Workgroup: Network Working Group
Internet-Draft:
draft-bormann-cbor-notable-tags-06
Published: 23 February 2022
Intended Status: Informational
Expires: 27 August 2022
Authors: C. Bormann
Universität Bremen TZI

Notable CBOR Tags

Abstract

The Concise Binary Object Representation (CBOR, RFC 8949) 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.

In CBOR, one point of extensibility is the definition of CBOR tags. RFC 8949's original edition, RFC 7049, defined a basic set of tags as well as a registry that can be used to contribute additional tag definitions [IANA.cbor-tags]. Since RFC 7049 was published, some 80 tag definitions have been added to that registry.

The present document provides a roadmap to a large subset of these tag definitions. Where applicable, it points to a IETF standards or standard development document that specifies the tag. Where no such document exists, the intention is to collect specification information from the sources of the registrations. After some more development, the present document is intended to be useful as a reference document for the IANA registrations of the CBOR tags the definitions of which have been collected.

Note to Readers

This is an individual submission to the CBOR working group of the IETF, https://datatracker.ietf.org/wg/cbor/about/. Discussion currently takes places on the github repository https://github.com/cabo/notable-tags. 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.

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

Copyright Notice

Table of Contents

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 selection of the tags presented here is somewhat arbitrary; considerations such as how wide the scope and area of application of a tag definition is combine with an assessment how "ready to use" the tag definition is (i.e., is the tag specification in a state where it can be used).

This document can only be a snapshot of a subset of the current registrations. The most up to date set of registrations is always available in the registry "CBOR Tags" [IANA.cbor-tags].

1.1. Terminology

The definitions of [STD94] apply. Specifically: The term "byte" is used in its now customary sense as a synonym for "octet"; "byte strings" are CBOR data items carrying a sequence of zero or more (binary) bytes, while "text strings" are CBOR data items carrying a sequence of zero or more Unicode code points, encoded in UTF-8 [STD63]. Where bit arithmetic is explained, this document uses the notation familiar from the programming language C ([C], including C++14's 0bnnn binary literals [Cplusplus20]), except that superscript notation (example for two to the power of 64: 2^{64}) denotes

exponentiation; in the plain text version of this document, superscript notation is rendered in paragraph text by C-incompatible surrogate notation as seen in this example. Ranges expressed using .. are inclusive of the limits given. Type names such as "int", "bigint" or "decfrac" are taken from Appendix D of [RFC8610], the Concise Data Definition Language (CDDL).

2. RFC 7049 (original CBOR specification)

[RFC7049] defines a number of tags that are listed here for convenience only.

Tag number	Tag content	Short Description	Section of RFC 7049
0	UTF-8 string	Standard date/time string	2.4.1
1	multiple	Epoch-based date/time	2.4.1
2	byte string	Positive bignum	2.4.2
3	byte string	Negative bignum	2.4.2
4	array	Decimal fraction	2.4.3
5	array	Bigfloat	2.4.3
21	multiple	Expected conversion to base64url encoding	2.4.4.2
22	multiple	Expected conversion to base64 encoding	2.4.4.2
23	multiple	Expected conversion to base16 encoding	2.4.4.2
24	byte string	Encoded CBOR data item	2.4.4.1
32	UTF-8 string	URI	2.4.4.3
33	UTF-8 string	base64url	2.4.4.3
34	UTF-8 string	base64	2.4.4.3
35	UTF-8 string	Regular expression	2.4.4.3
36	UTF-8 string	MIME message	2.4.4.3
55799	multiple	Self-describe CBOR	2.4.5

Table 1: Tag numbers defined in RFC 7049

2.1. Tags Related to Those Defined in RFC 7049

Separately registered tags that are directly related to the tags predefined in RFC 7049 include:

* Tag 63, registered by this document, is a parallel to tag 24, with the single difference that its byte string tag content carries a CBOR Sequence [RFC8742] instead of a single CBOR data item.

* Tag 257, registered by Peter Occil with a specification in http://peteroupc.github.io/CBOR/binarymime.html, is a parallel to tag 36, except that the tag content is a byte string, which therefore can also carry binary MIME messages as per [RFC2045].

2.2. Tags from RFC 7049 not listed in RFC 8949

Appendix G.3 of [STD94] states:

> Tag 35 is not defined by this document; the registration based on the definition in RFC 7049 remains in place.

The reason for this exclusion is that the definition of Tag 35 in Section 2.4.4.3 of [RFC7049], leaves too much open to ensure interoperability:

> Tag 35 is for regular expressions in Perl Compatible Regular Expressions (PCRE) / JavaScript syntax [ECMA262].

Not only are two partially incompatible specifications given for the semantics, JavaScript regular expressions have also developed significantly within the decade since JavaScript 5.1 (which was referenced as "ECMA262" by [RFC7049]), making it less reliable to assume that a producing application will manage to stay within that 2011 subset.

Nonetheless, the registration is in place, so it is available for applications that simply want to mark a text string as being a regular expression roughly of the PCRE/Javascript flavor families.

3. Security

A number of CBOR tags are defined in security specifications that make use of CBOR.

3.1. RFC 8152 (COSE)

[RFC8152] defines CBOR Object Signing and Encryption (COSE). A revision is in process that splits this specification into the data structure definitions [I-D.ietf-cose-rfc8152bis-struct], which will

define another tag for COSE standalone counter signature, and the algorithms employed [I-D.ietf-cose-rfc8152bis-algs].

Tag number	Tag content	Short Description
16	COSE_Encrypt0	COSE Single Recipient Encrypted Data Object
17	COSE_Mac0	COSE Mac w/o Recipients Object
18	COSE_Sign1	COSE Single Signer Data Object
96	COSE_Encrypt	COSE Encrypted Data Object
97	COSE_Mac	COSE MACed Data Object
98	COSE_Sign	COSE Signed Data Object

Table 2: Tag numbers defined in RFC 8152, COSE

3.2. RFC 8392 (CWT)

[RFC8392] defines the CBOR Web Token (CWT), making use of COSE to define a CBOR variant of the JOSE Web Token (JWT), [RFC7519], a standardized security token that has found use in the area of web applications, but is not technically limited to those.

Tag number	Tag content	Short Description
61	CBOR Web Token (CWT)	CBOR Web Token (CWT)

Table 3: Tag number defined for RFC 8392 CBOR Web Token (CWT)

4. CBOR-based Representation Formats

Representation formats can be built on top of CBOR.

4.1. YANG-CBOR

YANG [RFC7950] is a data modeling language originally designed in the context of the Network Configuration Protocol (NETCONF) [RFC6241], now widely used for modeling management and configuration information. [RFC7950] defines an XML-based representation format, and [RFC7951] defines a JSON-based [RFC8259] representation format for YANG.

YANG-CBOR [I-D.ietf-core-yang-cbor] is a representation format for YANG data in CBOR.

Tag number	Tag content	Short Description	Section of YANG-CBOR
43	byte string	YANG bits datatype	6.7
44	unsigned integer		6.6

Tag number	Tag content	Short Description	Section of YANG-CBOR
		YANG enumeration datatype	
45	unsigned integer or text string	YANG identityref datatype	6.10
46	unsigned integer or text string or array	YANG instance-identifier datatype	6.13
47	unsigned integer	YANG Schema Item iDentifier (sid)	3.2

Table 4: Tag number defined for YANG-CBOR

5. Protocols

Protocols may want to allocate CBOR tag numbers to identify specific protocol elements.

5.1. DOTS

DDoS Open Threat Signaling (DOTS) defines tag number 271 for the DOTS signal channel object in [RFC9132].

5.2. RAINS

As an example for how experimental protocols can make use of CBOR tag definitions, the RAINS (Another Internet Naming Service) Protocol Specification defines tag number 15309736 for a RAINS Message [I-D.trammell-rains-protocol]. (The seemingly random tag number was chosen so that, when represented as an encoded CBOR tag argument, it contains the Unicode character " " (U+96E8) in UTF-8, which represents rain in a number of languages.)

6. Datatypes

6.1. Advanced arithmetic

A number of tags have been registered for arithmetic representations beyond those built into CBOR and defined by tags in [RFC7049]. These are all documented under http://peteroupc.github.io/CBOR/; the last pathname component for the URL is given in Table 5.

Tag number	Tag content	Short Description	Reference
30	array	Rational number	rational.html
264	array	Decimal fraction with arbitrary exponent	bigfrac.html
265	array	Bigfloat with arbitrary exponent	bigfrac.html

Tag number	Tag content	Short Description	Reference
268	array	Extended decimal fraction	extended.html
269	array	Extended bigfloat	extended.html
270	array	Extended rational number	extended.html

Table 5: Tags for advanced arithmetic

CBOR's basic generic data model (Section 2 of [STD94]) has a number system with limited-range integers (major types 0 and 1: $-2^{64}..2^{64}-1$) and floating point numbers that cover binary16, binary32, and binary64 (including non-finites) from [IEEE754]. With the tags defined with [RFC7049], the extended generic data model (Section 2.1 of [STD94]) adds unlimited-range integers (tag numbers 2 and 3, "bigint" in CDDL) as well as floating point values using the bases 2 (tag number 5, "bigfloat") and 10 (tag number 4, "decfrac").

This pre-defined number system has a number of limitations that are addressed in three of the tags discussed here:

* Tag number 30 allows the representation of rational numbers as a ratio of two integers: a numerator (usually written as the top part of a fraction), and a denominator (the bottom part), where both integers can be limited-range basic and unlimited-range integers. The mathematical value of a rational number is the numerator divided by the denominator. This tag can express all numbers that the extended generic data model of [RFC7049] can express, except for non-finites [IEEE754]; it also can express rational numbers that cannot be expressed with denominators that are a power of 2 or a power of 10.

  For example, the rational number 1/3 is encoded:

```
  d8 1e      ---- Tag 30
     82      ---- Array length 2
        01   ---- 1
        03   ---- 3
```

  Many programming languages have built-in support for rational numbers or support for them is included in their standard libraries; tag number 30 is a way for these platforms to interchange these rational numbers in CBOR.

* Tag numbers 4 and 5 are limited in the range of the (base 10 or base 2) exponents by the limited-range integers in the basic generic data model. Tag numbers 264 and 265 are exactly equivalent to 4 and 5, respectively, but also allow unlimited-range integers as exponents. While applications for floating

point numbers with exponents outside the CBOR basic integer range are limited, tags 264 and 265 allow unlimited roundtripping with other formats that allow very large or very small exponents, such as those JSON [RFC8259] can provide if the limitations of I-JSON [RFC7493] do not apply.

The tag numbers 268..270 extend these tags further by providing a way to express non-finites within a tag with this number. This does not increase the expressiveness of the data model (the non-finites can already be expressed using major type 7 floating point numbers), but does allow both finite and non-finite values to carry the same tag. In most applications, a choice that includes some of the three tags 30, 264, 265 for finite values and major type 7 floating point values for non-finites (as well as possibly other parts of the CBOR number system) will be the preferred solution.

This document suggests using the CDDL typenames defined in Figure 1 for the three most useful tag numbers in this section.

```
rational = #6.30([numerator: integer, denominator: integer .ne 0])
rational_of<N,D> = #6.30([numerator: N, denominator: D])
; the value 1/3 can be notated as rational_of<1, 3>

extended_decfrac = #6.264([e10: integer, m: integer])
extended_bigfloat = #6.265([e2: integer, m: integer])
```

Figure 1: CDDL for extended arithmetic tags

6.2. Variants of undefined

https://github.com/svaarala/cbor-specs/blob/master/cbor-absent-tag.rst defines tag 31 to be applied to the CBOR value Undefined (0xf7), slightly modifying its semantics to stand for an absent value in a CBOR Array.

(TO DO: Obtain permission to copy the definitions here.)

6.3. Typed and Homogeneous Arrays

[RFC8746] defines tags for various kinds of arrays. A summary is reproduced in Table 6.

Tag	Data Item	Semantics
64	byte string	uint8 Typed Array
65	byte string	uint16, big endian, Typed Array
66	byte string	uint32, big endian, Typed Array
67	byte string	uint64, big endian, Typed Array
68	byte string	uint8 Typed Array, clamped arithmetic

Tag	Data Item	Semantics
69	byte string	uint16, little endian, Typed Array
70	byte string	uint32, little endian, Typed Array
71	byte string	uint64, little endian, Typed Array
72	byte string	sint8 Typed Array
73	byte string	sint16, big endian, Typed Array
74	byte string	sint32, big endian, Typed Array
75	byte string	sint64, big endian, Typed Array
76	byte string	(reserved)
77	byte string	sint16, little endian, Typed Array
78	byte string	sint32, little endian, Typed Array
79	byte string	sint64, little endian, Typed Array
80	byte string	IEEE 754 binary16, big endian, Typed Array
81	byte string	IEEE 754 binary32, big endian, Typed Array
82	byte string	IEEE 754 binary64, big endian, Typed Array
83	byte string	IEEE 754 binary128, big endian, Typed Array
84	byte string	IEEE 754 binary16, little endian, Typed Array
85	byte string	IEEE 754 binary32, little endian, Typed Array
86	byte string	IEEE 754 binary64, little endian, Typed Array
87	byte string	IEEE 754 binary128, little endian, Typed Array
40	array of two arrays*	Multi-dimensional Array, row-major order
1040	array of two arrays*	Multi-dimensional Array, column-major order
41	array	Homogeneous Array

Table 6: Tag numbers defined for Arrays

7. Domain-Specific

(TO DO: Obtain permission to copy the definitions here; explain how tags 52 and 54 essentially obsolete 260/261.)

Tag number	Tag content	Short Description	Reference	Author
37	byte string	Binary UUID (Section 4.1.2 of [RFC4122])	https://github.com/lucas-clemente/cbor-specs/blob/master/uuid.md	Lucas Clemente
38	array	Language-tagged string	http://peteroupc.github.io/CBOR/langtags.html	Peter Occil
257	byte string	Binary MIME message		Peter Occil

Tag number	Tag content	Short Description	Reference	Author
			http://peteroupc.github.io/CBOR/binarymime.html	
260	byte string	Network Address (IPv4 or IPv6 or MAC Address)	http://www.employees.org/~ravir/cbor-network.txt	Ravi Raju
261	map	Network Address Prefix (IPv4 or IPv6 Address + Mask Length)	https://github.com/toravir/CBOR-Tag-Specs/blob/master/networkPrefix.md	Ravi Raju
263	byte string	Hexadecimal string	https://github.com/toravir/CBOR-Tag-Specs/blob/master/hexString.md	Ravi Raju
266	text string	Internationalized resource identifier (IRI)	https://peteroupc.github.io/CBOR/iri.html	Peter Occil
267	text string	Internationalized resource identifier reference (IRI reference)	https://peteroupc.github.io/CBOR/iri.html	Peter Occil

Table 7

7.1. Extended Time Formats

Additional tag definitions have been provided for date and time values.

Tag	Data Item	Semantics	Reference
100	integer	date in number of days since epoch	[RFC8943]
1004	text string	RFC 3339 full-date string	[RFC8943]
1001	map	extended time	[I-D.ietf-cbor-time-tag]
1002	map	duration	[I-D.ietf-cbor-time-tag]
1003	map	period	[I-D.ietf-cbor-time-tag]

Table 8: Tag numbers for date and time

Note that tags 100 and 1004 are for calendar dates that are not anchored to a specific time zone; they are meant to specify calendar

dates as perceived by humans, e.g. for use in personal identification documents. Converting such a calendar date into a specific point in time needs the addition of a time-of-day (for which a CBOR tag is outstanding) and timezone information (also outstanding). Alternatively, a calendar date plus timezone information can be converted into a time period (range of time values given by the starting and the ending time); note that these time periods are not always exactly 24 h (86400 s) long.

[RFC8943] does not suggest CDDL [RFC8610] type names for the two tags. We suggest copying the definitions in Figure 2 into application-specific CDDL as needed.

```
caldate = #6.100(int) ; calendar date as a number of days from 1970-01-01
tcaldate = #6.1004(tstr) ; calendar date as an RFC 3339 full-date string
```

Figure 2: CDDL for calendar date tags (RFC8943)

Tag 1001 extends tag 1 by additional information (such as picosecond resolution) and allows the use of Decimal and Bigfloat numbers for the time.

8. Platform-oriented

8.1. Perl

(These are actually not as Perl-specific as the title of this section suggests. See also the penultimate paragraph of Section 3.4 of [STD94].)

These are all documented under http://cbor.schmorp.de/; the last pathname component is given in Table 9.

(TO DO: Obtain permission to copy the definitions here.)

Tag	Data Item	Semantics	Reference
256	multiple	mark value as having string references	stringref
25	unsigned integer	reference the nth previously seen string	stringref
26	array	Serialized Perl object with classname and constructor arguments	perl-object
27	array	Serialized language-independent object with type name and constructor arguments	generic-object
28	multiple	mark value as (potentially) shared	value-sharing

Tag	Data Item	Semantics	Reference
29	unsigned integer	reference nth marked value	value-sharing
22098	multiple	hint that indicates an additional level of indirection	indirection

Table 9: Tag numbers that aid the Perl platform

8.2. JSON

(TO DO: Obtain permission to copy the definitions here.)

Tag number 262 has been registered to identify byte strings that carry embedded JSON text (https://github.com/toravir/CBOR-Tag-Specs/blob/master/embeddedJSON.md).

Tag number 275 can be used to identify maps that contain keys that are all of type Text String, as they would occur in JSON (https://github.com/ecorm/cbor-tag-text-key-map).

8.3. Weird text encodings

(TO DO: Obtain permission to copy the definitions here.)

Some variants of UTF-8 are in use in specific areas of application. Tags have been registered to be able to carry around strings in these variants in case they are not also valid UTF-8 and can therefore not be represented as a CBOR text string (https://github.com/svaarala/cbor-specs/blob/master/cbor-nonutf8-string-tags.rst).

Tag Number	Data Item	Semantics
272	byte string	Non-UTF-8 CESU-8 string
273	byte string	Non-UTF-8 WTF-8 string
274	byte string	Non-UTF-8 MUTF-8 string

Table 10: Tag numbers for UTF-8 variants

9. Application-specific

(TO DO: Obtain permission to copy the definitions here.)

Tag number	Tag content	Short Description	Reference	Author
39	multiple	Identifier	[https://github.com/lucas-clemente/cbor-specs/blob/master/id.md	Lucas Clemente
42				

Tag number	Tag content	Short Description	Reference	Author
	byte string	IPLD content identifier	[https://github.com/ipld/cid-cbor/	Volker Mische
103	array	Geographic Coordinates	[https://github.com/allthingstalk/cbor/blob/master/CBOR-Tag103-Geographic-Coordinates.md	Danilo Vidovic
104	multiple	Geographic Coordinate Reference System WKT or EPSG number	[I-D.clarke-cbor-crs]	
120	multiple	Internet of Things Data Point	[https://github.com/allthingstalk/cbor/blob/master/CBOR-Tag120-Internet-of-Things-Data-Points.md	Danilo Vidovic
258	array	Mathematical finite set	[https://github.com/input-output-hk/cbor-sets-spec/blob/master/CBOR_SETS.md	Alfredo Di Napoli
259	map	Map datatype with key-value operations (e.g. .get()/.set()/.delete())	[https://github.com/shanewholloway/js-cbor-codec/blob/master/docs/CBOR-259-spec--explicit-maps.md	Shane Holloway

Table 11

9.1. Enumerated Alternative Data Items

(Original Text for this section was contributed by Duncan Coutts and Michael Peyton Jones; all errors are the author's.)

A set of CBOR tag numbers has been allocated (to do, Section 11) for encoding data composed of enumerated alternatives:

Tags	Data Item	Meaning
121..127	any	alternatives 0..6, 1+1 encoding

Tags	Data Item	Meaning
1280..1400	any	alternatives 7..127, 1+2 encoding
101	array [uint, any]	alternatives as given by the uint + 128

Table 12: Tags for Enumerated Alternative Data Items

The tags defined in this section are for encoding data that can be in one of a number of different enumerated forms.

For example data representing the result of some action might be either a failure with some failure detail, or a success with some result. In this example there are two cases, the failure case and the success case, and we can enumerate them as 0 and 1.

In general the number of alternatives, and what data is expected in each alternative case is entirely application dependent.

The tags defined in this specification allow the encoding of any number of alternatives, but provide compact encoding for the common cases of low numbers of alternatives:

- Alternatives 0..6 can be encoded in 2 bytes;
- Alternatives 7..127 can be encoded in 3 bytes;
- Alternatives 128+ can be encoded in 3-12 bytes.

There are no special considerations for deterministic encoding Section 4.2 of [STD94]: The case numbers covered by each tag do not overlap; particularly, tag 101 encoding starts where the more compact special encodings for 0..6 and 7..127 end.

9.1.1. Semantics

The value consists of a case number and a case body. The case number is an unsigned integer that indicates which case out of the set of alternatives is used. The case body is any CBOR data value.

In a setting where the application uses a schema (formally or informally), then there will be an appropriate sub-schema for each case in the set of alternatives. The representation of the case body should comply with the schema corresponding to the case number used.

To continue the example above about representing failure or success, suppose that the failure detail consists of an integer code and a string, and suppose that the successful result is a byte string. A failure value will use case 0 and the case body will be a CBOR list containing an integer and a text string. Alternatively, a success value will use case 1 and the body will be a single CBOR byte string.

Decoders that enforce a schema must check the case number is within the range of cases allowed, and that the case body follows the schema for the supplied case number. Generic decoders should allow any case number and any CBOR data value for the case body.

9.1.2. Rationale

CBOR has direct support for *combinations* of multiple values but not for *alternatives* of multiple values. Combinations are expressed in CBOR using lists or maps.

Most programming languages have a notion of data consisting of combinations of data values, often called records or objects. Many programming languages also have a notion of data consisting of multiple alternative data values. For example C has unions, and other languages have "tagged" unions (where it is always clear which alternative is in use).

Crucially for this set of tags, the set of alternatives must be closed and ordered. This allows encoding using an unsigned number to distinguish each case.

Note that this does *not* correspond to the notion in some programming languages of classes and subclasses since in that context the set of alternatives is open and unordered. Alternatives of this kind are well-supported by tag 27 "Serialized language-independent object with type name and constructor arguments".

In functional programming languages, the primary way of forming new data types is to enumerate a set of alternatives (each of which may be a record). Such forms of data are also supported in hybrid functional languages or languages with functional features.

Thus, in some applications, it is very common to have data making use of alternatives, and it is worth finding a compact encoding, at least for the common cases. Just as most records are small, most alternatives are also small.

In this specification we reserve 7 values in the 2-byte part of the available tag encoding space for alternatives 0..6 which are by far the most common. We reserve a range of 121 values in the 3-bytes tag encoding space. To cover the general case we use an encoding using a pair consisting of an unsigned integer and the case body, the first 24 of which also result in a 3-byte encoding.

9.1.3. Examples

To elaborate on the example from the introduction, we have a "result" that is a failure or success, where:

- the failure detail consists of an integer code and a string;
- the successful result is a byte string.

This corresponds to the following schema, in CDDL notation:

```
result = #6.121([int, text])
       / #6.122(bytes)
```

Example values:

```
121([3, "the printer is on fire"])
```

```
122(h'ff00')
```

As a second example, here is one based on a data type defined within the Haskell programming language, representing a simple expression tree.

```
-- A data type representing simple arithmetic expressions

data Expr = Lit Int -- integer literal
| Add Expr Expr -- addition
| Sub Expr Expr -- subtraction
| Neg Expr -- unary negation
| Mul Expr Expr -- multiplication
| Div Expr Expr -- integer division
```

In CDDL notation, and using the tags in this specification, such data could be encoded using this schema:

```
; A data type representing simple arithmetic expressions

expr = 121(int)          ; integer literal
     / 122([expr, expr]) ; addition
     / 123([expr, expr]) ; subtraction
     / 124(expr)         ; unary negation
     / 125([expr, expr]) ; multiplication
     / 126([expr, expr]) ; integer division
```

10. Implementation aids

10.1. Invalid Tag

The present document registers tag numbers 65535, 4294967295, and 18446744073709551615 (16-bit 0xffff, 32-bit 0xffffffff, and 64-bit 0xffffffffffffffff) as Invalid Tags, tags that are always invalid, independent of the tag content provided. The purpose of these tag number registrations is to enable the tag numbers to be reserved for internal use by implementations to note the absence of a tag on a data item where a tag could also be expected with that data item as tag content.

The Invalid Tags are not intended to ever occur in interchanged CBOR data items. Generic CBOR decoder implementations are encouraged to raise an error if an Invalid Tag occurs in a CBOR data item even if there is no validity checking implemented otherwise.

11. IANA Considerations

In the registry "CBOR Tags" [IANA.cbor-tags], IANA has allocated the first to third tag in Table 13 from the FCFS space, with the present document as the specification reference. IANA has allocated the fourth tag from the Specification Required space, with the present document as the specification reference.

Tag	Data Item	Semantics	Reference
65535	(none valid)	always invalid	draft-bormann-cbor-notable-tags, Section 10.1
4294967295	(none valid)	always invalid	draft-bormann-cbor-notable-tags, Section 10.1
18446744073709551615		always invalid	

Tag	Data Item	Semantics	Reference
	(none valid)		draft-bormann-cbor-notable-tags, Section 10.1
63	byte string	Encoded CBOR Sequence [RFC8742]	draft-bormann-cbor-notable-tags, Section 2.1

Table 13: Values for Tags

In addition, IANA is requested to allocate the tags from Table 12, with a reference to the present document.

12. Security Considerations

The security considerations of [STD94] apply; the tags discussed here may also have specific security considerations that are mentioned in their specific sections above.

13. References

13.1. Normative References

[I-D.ietf-core-yang-cbor] Veillette, M., Petrov, I., Pelov, A., Bormann, C., and M. Richardson, "CBOR Encoding of Data Modeled with YANG", Work in Progress, Internet-Draft, draft-ietf-core-yang-cbor-18, 19 December 2021, <https://www.ietf.org/archive/id/draft-ietf-core-yang-cbor-18.txt>.

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

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

[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>.

[RFC8746] Bormann, C., Ed., "Concise Binary Object Representation (CBOR) Tags for Typed Arrays", RFC 8746, DOI 10.17487/RFC8746, February 2020, <https://www.rfc-editor.org/info/rfc8746>.

[RFC9132] Boucadair, M., Ed., Shallow, J., and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 9132, DOI 10.17487/RFC9132, September 2021, <https://www.rfc-editor.org/info/rfc9132>.

[STD94] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

13.2. Informative References

[C] International Organization for Standardization, "Information technology - Programming languages - C", ISO/IEC 9899:2018, June 2018, <https://www.iso.org/standard/74528.html>.

[Cplusplus20] International Organization for Standardization, "Programming languages - C++", ISO/IEC ISO/IEC JTC1 SC22 WG21 N 4860, March 2020, <https://isocpp.org/files/papers/N4860.pdf>.

[I-D.clarke-cbor-crs] Clarke, T. R., "Concise Binary Object Representation (CBOR) Tag for Coordinate Reference System (CRS) Specification", Work in Progress, Internet-Draft, draft-clarke-cbor-crs-02, 17 March 2020, <https://www.ietf.org/archive/id/draft-clarke-cbor-crs-02.txt>.

[I-D.ietf-cbor-time-tag] Bormann, C., Gamari, B., and H. Birkholz, "Concise Binary Object Representation (CBOR) Tags for Time, Duration, and Period", Work in Progress, Internet-Draft, draft-ietf-cbor-time-tag-00, 19 May 2021, <https://www.ietf.org/archive/id/draft-ietf-cbor-time-tag-00.txt>.

[I-D.ietf-cose-rfc8152bis-algs]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-algs-12, 24 September 2020, <https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-algs-12.txt>.

[I-D.ietf-cose-rfc8152bis-struct]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-struct-15, 1 February 2021, <https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-struct-15.txt>.

[I-D.trammell-rains-protocol] Trammell, B. and C. Fehlmann, "RAINS (Another Internet Naming Service) Protocol Specification", Work in Progress, Internet-Draft, draft-trammell-rains-protocol-05, 29 January 2019, <https://www.ietf.org/archive/id/draft-trammell-rains-protocol-05.txt>.

[IEEE754] IEEE, "IEEE Standard for Floating-Point Arithmetic", IEEE Std 754-2019, DOI 10.1109/IEEESTD.2019.8766229, <https://ieeexplore.ieee.org/document/8766229>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol

(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[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>.

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

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <https://www.rfc-editor.org/info/rfc7493>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

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

[RFC8943] Jones, M., Nadalin, A., and J. Richter, "Concise Binary Object Representation (CBOR) Tags for Date", RFC 8943, DOI 10.17487/RFC8943, November 2020, <https://www.rfc-editor.org/info/rfc8943>.

[STD63] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

Acknowledgements

(Many, TBD)

Contributors

Peter Occil

Email: poccil14 at gmail dot com

Peter Occil registered tags 30, 264, 265, 268-270 (Section 6.1), 38, 257, 266 and 267 (Section 7), and contributed much of the text about these tags in this document.

Duncan Coutts

Email: duncan@well-typed.com

Michael Peyton Jones

Email: me@michaelpj.com

Jane Doe
To do

Further contributors will be listed here as text is added.

Plase stay tuned.

Author's Address

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

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