CBOR

— CBOR tag: file-magic

— CBOR: packed

— CDDL 2.0

file-magic

55799(NNNN(data in CBOR item))

55800(NNNN('BOR')),
(data in CBOR sequence)

ct → NNNN

— 8-Byte prefix
for CBOR data items

— 12-Byte prefix
for CBOR sequences

— preallocated tag for
CBOR content-formats

Bug in -06

Examples cite non-CBOR content-formats!

Fix?

Would need bstr wrap to fit in 55799 (or 55800)
No longer constant prefix

Fix!

```
55801(NNNN('BOR')),
non-CBOR data
```

Almost, but not entirely **unlike** a CBOR sequence

Requires CBOR decoder to be able to decode one item and hand up raw data for rest of input

```
tag, rest_data = CBOR.decode_with_rest(file)
```

(or simply read first 12 bytes and decode that)

Plan for -07

Put in 55801

Examples:

- Keep non-CBOR content-format, but with 55801
- Add in CBOR content-format with 55799 and 55800

WGLC

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)

Packed: components

- Processing model: in-place reference chasing
- Referencing items:
  - flexible encoding within CBOR basic data model
- Table building
  - Nesting: Push (shift) model

Plan for Base document

- referrers (tags/simple) for: share, prefix, suffix
- table model (incl. push)
  - extensible for future kinds of referrers (record, template?)
- "basic" table setup tag (push to: share, prefix, suffix)
- Framework for defining more specific setup tags:
  - implicit reference (immutable table via per-application tag)
  - hashed reference (COSE alg, hash) to table in context
    - Needs hash input spec!
      (CBOR deterministic encoding of equivalent table tag)

CDDL 2.0

Highest Priority:

— Annotation
— Composition

Composition

Currently:

— single file = sequence of rules
— first rule (type!) is entry point (i.e., one data item only)

What do we need?

Ability to build libraries

— "export" one or more rules

— import rule from other CDDL spec

➔ name the library
➔ name exported/imported rule

import

implicit:

```
oid-tagged-data = [RFC9090.oid, bstr]

; short name needed?
oid = RFC9090.oid
```

explicit:

— identify library (possibly versioned: semver references)

— manage what names are introduced; potential conflicts

export

— name the library for export

— identify rule names actually exported

  — default set; can reach into library as needed

    💣 Yes, protection violation 👍

linkage?

- outside spec (e.g., cli parameters)
- give (overrideable) hint inside spec (URI?)

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, cddl-control § 4)

Don't keep this on the lexical level (like `#ifdef`)
Ideally, should enable translation between alternatives!

automation

Should be able to generate libraries from

— RFCs and I-Ds (legacy and new conventions)
— IANA registries
— Similar non-IETF sources

Trigger automation from a CDDL spec

syntax

— 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

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

Annotation: MVP

- rulename attributes:
  - select which rules actively annotate
  - associate rulename with some real-world concept (URI?)
- description attributes (from comments etc.)
- spec–writer-defined attributes (say, units)
- CDDL-generated (implicit) tags [unwrap!]

timelines

implementation needs to closely follow design

EOY 2021 objectives (spec+impl):
- Usable prototype of most of composition
- First elements of annotation

IETF113 objective:
- 2.0 Spec complete (and implemented)
- Decide on any document splits, steps for publication