(2) make that namespace
available to other specs

"import":

(1) include (prefixed)
definitions from a source
(2a) use as is: `RFC9090.oid`
(2b) unprefix ➔ `oid`

prelude processing:
include+unprefix from
Appendix D of RFC8610!

Decisions, Decisions:

How to find the document that exports a namespace
(IANA? Use by other SDOs? Internal use in an org?
How to transition between these states?)

Multiple documents exporting into one namespace
(Immutable RFC9090 namespace vs. "OID"-namespace?
Who manages mutable namespaces?)

Updates, revisions, versions, semver (oops)
`;# insert OID ~> 2.2` --[twiddle-wakka]

Map-like data

Status: draft-bormann-cbor-cddl-map-like-data-01
Waiting for further analysis from Emile Cornier.
But could adopt if we like the general direction.

— Map-like data are really lists/sets of pairs ➔
Should we also look at lists/sets in general?

`https://github.com/ecorm/cbor-map-like/blob/with-lists/README.md`

Records

https://github.com/kriszyp/cbor-records (not an I-D)
Two elements, hard to separate in discussion:

— Semantic Identification of tuple as record data

— Compression aspect (~ cbor-packed)

The specific proposal relies on ordering — may not be available to the application

https://www.rfc-editor.org/rfc/rfc8949.html#section-3.4-10

Can/should this be done in the CBOR-packed framework?