

CBOR@IETF115

- Heads-up: Usable Formal Methods pRG
- 🕒 CBOR tags: draft-ietf-cbor-time-tag (tag 1001..)
- 🏠 DNS and CBOR
- ⚡ CDDL evolution (CDDL 2.0)
 - 2.0 plans
 - planning with other WGs



draft-ietf-cbor-time-tag

draft-ietf-cbor-time-tag-00 adopted 2021-05

"no rush": tags registered, in use in implementations

Aiming for synchronized publication with SEDATE WG Internet Extended Date/Time Format (IXDTF) extending RFC 3339 with [hints](#):

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

SEDATE converged.

[draft-ietf-cbor-time-tag-01](#)

added parsed SEDATE hints:

[-02](#) addressed TBDs

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



SEDATE WGGLC completed

IXDTF (SEDATE-06) WGGLC is now complete

Discussions with art@, sedate@, ntp@, tictoc@, cbor@ brought no further light

Waiting for RFC 3339 bugfix discussion

<https://mailarchive.ietf.org/arch/msg/sedate/sHH5agXVW82QIsYBZLr8dsYRjg>

synchronize IESG submission of SEDATE and time-tag

→ time-tag should be able to proceed with WGGLC



Outstanding issues

Plan was

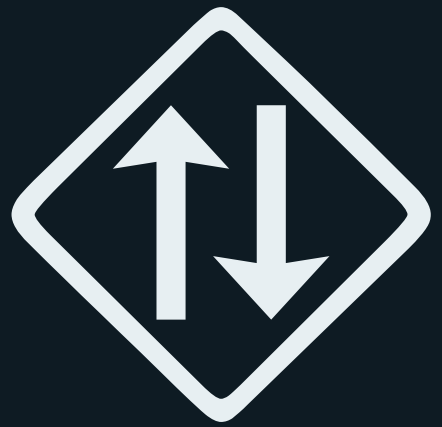
- Have a –03 ready for WGLC before IETF 115
- process WGLC at 115
- won't complete UT1 definition in time for WGLC (#9)
- Postpone #7 (planned/actual)
- do generate PR for #8 (number of unsigned map keys)



DNS and CBOR

See [Martine's slides](#) instead

- DNS over CoAP could use concise DNS format
- [draft-lenders-dns-cbor](#):
Promising new draft using CBOR-Packed
- Pushback from DNSOPS:
 - Swapping out DNS format for concise format
could lead to ossification if DNS innovation is not reflected
 - Good point, need to address
 - E.g., content negotiation, tunneling



CDDL 2.0

- Small language changes: Non-Literal Tag Numbers
- Update processing model; annotations?
- Modules, import, export; include
- RFC/I-D and IANA references



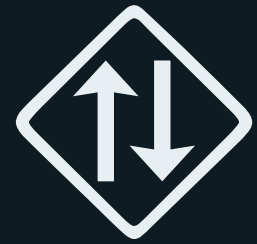
Non-literal tag numbers

```
type2 /= "#" "6" [ "." uint ] "(" S type S ")"
```

CDDL 2.0 extends this to

```
type2 /= "#" "6" [ "." tag-number ] "(" S type S ")"  
tag-number = uint / ("<" type ">")
```

```
ct-tag<content> = #6.<ct-tag-number>(content)  
ct-tag-number = 1668546817..1668612095  
; or use 0x63740101..0x6374FFFF
```



Processing model

RFC 8610: Validation (yes/no)

RFC 9165: add **features** (list of features used)

cddl tool: "annotated" instance

not currently influenceable from model

→ annotations similar to Relax_NG

(Brendan Moran's slides attached)

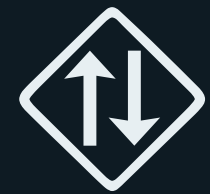
Next step: **transformation**

Generating a CBOR pull parser from CDDL

- What is it?
 - Code & “schema” data structure
 - Consumes values when needed
 - Does not parse to structure
 - Examines expected keys/types
 - “schema” has guidance flags to:
 - Repeat elements
 - Mark as optional
 - Unwrap CBOR in bstr
 - Pass to handler function
 - Handle key/value pairs
- What’s missing in CDDL
 - Entrypoints
 - Annotations for the parser (maybe too specific to code generation)
 - Handler function name
 - Extract-to-variable
 - Entrypoint-dependent-handling
 - E.g. just need one value; no need to unwrap bstrs
 - Ordered Multi-Maps
 - Imports
 - Currently have to concatenate CDDL
 - Lots of unused schema

Examples: annotations

- Challenge Response
 - Responder knows a priori that it receives challenges, not responses.
 - Its parser does not need to decode response message.
 - Entrypoint points to challenge.
- Use in other protocols
 - SUIP does not use every COSE structure.
 - Annotations could be used to strip unused structures



Module superstructure

Support for larger CDDL models:

— CDDL 1.0: Concatenating files (which files: external)

— CDDL 2.0: explicit references!

→ stay compatible with CDDL 1.0

```
;  
# export oid, roid, pen as RFC9090
```

↕ **Cross-universe (IANA) references**

```
cose-algorithm = int .iana ["cose", "algorithms", "value"]
```

```
//iana:registry[@id='algorithms']/iana:record/iana:value
```

→ <https://www.iana.org/assignments/cose/cose.xml>

⬆️⬆️ RFC references: Potential examples

RFC8610.int ↔ int

RFC9090.oid ↔ oid

namespace for the prelude (RFC8610.int)?

↕ explicitly interacting with namespaces

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

implicit import?

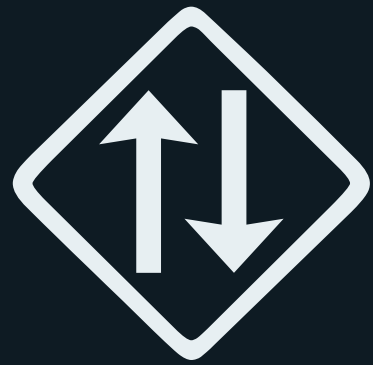
```
; unadorned, just import?  
a = [RFC9090.oid]
```

there also could be an explicit import syntax:

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

```
;/# include draft-ietf-cbor-time-tag-02.txt as time-tag  
event-start = time-tag.etime
```

Old RFCs: export ...all... as RFCnnnn
(per-section exports as in RFC8610.D for the prelude?)



Operations

- "export":
 1. prefix: add a namespace name to "local" rulenames: oid → RFC9090.oid
 2. make that namespace available to other specs
- "import": include (prefixed) definitions from a source
 1. use as is: RFC9090.oid
 2. unprefix: oid

Example: prelude processing — include+unprefix from Appendix D of RFC8610.
- "include"/"use": find files, turn into namespaces to import from

To be discussed

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, ...?

```
 ;# insert OID ~> 2.2 ; twiddle-wakka: this version or higher
```



ABNF is a lot like CDDL

ABNF = CDDL for flat sequences (of characters)

Integrated in CDDL via `.abnf/.abnfb`

CDDL 2.0:

Could provide many of the innovations for ABNF as well