Background: CBOR-associated languages

— CBOR = representation and interchange format (binary, concise, efficient)
  — low-level visualization in text as cbor-pretty (hex with comments)

Two associated textual languages:

— EDN (cbor-diag) ➔ **examples, diagnostics**
  — Text form for single instance (item/sequence), convert back and forth (cbor.me)
  — Derived from JSON, made more useful for humans, added binary, tags, …
— CDDL ➔ **specification, validation**
  — Describe specific data model (grammar)
  — Inspired by ABNF, can describe JSON, CBOR, CSV*

Agenda today

CBOR	tags/apps	EDN	CDDL
CDE			
	cbor-packed		

CDE: Status

- 2023-11-27 WG document
- lots of editorial tweaking, merged PRs
- 2024-10-16 draft-ietf-cbor-cde-06
  - introduces ALDR discussed at 2024-10-16 interim
  - references cbor-numbers for NaN appendix
- PRs:
  - stale PR #10 (editorial)?
- 2025-01-21 draft-ietf-cbor-cde-07
  - merged PR #23: Do not use "Profile" for ALDR

Would like to WGLC this now.

What is ALDR?

ALDR is Application-layer deterministic representation.

ALDR rules describe how applications represent application data so data with the same semantics always have the same representation in the generic CBOR data model.

(ALDR rules are not about the encoding layer.)

ALDR aren't new

Section 4.2.2 of RFC 8949 is mostly about ALDR

4.2.2 is as muddled as it is because it didn't clearly separate:

— encoding (say, float16 vs. float32) from
— representation in the data model (e.g., absence/presence of a tag)

Clearly identifying the latter as ALDR helps!

ALDR is a fact

An application that

— wants to: create deterministic CBOR messages
— has: data at rest that isn't already CBOR-formatted

needs rules for converting the data at rest to the generic CBOR data model.

If these rules exist in order to achieve determinism, **they are ALDR rules**.

ALDR are often

Data modeling (CDDL) can be used to express some ALDR

(e.g., for a timestamp:
choose tag 0, tag 1, a certain subset of tag 1001.)

Objective: achieve deterministic representation
➔ may not be explicit

(It may also fail, as in float vs. int for tag 1.)

Can you implement CDE without understanding ALDR?

Absolutely!

Essence of CDE design: well-defined boundary to ALDR

Can you use CDE without understanding ALDR?

You don't need to understand ALDR rules as such:

— if the rules already have been written
(e.g., Section 4 in RFC 9679)

— if your application data are already in CBOR form

— if you don't really care about determinism that much
(false negatives are OK, e.g. for testing)

ALDR example (from Section 4.2 of RFC 9679)

Note: [RFC9052] supports both compressed and uncompressed point representations.
For interoperability, implementations adhering to this specification MUST use the uncompressed point representation.
Therefore, the y-coordinate is expressed as a bstr.
If an implementation uses the compressed point representation, it MUST first convert it to the uncompressed form for the purpose of thumbprint calculation.

Should we have a term for ALDR?

Yes.

What do we get from hushing up ALDR?

One Appendix less.

Less awareness that application data often need ALDR care before they can be handed to CDE.

People that want to mutate CDE because it doesn't fulfill their (unnamed) ALDR needs (of course not!).

Inefficient discussions because a term is missing.

draft-ietf-cbor-packed

- Data items can have considerable redundancy
- Method 1: deflate, brotli, zstd:
  Apply (byte-wise) data compression on encoded representation
  - can have very good compression (e.g., lzma)
  - Relationship to uncompressed completely lost
    ➔ decompression = copy step, full expansion?
- Method 2: Save space at the data model level ("packing")
  - Consumer can directly work out of stored packed representation
  - ➔ Much more accessible to constrained implementation

What is draft-ietf-cbor-packed?

- Set of tags (also a few simple values)
  1. stable: reference tags/simples (universal)
  2. innovating: table building tags (often defined by applications)
- Used through adoption by an application protocol
- Why have a common set of reference tags/simples?
  - Enables generic ingesting implementations
  - Sharing tags between application protocols (saves precious short tag space)
  - Saves app protocol designers some reinvention

Packed vs. compression: Confusion

- Data compression algs (deflate, brotli, zstd; lzma, lz4)
  - well-advanced
    - hard to compete with just on compression ratio
  - easy to find
- IETF probably doesn't need another one
- Looking at Packed CBOR with a general data compression perspective can be confusing: That's just not the point!

Technical Discussion

— Idiom of using ranges of tags: unfamiliar to some
(one man's hack is other man's Internet)
(background: CBOR's single-content-item tags)

— Sizing the exact allocation of short tags/simples
("design for decades" curation of allocated resources!)
➔ Haggling???

— Reducing allocation footprint by losing efficiency?
(is losing "1.22 bytes" per reference a problem?)