

Preliminary

draft-ietf-cbor-serialization

Status and discussion, IETF 125, Shenzhen

Laurence Lundblade, IETF 125, Shenzhen

Draft Status — 04 published

(Same as March 4 interim)

- “ordinary” —> “preferred-plus”
- Remove appendix defining deterministic serializations for common tags
- Expanded introduction to include rationale
- Control operators are now `.prefp`, `prefpseq`, `.dtrm`, `.dtrmseq`
- Lots of wording of improvements, much of which is from Rohan’s review a few months back

Draft Status — 5 Open Issues, 1 PR

“TODO” means writing work is next step, not discussion, consensus or review

- TODO: Rohan requests additional security considerations; I've requested Rohan write them
- TODO: Improve COSE example appendix; a little more work needed
- Move Appendix B (data model determinism) to main body.
- Move NaN appendix elsewhere
- Examples / test vectors — This is the big one, discussion topic

- PR from Carsten on Section 1

Discussion:

Proposed Test Vectors and Examples

Test the three serializations

- NOT general CBOR testing, just of three serializations:
 - General
 - Preferred-plus
 - Deterministic
- CBOR library support for serializations will vary
 - One library may support all three, perhaps with distinct modes
 - Another library may support only one
 - Each library decides how to use the test data based on features

Test Data Item — The integer 0

```
{  
  "description": "The integer 0" ,  
  "edn-representations" : ["0"] ,  
  "general-serializations": [h'00',  
                             h'1800',  
                             h'190000',  
                             h'1a00000000',  
                             h'1b0000000000000000',  
                             h'C2420000',  
                             h'c240'],  
  "preferred-plus-serializations": [h'00'],  
  "deterministic-serialization": [h'00']  
}
```

Test Data Item — Map with 3 Items

```
{
  "description": "map with three items. Note that map order is never significant in th data model in CBOR",
  "edn-representations" : [
    "{1:\\"x\\", 2:\\"y\\", 3:\\"z\\"}",
    "{1:\\"x\\", 3:\\"z\\", 2:\\"y\\"}",
    "{2:\\"y\\", 3:\\"z\\", 1:\\"x\\"}",
    "{2:\\"y\\", 1:\\"x\\", 3:\\"z\\"}",
    "{3:\\"z\\", 1:\\"x\\", 2:\\"y\\"}",
    "{3:\\"z\\", 2:\\"y\\", 1:\\"x\\"}"
  ],
  "general-serializations": [
    h'a301617802617903617A',
    h'A301617803617A026179',
    h'A302617903617A016178',
    h'A302617901617803617A',
    h'A303617A016178026179',
    h'A303617A026179016178'
  ],
  "preferred-plus-serializations": [
    h'a301617802617903617A',
    h'A301617803617A026179',
    h'A302617903617A016178',
    h'A302617901617803617A',
    h'A303617A016178026179',
    h'A303617A026179016178'
  ],
  "deterministic-serialization": [h'a301617802617903617A']
}
```

Cover data types/values affected by serialization

Planned data types

- The integer 0
- The integer 3
- The integer -25
- The floating-point value 1.5 (which can be reduced to half-precision)
- Positive infinity
- Negative infinity
- The floating-point quiet NaN
- Floating-point NaN with payload 0x1ff
- The integer $2^{96} - 1$, which can only be represented by a big number
- The byte string of length 3 0x010203
- text string "hi there"
- the array [1, 2, 3]
- map with three items. Note that map order is never significant in the data model in CBOR

The Obvious Tests

- For each of the 3 serializations
 - Encode Test
 - Encode the value described in EDN and see that it encodes to one of the serializations
 - Decode Test
 - Decode all the serializations and get the expected value described in EDN

```
{  
  "description": "The integer 0" ,  
  "edn-representations" : ["0"] ,  
  "general-serializations": [h'00'  
                              h'1800'  
                              h'190000'  
                              h'1a000000000'  
                              h'1b00000000000000000000'  
                              h'C2420000'  
                              h'c240'],  
  "preferred-plus-serializations": [h'00'],  
  "deterministic-serialization": [h'00']  
}
```

Testing Checking Decoders (also obvious)

- Make sure that
 - Preferred-plus checking decoder errors out on all general serializations
 - Except those OK for preferred-plus
 - Deterministic checking decoder errors out on all preferred-plus and general serializations
 - Except those OK for deterministic

```
{  
  "description": "The integer 0" ,  
  "edn-representations" : ["0"] ,  
  "general-serializations": [h'00',  
                             h'1800',  
                             h'190000',  
                             h'1a00000000',  
                             h'1b0000000000000000',  
                             h'C2420000',  
                             h'c240'],  
  "preferred-plus-serializations": [h'00'],  
  "deterministic-serialization": [h'00']  
}
```

Non-Checking Decoders

- For example, feed general serialization into a preferred-plus decoder
 - May error out (e.g., indefinite-lengths fed to a decoder that is definite-lengths-only).
 - May decode successfully (e.g, non-shortest CBOR argument).
- Test Success/failure varies by CBOR library
 - Depends on features supported by library

```
{  
  "description": "The integer 0" ,  
  "edn-representations" : ["0"] ,  
  "general-serializations": [h'00',  
                             h'1800',  
                             h'190000',  
                             h'1a00000000',  
                             h'1b0000000000000000',  
                             h'C2420000',  
                             h'c240'],  
  "preferred-plus-serializations": [h'00'],  
  "deterministic-serialization": [h'00']  
}
```

NaN Payloads

- Q: How does a non-checking preferred-plus decoder respond to a NaN payload?
- A: Anyway it wants:
 - Error
 - Convert to a quiet NaN (CPU instructions does this)
 - Return it as a double or single with a NaN payload
 - Return it as a structure with a sign bit and payload bits