Network Working Group
Internet-Draft
Intended status: Informational
Expires: 3 April 2021

C. Bormann
Universität Bremen TZI
30 September 2020

Packed CBOR
draft-ietf-cbor-packed-00

Abstract

The Concise Binary Object Representation (CBOR, RFC 7049) is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation.

CBOR does not provide any forms of data compression. CBOR data items, in particular when generated from legacy data models often allow considerable gains in compactness when applying data compression. While traditional data compression techniques such as DEFLATE (RFC 1951) work well for CBOR, their disadvantage is that the receiver needs to unpack the compressed form to make use of data.

This specification describes Packed CBOR, a simple transformation of a CBOR data item into another CBOR data item that is almost as easy to consume as the original CBOR data item. A separate decompression step is therefore often not required at the receiver.

Note to Readers

This is an individual submission to the CBOR working group of the IETF, https://datatracker.ietf.org/wg/cbor/about/ (https://datatracker.ietf.org/wg/cbor/about/). Discussion currently takes places on the github repository https://github.com/cabo/cbor-packed (https://github.com/cabo/cbor-packed). If the CBOR WG believes this is a useful document, discussion is likely to move to the CBOR WG mailing list and a github repository at the CBOR WG github organization, https://github.com/cbor-wg (https://github.com/cbor-wg).

The current version is true work in progress; some of the sections haven't been filled in yet, and in particular, permission has not been obtained from tag definition authors to copy over their text.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 3 April 2021.

Table of Contents

1. Introduction 2
1.1. Terminology 3
2. Packed CBOR 3
2.1. Referencing Shared Items 4
2.2. Referencing Prefix Items 4
3. Discussion 5
4. IANA Considerations 5
5. Security Considerations 6
6. References 7
6.1. Normative References 7
6.2. Informative References 7
Appendix A. Example 8
Acknowledgements 9
Author's Address 9

1. Introduction

(TO DO, expand on text from abstract here; move references here and neuter them in the abstract as per Section 4.3 of [RFC7322].)

The specification defines a transformation from a Packed CBOR data item to the original CBOR data item; it does not define an algorithm for an actual packer. Different packers can differ in the amount of effort they invest in arriving at a minimal packed form.

Packed CBOR can employ two kinds of optimization:

* structure sharing: substructures (data items) that occur repeatedly in the original CBOR data item can be collapsed to a simple reference to a common representation of that data item. The processing required during consumption is limited to following that reference.

* prefix sharing: strings that share a prefix can be replaced by a reference to a common prefix plus the rest of the string. The processing required during consumption is similar to following the prefix reference plus that for an indefinite-length string.

A specific application protocol that employs Packed CBOR might allow both kinds of optimization or limit the representation to structure sharing only.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The definitions of [I-D.ietf-cbor-7049bis] apply. The term "byte" is used in its now customary sense as a synonym for "octet". Where bit arithmetic is explained, this document uses the notation familiar from the programming language C (including C++14's 0bnnn binary literals), except that, in the plain text form of this document, the operator "^" stands for exponentiation.

2. Packed CBOR

Packed CBOR is defined in CDDL [RFC8610] as in Figure 1:

```
Packed-CBOR = #6.6([rump, [*prefix], *shared])
rump = any
prefix = any
shared = any
```

Figure 1: Packed CBOR in CDDL

(This assumes the allocation of tag number 6, which is motivated further below. Note that the semantics of Tag 6 depend on its content: An integer turns the tag into a shared reference, a string into a prefix reference, and an array into a complete Packed CBOR data item as described above.)

The original CBOR data item can be reconstructed by recursively replacing shared and prefix references encountered in the rump by their defined values.

2.1. Referencing Shared Items

Shared items are stored in the third to last element of the array used as tag content for tag number 6, numbered starting by 2.

The shared data items are referenced by using the data items in Table 1. When reconstructing the original data item, such a reference is replaced by the referenced data item, which is then recursively unpacked.

reference	element number
Simple value 0-15	2-17
Tag 6(unsigned integer N)	18 + 2*N
Tag 6(negative integer N)	18 - 2*N - 1

Table 1: Referencing Shared Values

Taking into account the encoding, there are 16 one-byte references, 48 two-byte references, 512 three-byte references, 131072 four-byte references, etc. As integers can grow to very large (or small) values, there is no practical limit to how many shared items might be used in a Packed CBOR item.

2.2. Referencing Prefix Items

Shared items are stored in an array that is the second element of the array used as tag content for tag number 6. This array is indexed from 0.

Prefix data items are referenced by using the data items in Table 2. When reconstructing the original data item, such a reference is replaced by a string constructed from the referenced prefix data item (prefix, which might need to be recursively unpacked first)

concatenated with the tag content (suffix, again possibly recursively unpacked). The result gets the type of the suffix; this way a single prefix can be used to build both byte and text strings, depending on what type of suffix is being used.

reference	element number
Tag 6(suffix)	0
Tag 224-255(suffix)	1-32
Tag 28672-32767(suffix)	33-4128
Tag 1879048192-2147483647(suffix)	4129-268439584

Table 2: Referencing Prefix Values

Taking into account the encoding, there is one one-byte prefix reference, 32 two-byte references, 4096 three-byte references, and 268435456 five-byte references. 268439585 (2^(28)+2^(12)+2^(5)+2^(0)) is an artificial limit, but should be high enough that there, again, is no practical limit to how many prefix items might be used in a Packed CBOR item.

3. Discussion

This specification uses up a large number of Simple Values and Tags, in particular one of the rare one-byte tags and half of the one-byte simple values. Since the objective is compression, this is warranted if and only if there is consensus that this specific format could be useful for a wide area of applications, while maintaining reasonable simplicity in particular at the side of the consumer.

A maliciously crafted Packed CBOR data item might contain a reference loop. A consumer/decompressor MUST protect against that.

The current definition does nothing to help with packing CBOR sequences [RFC8742]; maybe it should.

Nesting packed CBOR data items is not useful; maybe it should.

4. IANA Considerations

In the registry [IANA.cbor-tags], IANA is requested to allocate the tags defined in Table 3.

Tag	Data Item	Semantics	Reference
6	array, integer, text string, byte string	Packed CBOR: packed/shared/ prefix	draft-bormann-cbor-packed
224-255	text string or byte string	Packed CBOR: prefix	draft-bormann-cbor-packed
28672-32767	text string or byte string	Packed CBOR: prefix	draft-bormann-cbor-packed
1879048192-2147483647	text string or byte string	Packed CBOR: prefix	draft-bormann-cbor-packed

Table 3: Values for Tag Numbers

In the registry [IANA.cbor-simple-values], IANA is requested to allocate the simple values defined in Table 4.

Value	Semantics	Reference
0-15	Packed CBOR: shared	draft-bormann-cbor-packed

Table 4: Simple Values

5. Security Considerations

The security considerations of RFC 7049 apply.

Loops in the Packed CBOR can be used as a denial of service attack, see Section 3.

As the unpacking is deterministic, packed forms can be used as signing inputs. (Note that if external dictionaries are added to cbor-packed, this requires additional consideration.)

6. References

6.1. Normative References

[I-D.ietf-cbor-7049bis]
Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", Work in Progress, Internet-Draft, draft-ietf-cbor-7049bis-15, 24 September 2020, <http://www.ietf.org/internet-drafts/draft-ietf-cbor-7049bis-15.txt>.

[IANA.cbor-simple-values]
IANA, "Concise Binary Object Representation (CBOR) Simple Values", <http://www.iana.org/assignments/cbor-simple-values>.

[IANA.cbor-tags]
IANA, "Concise Binary Object Representation (CBOR) Tags", <http://www.iana.org/assignments/cbor-tags>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

6.2. Informative References

[RFC7322] Flanagan, H. and S. Ginoza, "RFC Style Guide", RFC 7322, DOI 10.17487/RFC7322, September 2014, <https://www.rfc-editor.org/info/rfc7322>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/info/rfc8742>.

Appendix A. Example

The (JSON-compatible) CBOR data structure depicted in Figure 2, 400 bytes of binary CBOR, could lead to a packed CBOR data item depicted in Figure 3, 307 bytes. Note that this example does not lend itself to prefix compression.

```
{ "store": {
    "book": [
      { "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95
      },
      { "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99
      },
      { "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99
      },
      { "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95
    }
  }
}
```

Figure 2: Example original CBOR data item

```
6([{"store": {
      "book": [
        {simple(1): "reference", simple(2): "Nigel Rees",
         simple(3): "Sayings of the Century", simple(0): simple(5)},
        {simple(1): simple(4), simple(2): "Evelyn Waugh",
         simple(3): "Sword of Honour", simple(0): 12.99},
        {simple(1): simple(4), simple(2): "Herman Melville",
         simple(3): "Moby Dick", simple(6): "0-553-21311-3",
         simple(0): simple(5)},
        {simple(1): simple(4), simple(2): "J. R. R. Tolkien",
         simple(3): "The Lord of the Rings",
         simple(6): "0-395-19395-8", simple(0): 22.99}],
      "bicycle": {"color": "red", simple(0): 19.95}}},
   [],
   "price", "category", "author", "title", "fiction", 8.95, "isbn"])
   /  0          1         2         3         4       5      6   /
```

Figure 3: Example packed CBOR data item

TBD: Do this for a W3C Thing Description again to get better packing and to exercise prefix compression...

Acknowledgements

CBOR packing was originally invented with the rest of CBOR, but did not make it into [RFC7049]. Various attempts to come up with a specification over the years didn't proceed. In 2017, Sebastian Käbisch proposed investigating compact representations of W3C Thing Descriptions, which prompted the author to come up with essentially the present design.

Author's Address

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 1 May 2021

C. Bormann
Universität Bremen TZI
S. Leonard
Penango, Inc.

28 October 2020

Concise Binary Object Representation (CBOR) Tags for Object Identifiers
draft-ietf-cbor-tags-oid-02

Abstract

The Concise Binary Object Representation (CBOR, draft-ietf-cbor-7049bis) is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation.

The present document defines CBOR tags for object identifiers (OIDs). It is intended as the reference document for the IANA registration of the CBOR tags so defined.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 1 May 2021.

Copyright Notice

Table of Contents

1. Introduction 2
2. Object Identifiers 3
3. Examples 5
4. Discussion 6
5. Tag Factoring with OID Arrays and Maps 6
6. Applications and Examples of OIDs 7
7. CDDL Control Operators 9
8. IANA Considerations 9
9. Security Considerations 10
10. References 11
10.1. Normative References 11
10.2. Informative References 11
Appendix A. Change Log 12
Acknowledgments 13
Authors' Addresses 13

1. Introduction

The Concise Binary Object Representation (CBOR, [I-D.ietf-cbor-7049bis]) provides for the interchange of structured data without a requirement for a pre-agreed schema. [I-D.ietf-cbor-7049bis] defines a basic set of data types, as well as a tagging mechanism that enables extending the set of data types supported via an IANA registry.

The present document defines CBOR tags for object identifiers (OIDs, [X.660]), which many IETF protocols carry. The ASN.1 Basic Encoding Rules (BER, [X.690]) specify binary encodings of both (absolute) object identifiers and relative object identifiers. The contents of these encodings (the "value" part of BER's type-length-value structure) can be carried in a CBOR byte string. This document defines two CBOR tags that cover the two kinds of ASN.1 object identifiers encoded in this way. The tags can also be applied to arrays and maps to efficiently tag all elements of an array or all keys of a map. It is intended as the reference document for the IANA registration of the tags so defined.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terminology of draft-ietf-cbor-7049bis applies; in particular the term "byte" is used in its now customary sense as a synonym for "octet".

2. Object Identifiers

The International Object Identifier tree [X.660] is a hierarchically managed space of identifiers, each of which is uniquely represented as a sequence of unsigned integer values [X.680]. (These integer values are called "primary integer values" in X.660 because they can be accompanied by (not necessarily unambiguous) secondary identifiers. We ignore the latter and simply use the term "integer values" here, occasionally calling out their unsignedness.)

While these sequences can easily be represented in CBOR arrays of unsigned integers, a more compact representation can often be achieved by adopting the widely used representation of object identifiers defined in BER; this representation may also be more amenable to processing by other software that makes use of object identifiers.

BER represents the sequence of unsigned integers by concatenating self-delimiting [RFC6256] representations of each of the integer values in sequence.

ASN.1 distinguishes absolute object identifiers (ASN.1 Type "OBJECT IDENTIFIER"), which begin at a root arc ([X.660] Clause 3.5.21), from relative object identifiers (ASN.1 Type "RELATIVE-OID"), which begin relative to some object identifier known from context ([X.680] Clause 3.8.63). As a special optimization, BER combines the first two integers in an absolute object identifier into one numeric identifier by making use of the property of the hierarchy that the first arc has only three integer values (0, 1, and 2), and the second arcs under 0 and 1 are limited to the integer values between 0 and 39. (The root arc "joint-iso-itu-t(2)" has no such limitations on its second arc.) If X and Y are the first two integer values, the single integer value actually encoded is computed as:

```
X * 40 + Y
```

The inverse transformation (again making use of the known ranges of X and Y) is applied when decoding the object identifier.

Since the semantics of absolute and relative object identifiers differ, this specification defines two tags, collectively called the "OID tags" here:

Tag TBD111: tags a byte string as the [X.690] encoding of an absolute object identifier (simply "object identifier" or "OID").

Tag TBD110: tags a byte string as the [X.690] encoding of a relative object identifier (also "relative OID"). Since the encoding of each number is the same as for [RFC6256] Self-Delimiting Numeric Values (SDNVs), this tag can also be used for tagging a byte string that contains a sequence of zero or more SDNVs.

2.1. Requirements on the byte string being tagged

To form a valid tag, a byte string tagged by TBD111 or TBD110 MUST be a syntactically valid BER representation of an object identifier: A concatenation of zero or more SDNV values, where each SDNV value is a sequence of one or more bytes that all have their most significant bit set, except for the last byte, where it must be unset; the first byte of each SDNV cannot be 0x80 (which would be a leading zero in SDNV's base-128 arithmetic).

In other words:

* its first byte, and any byte that follows a byte that has the most significant bit unset, MUST NOT be 0x80 (this requirement requires expressing the integer values in their shortest form, with no leading zeroes)

* its last byte MUST NOT have the most significant bit set (this requirement excludes an incomplete final integer value)

If either of these invalid conditions are encountered, the tag is invalid.

[X.680] restricts RELATIVE-OID values to have at least one arc, i.e., their encoding would have at least one SDNV. This specification permits empty relative object identifiers; they may still be excluded by application semantics.

To facilitate the search for specific object ID values, it is RECOMMENDED that definite length encoding (see Section 3.2.3 of [I-D.ietf-cbor-7049bis]) is used for the byte strings used as tag content for these tags.

The valid set of byte strings can also be expressed using regular expressions on bytes, using no specific notation but resembling [PCRE]. Unlike typical regular expressions that operate on character sequences, the following regular expressions take bytes as their domain, so they can be applied directly to CBOR byte strings.

For byte strings with tag TBD111:

```
"/^(([\x81-\xFF][\x80-\xFF]*)?[\x00-\x7F])+$/"
```

For byte strings with tag TBD110:

```
"/^(([\x81-\xFF][\x80-\xFF]*)?[\x00-\x7F])*$/"
```

A tag with tagged content that does not conform to the applicable regexp is invalid.

3. Examples

3.1. Encoding of the SHA-256 OID

ASN.1 Value Notation: { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm(4) hashalgs(2) sha256(1) }

Dotted Decimal Notation: 2.16.840.1.101.3.4.2.1

```
06                                  # UNIVERSAL TAG 6
   09                               # 9 bytes, primitive
      60 86 48 01 65 03 04 02 01    # X.690 Clause 8.19
#      |   840  1  |  3  4  2  1     show component encoding
#   2.16         101
```

Figure 1: SHA-256 OID in BER

```
D8 6F                               # tag(111)
   49                               # 0b010_01001: mt 2, 9 bytes
      60 86 48 01 65 03 04 02 01    # X.690 Clause 8.19
```

Figure 2: SHA-256 OID in CBOR

3.2. Encoding of a MIB Relative OID

Given some OID (e.g., "lowpanMib", assumed to be "1.3.6.1.2.1.226" [RFC7388]), to which the following is added:

ASN.1 Value Notation: { lowpanObjects(1) lowpanStats(1) lowpanOutTransmits(29) }

```
Dotted Decimal Notation:  .1.1.29

0D                                # UNIVERSAL TAG 13
   03                             # 3 bytes, primitive
      01 01 1D                    # X.690 Clause 8.20
#      1  1 29                      show component encoding
```

Figure 3: MIB relative object identifier, in BER

```
D8 6E                             # tag(110)
   43                             # 0b010_01001: mt 2 (bstr), 3 bytes
      01 01 1D                    # X.690 Clause 8.20
```

Figure 4: MIB relative object identifier, in CBOR

This relative OID saves seven bytes compared to the full OID encoding.

4. Discussion

Staying close to the way object identifiers are encoded in ASN.1 BER makes back-and-forth translation easy; otherwise we would choose a more efficient encoding. Object identifiers in IETF protocols are serialized in dotted decimal form or BER form, so there is an advantage in not inventing a third form. Also, expectations of the cost of encoding object identifiers are based on BER; using a different encoding might not be aligned with these expectations. If additional information about an OID is desired, lookup services such as the OID Resolution Service (ORS) [X.672] and the OID Repository [OID-INFO] are available.

5. Tag Factoring with OID Arrays and Maps

OID tags can tag byte strings (as discussed above), but also CBOR arrays and maps. The idea in the latter case is that the tag is factored out from each individual item in the container; the tag is placed on the array or map instead.

When an OID tag is applied to an array, it means that the respective tag is imputed to all elements of the array that are byte strings, arrays, or maps. (There is no effect on other elements, including text strings or tags.) For example, when an array is tagged with TBD111, every array element that is a byte string is an OID, and every element that is an array or map is in turn treated as discussed here.

When an OID tag is applied to a map, it means that the respective tag is imputed to all keys in the map that are byte strings, arrays, or maps; again, there is no effect on keys of other major types. Note that there is also no effect on the values in the map.

As a result of these rules, tag factoring in nested arrays and maps is supported. For example, a 3-dimensional array of OIDs can be composed by using a single TBD111 tag containing an array of arrays of arrays of byte strings. All such byte strings are then considered OIDs.

6. Applications and Examples of OIDs

6.1. X.500 Distinguished Name

Consider the X.500 distinguished name:

Attribute Types	Attribute Values
c (2.5.4.6)	US
l (2.5.4.7) s (2.5.4.8) postalCode (2.5.4.17)	Los Angeles CA 90013
street (2.5.4.9)	532 S Olive St
businessCategory (2.5.4.15) buildingName (0.9.2342.19200300.100.1.48)	Public Park Pershing Square

Table 1: Example X.500 Distinguished Name

Table 1 has four "relative distinguished names" (RDNs). The country and street RDNs are single-valued. The second and fourth RDNs are multi-valued.

The equivalent representations in CBOR diagnostic notation and CBOR are:

```
111([{ h'550406': "US" },
     { h'550407': "Los Angeles", h'550408': "CA",
       h'550411': "90013" },
     { h'550409': "532 S Olive St" },
     { h'55040f': "Public Park",
       h'0992268993f22c640130': "Pershing Square" }])
```

Figure 5: Distinguished Name, in CBOR Diagnostic Notation

```
d8 6f                                  # tag(111)
   84                                  # array(4)
      a1                               # map(1)
         43 550406                     # 2.5.4.6 (4)
         62                            # text(2)
            5553                       # "US"
      a3                               # map(3)
         43 550407                     # 2.5.4.7 (4)
         6b                            # text(11)
            4c6f7320416e67656c6573     # "Los Angeles"
         43 550408                     # 2.5.4.8 (4)
         62                            # text(2)
            4341                       # "CA"
         43 550411                     # 2.5.4.17 (4)
         65                            # text(5)
            3930303133                 # "90013"
      a1                               # map(1)
         43 550409                     # 2.5.4.9 (4)
         6e                            # text(14)
            3533322053204f6c697665205374   # "532 S Olive St"
      a2                               # map(2)
         43 55040f                     # 2.5.4.15 (4)
         6b                            # text(11)
            5075626c6963205061726b     # "Public Park"
         4a 0992268993f22c640130    # 0.9.2342.19200300.100.1.48 (11)
         6f                            # text(15)
            5065727368696e6720537175617265 # "Pershing Square"
```

Figure 6: Distinguished Name, in CBOR (109 bytes)

(This example encoding assumes that all attribute values are UTF-8 strings, or can be represented as UTF-8 strings with no loss of information.)

7. CDDL Control Operators

CDDL specifications may want to specify the use of SDNVs or SDNV sequences (as defined for the tag content for TBD110). This document introduces two new control operators that can be applied to a target value that is a byte string:

* ".sdnv", with a control type that contains unsigned integers. The byte string is specified to be encoded as an [RFC6256] SDNV (BER encoding) for the matching values of the control type.

* ".sdnvseq", with a control type that contains arrays of unsigned integers. The byte string is specified to be encoded as a sequence of [RFC6256] SDNVs (BER encoding) that decodes to an array of unsigned integers matching the control type.

* ".oid", like ".sdnvseq", except that the X*40+Y translation for absolute OIDs is included (see Figure 8).

Figure 7 shows an example for the use of ".sdnvseq" for a part of a structure using OIDs that could be used in Figure 6; Figure 8 shows the same with the ".oid" operator.

```
country-rdn = {country-oid => country-value}
country-oid = bytes .sdnvseq [85, 4, 6]
country-value = text .size 2
```

Figure 7: Using .sdnvseq

```
country-rdn = {country-oid => country-value}
country-oid = bytes .oid [2, 5, 4, 6]
country-value = text .size 2
```

Figure 8: Using .oid

Note that the control type need not be a literal; e.g., "bytes .oid [2, 5, 4, *uint]" matches all OIDs inside OID arc 2.5.4, "attributeType".

8. IANA Considerations

8.1. CBOR Tags

IANA is requested to assign the CBOR tags in Table 2, with the present document as the specification reference.

Tag	Data Item	Semantics
TBD111	byte string or array or map	object identifier (BER encoding)
TBD110	byte string or array or map	relative object identifier (BER encoding); SDNV [RFC6256] sequence

Table 2: Values for New Tags

8.2. CDDL Control Operators

IANA is requested to assign the CDDL Control Operators in Table 3, with the present document as the specification reference.

Name	Reference
.sdnv	[this document, Section 7]
.sdnvseq	[this document, Section 7]
.oid	[this document, Section 7]

Table 3: New CDDL Operators

9. Security Considerations

The security considerations of [I-D.ietf-cbor-7049bis] apply.

The encodings in Clauses 8.19 and 8.20 of [X.690] are quite compact and unambiguous, but MUST be followed precisely to avoid security pitfalls. In particular, the requirements set out in Section 2.1 of this document need to be followed; otherwise, an attacker may be able to subvert a checking process by submitting alternative representations that are later taken as the original (or even something else entirely) by another decoder supposed to be protected by the checking process.

OIDs and relative OIDs can always be treated as opaque byte strings. Actually understanding the structure that was used for generating them is not necessary, and, except for checking the structure requirements, it is strongly NOT RECOMMENDED to perform any processing of this kind (e.g., converting into dotted notation and

back) unless absolutely necessary. If the OIDs are translated into other representations, the usual security considerations for non-trivial representation conversions apply; the integer values are unlimited in range.

10. References

10.1. Normative References

[I-D.ietf-cbor-7049bis]
Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", Work in Progress, Internet-Draft, draft-ietf-cbor-7049bis-16, 30 September 2020, <http://www.ietf.org/internet-drafts/draft-ietf-cbor-7049bis-16.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6256] Eddy, W. and E. Davies, "Using Self-Delimiting Numeric Values in Protocols", RFC 6256, DOI 10.17487/RFC6256, May 2011, <https://www.rfc-editor.org/info/rfc6256>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[X.660] International Telecommunications Union, "Information technology Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree", ITU-T Recommendation X.660, July 2011.

[X.680] International Telecommunications Union, "Information technology Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, August 2015.

[X.690] International Telecommunications Union, "Information technology ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, August 2015.

10.2. Informative References

[OID-INFO] Orange SA, "OID Repository", 2016, <http://www.oid-info.com/>.

[PCRE] Ho, A., "PCRE - Perl Compatible Regular Expressions", 2018, <http://www.pcre.org/>.

[RFC7388] Schoenwaelder, J., Sehgal, A., Tsou, T., and C. Zhou, "Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 7388, DOI 10.17487/RFC7388, October 2014, <https://www.rfc-editor.org/info/rfc7388>.

[X.672] International Telecommunications Union, "Information technology Open systems interconnection Object identifier resolution system", ITU-T Recommendation X.672, August 2010.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. Changes from -01 to -02

Minor editorial changes, remove some remnants, ready for WGLC.

A.2. Changes from -00 to -01

Clean up OID tag factoring.

A.3. Changes from -07 (bormann) to -00 (ietf)

Resubmitted as WG draft after adoption.

A.4. Changes from -06 to -07

Reduce the draft back to its basic mandate: Describe CBOR tags for what is colloquially know as ASN.1 Object IDs.

A.5. Changes from -05 to -06

Refreshed the draft to the current date ("keep-alive").

A.6. Changes from -04 to -05

Discussed UUID usage in CBOR, and incorporated fixes proposed by Olivier Dubuisson, including fixes regarding OID nomenclature.

A.7. Changes from -03 to -04

Changes occurred based on limited feedback, mainly centered around the abstract and introduction, rather than substantive technical changes. These changes include:

* Changed the title so that it is about tags and techniques.
* Rewrote the abstract to describe the content more accurately, and to point out that no changes to the wire protocol are being proposed.
* Removed "ASN.1" from "object identifiers", as OIDs are independent of ASN.1.
* Rewrote the introduction to be more about the present text.
* Proposed a concise OID arc.
* Provided binary regular expression forms for OID validation.
* Updated IANA registration tables.

A.8. Changes from -02 to -03

Many significant changes occurred in this version. These changes include:

* Expanded the draft scope to be a comprehensive CBOR update.
* Added OID-related sections: OID Enumerations, OID Maps and Arrays, and Applications and Examples of OIDs.
* Added Tag 36 update (binary MIME, better definitions).
* Added stub/experimental sections for X.690 Series Tags (tag <<X>>) and Regular Expressions (tag 35).
* Added technique for representing sets and multisets.
* Added references and fixed typos.

Acknowledgments

Jim Schaad provided a review of this document.

Authors' Addresses

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Sean Leonard
Penango, Inc.
5900 Wilshire Boulevard
21st Floor
Los Angeles, CA, 90036
United States of America

Email: dev+ietf@seantek.com
URI: http://www.penango.com/

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 21 May 2021

C. Bormann
Universität Bremen TZI
S. Leonard
Penango, Inc.

17 November 2020

Concise Binary Object Representation (CBOR) Tags for Object Identifiers
draft-ietf-cbor-tags-oid-03

Abstract

The Concise Binary Object Representation (CBOR, draft-ietf-cbor-7049bis) is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation.

The present document defines CBOR tags for object identifiers (OIDs). It is intended as the reference document for the IANA registration of the CBOR tags so defined.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 21 May 2021.

Copyright Notice

Table of Contents

1. Introduction 2
2. Object Identifiers 3
3. Examples 5
4. Discussion 6
5. Tag Factoring with OID Arrays and Maps 6
6. Applications and Examples of OIDs 7
7. CDDL Control Operators 9
8. CDDL typenames 9
9. IANA Considerations 10
10. Security Considerations 10
11. References 11
11.1. Normative References 11
11.2. Informative References 12
Appendix A. Change Log 12
Acknowledgments 14
Authors' Addresses 14

1. Introduction

The Concise Binary Object Representation (CBOR, [I-D.ietf-cbor-7049bis]) provides for the interchange of structured data without a requirement for a pre-agreed schema. [I-D.ietf-cbor-7049bis] defines a basic set of data types, as well as a tagging mechanism that enables extending the set of data types supported via an IANA registry.

The present document defines CBOR tags for object identifiers (OIDs, [X.660]), which many IETF protocols carry. The ASN.1 Basic Encoding Rules (BER, [X.690]) specify binary encodings of both (absolute) object identifiers and relative object identifiers. The contents of these encodings (the "value" part of BER's type-length-value structure) can be carried in a CBOR byte string. This document defines two CBOR tags that cover the two kinds of ASN.1 object identifiers encoded in this way. The tags can also be applied to arrays and maps to efficiently tag all elements of an array or all keys of a map. It is intended as the reference document for the IANA registration of the tags so defined.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terminology of draft-ietf-cbor-7049bis applies; in particular the term "byte" is used in its now customary sense as a synonym for "octet".

2. Object Identifiers

The International Object Identifier tree [X.660] is a hierarchically managed space of identifiers, each of which is uniquely represented as a sequence of unsigned integer values [X.680]. (These integer values are called "primary integer values" in X.660 because they can be accompanied by (not necessarily unambiguous) secondary identifiers. We ignore the latter and simply use the term "integer values" here, occasionally calling out their unsignedness.)

While these sequences can easily be represented in CBOR arrays of unsigned integers, a more compact representation can often be achieved by adopting the widely used representation of object identifiers defined in BER; this representation may also be more amenable to processing by other software that makes use of object identifiers.

BER represents the sequence of unsigned integers by concatenating self-delimiting [RFC6256] representations of each of the integer values in sequence.

ASN.1 distinguishes absolute object identifiers (ASN.1 Type "OBJECT IDENTIFIER"), which begin at a root arc ([X.660] Clause 3.5.21), from relative object identifiers (ASN.1 Type "RELATIVE-OID"), which begin relative to some object identifier known from context ([X.680] Clause 3.8.63). As a special optimization, BER combines the first two integers in an absolute object identifier into one numeric identifier by making use of the property of the hierarchy that the first arc has only three integer values (0, 1, and 2), and the second arcs under 0 and 1 are limited to the integer values between 0 and 39. (The root arc "joint-iso-itu-t(2)" has no such limitations on its second arc.) If X and Y are the first two integer values, the single integer value actually encoded is computed as:

```
X * 40 + Y
```

The inverse transformation (again making use of the known ranges of X and Y) is applied when decoding the object identifier.

Since the semantics of absolute and relative object identifiers differ, this specification defines two tags, collectively called the "OID tags" here:

Tag TBD111: tags a byte string as the [X.690] encoding of an absolute object identifier (simply "object identifier" or "OID").

Tag TBD110: tags a byte string as the [X.690] encoding of a relative object identifier (also "relative OID"). Since the encoding of each number is the same as for [RFC6256] Self-Delimiting Numeric Values (SDNVs), this tag can also be used for tagging a byte string that contains a sequence of zero or more SDNVs.

Tag TBD112: structurally like TBD110, but understood to be relative to "1.3.6.1.4.1" (IANA Private Enterprise Number OID). Hence, the semantics of the result are that of an absolute object identifier.

2.1. Requirements on the byte string being tagged

To form a valid tag, a byte string tagged by TBD111, TBD110, or TBD112 MUST be a syntactically valid BER representation of an object identifier: A concatenation of zero or more SDNV values, where each SDNV value is a sequence of one or more bytes that all have their most significant bit set, except for the last byte, where it must be unset; the first byte of each SDNV cannot be 0x80 (which would be a leading zero in SDNV's base-128 arithmetic).

In other words:

* its first byte, and any byte that follows a byte that has the most significant bit unset, MUST NOT be 0x80 (this requirement requires expressing the integer values in their shortest form, with no leading zeroes)

* its last byte MUST NOT have the most significant bit set (this requirement excludes an incomplete final integer value)

If either of these invalid conditions are encountered, the tag is invalid.

[X.680] restricts RELATIVE-OID values to have at least one arc, i.e., their encoding would have at least one SDNV. This specification permits empty relative object identifiers; they may still be excluded by application semantics.

To facilitate the search for specific object ID values, it is RECOMMENDED that definite length encoding (see Section 3.2.3 of [I-D.ietf-cbor-7049bis]) is used for the byte strings used as tag content for these tags.

The valid set of byte strings can also be expressed using regular expressions on bytes, using no specific notation but resembling [PCRE]. Unlike typical regular expressions that operate on character sequences, the following regular expressions take bytes as their domain, so they can be applied directly to CBOR byte strings.

For byte strings with tag TBD111:

```
"/^(([\x81-\xFF][\x80-\xFF]*)?[\x00-\x7F])+$/"
```

For byte strings with tag TBD110 or TBD112:

```
"/^(([\x81-\xFF][\x80-\xFF]*)?[\x00-\x7F])*$/"
```

A tag with tagged content that does not conform to the applicable regexp is invalid.

3. Examples

3.1. Encoding of the SHA-256 OID

ASN.1 Value Notation: { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm(4) hashalgs(2) sha256(1) }

Dotted Decimal Notation: 2.16.840.1.101.3.4.2.1

```
06                                  # UNIVERSAL TAG 6
   09                               # 9 bytes, primitive
      60 86 48 01 65 03 04 02 01    # X.690 Clause 8.19
#      |   840  1  |  3  4  2  1     show component encoding
#   2.16         101
```

Figure 1: SHA-256 OID in BER

```
D8 6F                               # tag(111)
   49                               # 0b010_01001: mt 2, 9 bytes
      60 86 48 01 65 03 04 02 01    # X.690 Clause 8.19
```

Figure 2: SHA-256 OID in CBOR

3.2. Encoding of a MIB Relative OID

Given some OID (e.g., "lowpanMib", assumed to be "1.3.6.1.2.1.226" [RFC7388]), to which the following is added:

ASN.1 Value Notation: { lowpanObjects(1) lowpanStats(1) lowpanOutTransmits(29) }

Dotted Decimal Notation: .1.1.29

```
0D                                # UNIVERSAL TAG 13
   03                             # 3 bytes, primitive
      01 01 1D                    # X.690 Clause 8.20
#      1  1 29                      show component encoding
```

Figure 3: MIB relative object identifier, in BER

```
D8 6E                             # tag(110)
   43                             # 0b010_01001: mt 2 (bstr), 3 bytes
      01 01 1D                    # X.690 Clause 8.20
```

Figure 4: MIB relative object identifier, in CBOR

This relative OID saves seven bytes compared to the full OID encoding.

4. Discussion

Staying close to the way object identifiers are encoded in ASN.1 BER makes back-and-forth translation easy; otherwise we would choose a more efficient encoding. Object identifiers in IETF protocols are serialized in dotted decimal form or BER form, so there is an advantage in not inventing a third form. Also, expectations of the cost of encoding object identifiers are based on BER; using a different encoding might not be aligned with these expectations. If additional information about an OID is desired, lookup services such as the OID Resolution Service (ORS) [X.672] and the OID Repository [OID-INFO] are available.

5. Tag Factoring with OID Arrays and Maps

OID tags can tag byte strings (as discussed above), but also CBOR arrays and maps. The idea in the latter case is that the tag is factored out from each individual item in the container; the tag is placed on the array or map instead.

When an OID tag is applied to an array, it means that the respective tag is imputed to all elements of the array that are byte strings, arrays, or maps. (There is no effect on other elements, including text strings or tags.) For example, when an array is tagged with TBD111, every array element that is a byte string is an OID, and every element that is an array or map is in turn treated as discussed here.

When an OID tag is applied to a map, it means that the respective tag is imputed to all keys in the map that are byte strings, arrays, or maps; again, there is no effect on keys of other major types. Note that there is also no effect on the values in the map.

As a result of these rules, tag factoring in nested arrays and maps is supported. For example, a 3-dimensional array of OIDs can be composed by using a single TBD111 tag containing an array of arrays of arrays of byte strings. All such byte strings are then considered OIDs.

6. Applications and Examples of OIDs

6.1. X.500 Distinguished Name

Consider the X.500 distinguished name:

Attribute Types	Attribute Values
c (2.5.4.6)	US
l (2.5.4.7) s (2.5.4.8) postalCode (2.5.4.17)	Los Angeles CA 90013
street (2.5.4.9)	532 S Olive St
businessCategory (2.5.4.15) buildingName (0.9.2342.19200300.100.1.48)	Public Park Pershing Square

Table 1: Example X.500 Distinguished Name

Table 1 has four "relative distinguished names" (RDNs). The country and street RDNs are single-valued. The second and fourth RDNs are multi-valued.

The equivalent representations in CBOR diagnostic notation and CBOR are:

```
111([{ h'550406': "US" },
     { h'550407': "Los Angeles", h'550408': "CA",
       h'550411': "90013" },
     { h'550409': "532 S Olive St" },
     { h'55040f': "Public Park",
       h'0992268993f22c640130': "Pershing Square" }])
```

Figure 5: Distinguished Name, in CBOR Diagnostic Notation

```
d8 6f                                       # tag(111)
   84                                       # array(4)
      a1                                    # map(1)
         43 550406                          # 2.5.4.6 (4)
         62                                 # text(2)
            5553                            # "US"
      a3                                    # map(3)
         43 550407                          # 2.5.4.7 (4)
         6b                                 # text(11)
            4c6f7320416e67656c6573          # "Los Angeles"
         43 550408                          # 2.5.4.8 (4)
         62                                 # text(2)
            4341                            # "CA"
         43 550411                          # 2.5.4.17 (4)
         65                                 # text(5)
            3930303133                      # "90013"
      a1                                    # map(1)
         43 550409                          # 2.5.4.9 (4)
         6e                                 # text(14)
            3533322053204f6c697665205374    # "532 S Olive St"
      a2                                    # map(2)
         43 55040f                          # 2.5.4.15 (4)
         6b                                 # text(11)
            5075626c6963205061726b          # "Public Park"
         4a 0992268993f22c640130    # 0.9.2342.19200300.100.1.48 (11)
         6f                                 # text(15)
            5065727368696e6720537175617265 # "Pershing Square"
```

Figure 6: Distinguished Name, in CBOR (109 bytes)

(This example encoding assumes that all attribute values are UTF-8 strings, or can be represented as UTF-8 strings with no loss of information.)

7. CDDL Control Operators

CDDL specifications may want to specify the use of SDNVs or SDNV sequences (as defined for the tag content for TBD110). This document introduces two new control operators that can be applied to a target value that is a byte string:

* ".sdnv", with a control type that contains unsigned integers. The byte string is specified to be encoded as an [RFC6256] SDNV (BER encoding) for the matching values of the control type.

* ".sdnvseq", with a control type that contains arrays of unsigned integers. The byte string is specified to be encoded as a sequence of [RFC6256] SDNVs (BER encoding) that decodes to an array of unsigned integers matching the control type.

* ".oid", like ".sdnvseq", except that the X*40+Y translation for absolute OIDs is included (see Figure 8).

Figure 7 shows an example for the use of ".sdnvseq" for a part of a structure using OIDs that could be used in Figure 6; Figure 8 shows the same with the ".oid" operator.

```
country-rdn = {country-oid => country-value}
country-oid = bytes .sdnvseq [85, 4, 6]
country-value = text .size 2
```

Figure 7: Using .sdnvseq

```
country-rdn = {country-oid => country-value}
country-oid = bytes .oid [2, 5, 4, 6]
country-value = text .size 2
```

Figure 8: Using .oid

Note that the control type need not be a literal; e.g., "bytes .oid [2, 5, 4, *uint]" matches all OIDs inside OID arc 2.5.4, "attributeType".

8. CDDL typenames

For the use with CDDL [RFC8610], the typenames defined in Figure 9 are recommended:

```
oid = #6.111(bstr)
roid = #6.110(bstr)
pen = #6.112(bstr)
```

Figure 9: Recommended typenames for CDDL

9. IANA Considerations

9.1. CBOR Tags

IANA is requested to assign the CBOR tags in Table 2, with the present document as the specification reference.

Tag	Data Item	Semantics
TBD111	byte string or array or map	object identifier (BER encoding)
TBD110	byte string or array or map	relative object identifier (BER encoding); SDNV [RFC6256] sequence
TBD112	byte string or array or map	object identifier (BER encoding), relative to 1.3.6.1.4.1

Table 2: Values for New Tags

9.2. CDDL Control Operators

IANA is requested to assign the CDDL Control Operators in Table 3, with the present document as the specification reference.

Name	Reference
.sdnv	[this document, Section 7]
.sdnvseq	[this document, Section 7]
.oid	[this document, Section 7]

Table 3: New CDDL Operators

10. Security Considerations

The security considerations of [I-D.ietf-cbor-7049bis] apply.

The encodings in Clauses 8.19 and 8.20 of [X.690] are quite compact and unambiguous, but MUST be followed precisely to avoid security pitfalls. In particular, the requirements set out in Section 2.1 of this document need to be followed; otherwise, an attacker may be able to subvert a checking process by submitting alternative representations that are later taken as the original (or even something else entirely) by another decoder supposed to be protected by the checking process.

OIDs and relative OIDs can always be treated as opaque byte strings. Actually understanding the structure that was used for generating them is not necessary, and, except for checking the structure requirements, it is strongly NOT RECOMMENDED to perform any processing of this kind (e.g., converting into dotted notation and back) unless absolutely necessary. If the OIDs are translated into other representations, the usual security considerations for non-trivial representation conversions apply; the integer values are unlimited in range.

11. References

11.1. Normative References

[I-D.ietf-cbor-7049bis]
Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", Work in Progress, Internet-Draft, draft-ietf-cbor-7049bis-16, 30 September 2020, <http://www.ietf.org/internet-drafts/draft-ietf-cbor-7049bis-16.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6256] Eddy, W. and E. Davies, "Using Self-Delimiting Numeric Values in Protocols", RFC 6256, DOI 10.17487/RFC6256, May 2011, <https://www.rfc-editor.org/info/rfc6256>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[X.660] International Telecommunications Union, "Information technology Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree", ITU-T Recommendation X.660, July 2011.

[X.680] International Telecommunications Union, "Information technology Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, August 2015.

[X.690] International Telecommunications Union, "Information technology ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, August 2015.

11.2. Informative References

[OID-INFO] Orange SA, "OID Repository", 2016, <http://www.oid-info.com/>.

[PCRE] Ho, A., "PCRE - Perl Compatible Regular Expressions", 2018, <http://www.pcre.org/>.

[RFC7388] Schoenwaelder, J., Sehgal, A., Tsou, T., and C. Zhou, "Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 7388, DOI 10.17487/RFC7388, October 2014, <https://www.rfc-editor.org/info/rfc7388>.

[X.672] International Telecommunications Union, "Information technology Open systems interconnection Object identifier resolution system", ITU-T Recommendation X.672, August 2010.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. Changes from -01 to -02

Minor editorial changes, remove some remnants, ready for WGLC.

A.2. Changes from -00 to -01

Clean up OID tag factoring.

A.3. Changes from -07 (bormann) to -00 (ietf)

Resubmitted as WG draft after adoption.

A.4. Changes from -06 to -07

Reduce the draft back to its basic mandate: Describe CBOR tags for what is colloquially know as ASN.1 Object IDs.

A.5. Changes from -05 to -06

Refreshed the draft to the current date ("keep-alive").

A.6. Changes from -04 to -05

Discussed UUID usage in CBOR, and incorporated fixes proposed by Olivier Dubuisson, including fixes regarding OID nomenclature.

A.7. Changes from -03 to -04

Changes occurred based on limited feedback, mainly centered around the abstract and introduction, rather than substantive technical changes. These changes include:

* Changed the title so that it is about tags and techniques.

* Rewrote the abstract to describe the content more accurately, and to point out that no changes to the wire protocol are being proposed.

* Removed "ASN.1" from "object identifiers", as OIDs are independent of ASN.1.

* Rewrote the introduction to be more about the present text.

* Proposed a concise OID arc.

* Provided binary regular expression forms for OID validation.

* Updated IANA registration tables.

A.8. Changes from -02 to -03

Many significant changes occurred in this version. These changes include:

* Expanded the draft scope to be a comprehensive CBOR update.

* Added OID-related sections: OID Enumerations, OID Maps and Arrays, and Applications and Examples of OIDs.

* Added Tag 36 update (binary MIME, better definitions).

* Added stub/experimental sections for X.690 Series Tags (tag <<X>>) and Regular Expressions (tag 35).

* Added technique for representing sets and multisets.

* Added references and fixed typos.

Acknowledgments

Jim Schaad provided a review of this document.

Authors' Addresses

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Sean Leonard
Penango, Inc.
5900 Wilshire Boulevard
21st Floor
Los Angeles, CA, 90036
United States of America

Email: dev+ietf@seantek.com
URI: http://www.penango.com/