CBOR Tags and Techniques for Object Identifiers

and how to use them

draft-bormann-cbor-tags-oid-06

IETF 98 CBOR

Chicago, IL, USA, 2017-03-30

Sean Leonard <dev+ietf@seantek.com>

Penango, Inc.

draft-bormann-cbor-tags-oid Objectives

- Update other protocols and data models to CBOR
- Lots of identifiers already exist, want to reuse rather than reinvent the wheel
- Support Object Identifiers [X.660] [X.680] natively in CBOR
- And provide guidance on how to use them properly
- Fill out and specify other tags

About Object Identifiers

- Managed hierarchy [X.660] based on positive integers or strings
- Open access: once arc is assigned, you can assign anything under it
- Variable-length (as short as one octet) and only* equality semantics
- Two widely adopted wire formats *(canonical forms!)*
  - Dotted decimal [RFC1776] (genesis [RFC1228]) (~3.3 bits per octet, ASCII-safe):
    `2.16.840.1.101.3.4.2.1`
  - BER encoding [X.690] (self-delimiting values, ~7 bits per octet):
    `60 86 48 01 65 03 04 02 01`
- Two widely adopted notations
  - ASN.1 value notation [X.680] (braces, optional strings):
    `{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) hashAlgs(2) sha256(1)}`
  - Dotted decimal notation/dot notation (see above)

OID Advantages

- Variable octets, can be very short
- Relative OID (“ROID”) permits assumed or factored base arc = shorter
- Language neutral (no hardcoded ASCII or UTF-8 strings)
- Concise vocabulary (sequence of non-negative integers)
- Hundreds of thousands already minted
- No transcription or mapping needed with other protocols, e.g., crypto, SNMP, MIB, LDAP, etc.
- OID Repository Database facilitates easy lookup

OID Disadvantages

- Used to be hard to get an arc
- Still not easy to get a really short arc *(but, ROID)*
- Perception of ASN.1 *(boo…)*
- OIDs can be very long
  - If ever longer than 16 octets, **stop** and use UUID
- Requires lookup: not self-describing *(but OID Repository makes easy)*
- "Not Native to CBOR" *(NIH?)*
  - For simple, closed enumerations, OIDs are **not** the job

Where We Are with the Draft

- OID tag «6» and ROID tag «7» assignments (proposed)
- Diagnostic notation (dotted decimal, ASN.1 value notation)
- When to use OIDs versus other types (integers, UTF-8 strings, UUIDs)

- OID (and ROID) arrays and maps, “tag factoring”, “tag stacking”

Beyond OIDs

- Sets and multisets in CBOR
  - CBOR has no native set type (unordered); ASN.1 has no native map type
  - Technique to simulate set as map of key items, value items are all integer 1 (or ≥ 1 for multiset)
  - Use case: express “capabilities” or “features” as sets of identifiers (OIDs)
- Tagging binary non-CBOR items (MIME, other binary formats)
- Validating CBOR data (with regular expressions)

Enumeration Decision Tree

- If modeling a particular data item that already exists, use the native data item's type (duh!) Otherwise:
- Natively signal CBOR data type ➔ CBOR tag.
- Limited, closed set of values ➔ integer.
- Human-readable on the wire (US-English?) ➔ UTF-8 string.
- Limited set of values controlled exclusively by IETF ➔ *consider* integer w/ registry.
- Open registration ➔ *consider* OID or UUID w/ optional registry.
- Create randomly or dynamically, or need exact size (16 octets) ➔ UUID.
- Otherwise ➔ OID.
  - Need shorter identifiers (fewer octets) or many options drawn from one place ➔ *consider* ROID + OID.

WG Stuff to Consider

- Adopt the draft
- Split the draft
- Formalize enumerations
  - Formalize UUID «37»?
  - Relationship to CDDL (i.e., as keys in map, like ASN.1 Open Type)
- A solution in search of problems? (Address)