CBOR@IETF114

- CBOR tags: draft-ietf-cbor-time-tag (tag 1001..)
- Additions to the ecosystem:
  - draft-ietf-cbor-packed
  - RFC 9254 — YANG-CBOR
- CDDL evolution
  - "1.1" was RFC 9165
  - 2.0 roadmap

draft-ietf-cbor-time-tag

draft-ietf-cbor-time-tag-00 adopted 2021-05
"**no rush**": tags registered, in use in implementations

Waiting for SEDATE WG: extending RFC 3339 with hints

```
1996-12-19T16:39:57-08:00[America/Los_Angeles][u-ca=hebrew]
                          Time Zone Hint        Extension Suffix
```

SEDATE now converging.
draft-ietf-cbor-time-tag-01
adds parsed SEDATE hints:

```
1001({ 1: 851042397,
     -10: "America/Los_Angeles",
     -11: { "u-ca": "hebrew" }
})
```

SEDATE limits

SEDATE is chartered to stay within RFC3339 mold
- always UTC-referenced timestamp, no calendaring
- no "floating time" (compare tag 100 zoneless date)
- no overriding hints (but we have critical hints now)

RFC3339 added `-00:00`, incompatible with ISO 8601
(Numeric offset not mirrored into tag 1001)

Do we want to use our freedom to add one of these?

Floating Time

Tag 0, 1: UTC-referenced, always determined on that scale
Tag 100 (date, not time): zoneless
• Date interpreted in the reference time of the user ("floating")
Tag 1001: Like tag 0, 1, but more timescales (TAI)

NTPv5: likely to add timescale for local (floating) time (NTP then can use offset field to provide UTC time as well)

So just add another timescale, ignore offset?
Oh, and what about leap-smearing?

time-tag: Plan?

- Aim for synchronized publication with SEDATE (draft-ietf-sedate-datetime-extended)
- Watch NTPv5 as a role model for adding floating time
  - Maybe add leap-smearing timescale, too
  - Do not wait for NTPv5 completion, though
- Alternative: Do not add floating time to 1001
  - do this in separate tag

Packed

Recent addition of function tags as extension point

- with "join"/"ijoin" as extension point exercise

— Ongoing implementation work!

— Missing link: Tag equivalence principle

CBOR: Tag validity

- RFC 8949: Tags define shape of their valid tag content
  - top-down, structural control
  - recorded in tag registry

However:
Tags often define data that "stand in" for other data

- no way to record (or make use of) this intention

Where structural validity already hurt(s)

- RFC 8746 (Tagged Arrays):
  - some 25 tags creating and operating on arrays
  - defines term "Typed Arrays", uses that everywhere in this spec where an array is needed
  - approach doesn't embrace extension of "typed arrays" concept
- CBOR packed
  - reference tags "stand in" for the referenced data
  - Tag validator that operates before unpacking may break

Tag equivalence

draft-ietf-cbor-packed-07

- NEW: Tags can define what shape they stand in for (bottom-up: equivalence of tag to a shape)
- ➔ a data compression tag can stand in for a byte string
- ➔ a new typed array tag can stand in for a CBOR array
  - like the existing typed arrays already do for all intents and purposes
- cannot supply a larger domain than an outer tag would expect

tag equivalence: Plan?

draft-ietf-cbor-packed-07

- Pretty much a mandatory component of CBOR-packed
- Should not be over-used, requires caveats

1. Define it in CBOR-packed specification
   a. update 8949 from CBOR-packed/ b. just say, don't update
2. Put it into a separate document
   a. advance with CBOR-packed/ b. magic?
3. CDDL for tag equivalence?

RFC 9254: YANG-CBOR

YANG-CBOR defines representation of YANG data in CBOR
- YANG-XML: RFC 7950 (with YANG the language)
- YANG-JSON: RFC 7951

YANG: schema-affine ecosystem, now feeding into CBOR

YANG-CBOR does not "compress" XML-style data types (e.g., IP addresses, date/time)

YANG SIDs: pure-integer naming system, delta-coding gives major boost to efficiency
- allocation processes in IESG: CoRE-SID

Next Steps for CBOR WG (with CBOR turning 9 years)

1. **CBOR**: Stable; Extension point: Tag ecosystem
   - → guidelines for tag development
   - → "notable tags" with
     - Specification snapshots
     - usage guidance (e.g., "deprecating")
   - → highly reusable, shared tags
2. **CDDL**: Standardized "1.0". To be developed. → CDDL 2.0

Towards CDDL 2.0: "CDDL 1.1" ✓

CDDL's defined extension point: control operators

- application-specific control operators
  - e.g., RFC 9090 (ASN.1 Object Identifier data types)
- RFC 9165: "Additional control operators for CDDL"
  - .plus, .cat, .det: CDDL can compute (a bit)
  - .abnf, .abnfb: ABNF integration
  - .feature: beyond yes/no; indicate features being used

CDDL support infrastructure

Similar to diagnostic notation for CBOR , what do we need to write up to facilitate implementation and use?

Do we want to do more ABNF integration?

(E.g., section 7 of draft-bormann-cbor-cddl-freezer-09.txt)

```
cddlj = ["cddl", +rule]
rule = ["=" / "/=" / "//=", namep, type]
namep = ["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] /
  ["map", group] / ["ary", group] / ["tcho", 2*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]
```

CDDL 2.0

Discussion based on IETF 112

"CDDL 2.0"
== all evolution that needs to change the CDDL syntax

Highest Priorities:

— Annotation
— Composition

Composition

Multi-file CDDL specifications
Ability to build libraries

— "export" one or more rules

— import rule from other CDDL spec

➔ name the library
➔ name exported/imported rule

Namespacing

Give spec writer more control over rule names beyond aliasing: `oid = RFC9090.oid`

Make it look like prelude always has been in a namespace
(default import; cf. `using namespace std`)

Graciously handle some namespacing errors
(might be caused by revisions)

Alternatives

Use the same CDDL spec for alternative applications (e.g., JSON vs. CBOR)

Can do manually:

- RFC 8428 § 11 — lexical
- RFC 9165 § 4 — semantic (using .feature)

Ideally, should enable translation between alternatives!

Automation

Make libraries available from:

— RFCs and I-Ds (legacy and new conventions)

— IANA registries

— Similar non-IETF sources

Trigger this automation from a CDDL spec

Syntax

Ideal:
* CDDL 1.0 files fit right in (and are useful)
* CDDL 1.0 processors can do useful things with 2.0 files

We can use, via what would be a convention in 1.0:
- otherwise unused rules
- comments
- potentially: controls

Composition: Objective

1. Build a tool that can work on top of existing CDDL tools
   — e.g., as a "preprocessor" (try to avoid CPP's bugs)
2. Identify places where we do need to change CDDL itself

Make sure automation is available early

Preprocessor-like tool could be useful for ABNF as well

- Make sure it can supply ABNF that goes into CDDL

Annotation

1.0 processing model: Kernighan's car (yes/no)
Somewhat extended with `.feature` now

cddl tool: can **annotate** tree with rule names
No control by spec writer:

- Which rules are important, which are noise?
- Info beyond rule names (except .feature)
- Rule names cannot be related to real world

PSVI: Post-Schema Validation Instance

Use "Validation" process to:

- **augment** data with annotation information
- **transform** (e.g., filling in default values, **construct**)

What is the data model for a CBOR/CDDL PSVI?

- Attributes for data items
- Richer types for generic data model? (cf. tags!)

Define JSON/**cbor-diag**/YAML/CBOR representations

CDDL: timelines

implementation needs to closely follow design

IETF115 objectives (spec+impl):
- Usable prototype of most of composition
- First elements of annotation

IETF116 objective:
- 2.0 Spec technically complete (and implemented)
- Decide on any document splits, steps for publication

Longer-term objectives

Discuss automation with IANA

- ensure that web access interfaces are well-defined
- find a way to quantify additional load
  - enable good local caching
- possibly define rules for future registries
  - "automation-friendly registries"

cddlcatalog.org? (as in yangcatalog.org)
Make sure we have good interoperation between YANG and CDDL