

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 29 August 2026

C. Bormann
Universität Bremen TZI
25 February 2026

Notable CBOR Tags
draft-bormann-cbor-notable-tags-16

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 16 tags as well as a registry that can be used to contribute additional tag definitions [IANA.cbor-tags]. Since RFC 7049 was published, at the time of writing some 250 definitions of tags and ranges of tags 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 an 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/> (<https://datatracker.ietf.org/wg/cbor/about/>). Discussion currently takes places on the github repository <https://github.com/cabo/notable-tags> (<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> (<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 all authors of registered tag definitions 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.

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 29 August 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. The CBOR Tags Registry	4
1.2. Terminology	5
2. RFC 7049 (original CBOR specification)	5
2.1. Tags Related to Those Defined in RFC 7049	6
2.2. Tags from RFC 7049 not listed in RFC 8949	7
3. Security	8
3.1. COSE	8
3.1.1. Tags for Bare Hash Values	9
3.2. RFC 8392 (CWT)	11
4. CBOR-based Representation Formats	11
4.1. YANG-CBOR	11
5. Protocols	12
5.1. DOTS	12
5.2. RAINS	12

6. Datatypes	13
6.1. Advanced arithmetic	13
6.2. Variants of undefined	15
6.3. Typed and Homogeneous Arrays	15
7. Domain-Specific	16
7.1. Human-readable Text	18
7.2. Extended Time Formats	19
8. Platform-oriented	20
8.1. Perl	20
8.2. JSON	21
8.3. Weird text encodings	22
9. Application-specific	22
9.1. Enumerated Alternative Data Items	23
9.1.1. Semantics	25
9.1.2. Rationale	25
9.1.3. Examples	26
10. Implementation aids	27
10.1. Invalid Tag	27
10.2. Programming Aid for Simple Values	27
10.3. Test Tag	28
10.3.1. Semantics	28
10.3.2. Sample code	28
11. Managing redundancy	28
11.1. Packed CBOR	29
11.2. Stringref	29
11.2.1. Stringref Mechanism	29
11.2.2. Stringref Namespaces	30
11.2.3. Serialization Considerations	31
12. IANA Considerations	31
13. Security Considerations	33
14. References	33
14.1. Normative References	33
14.2. Informative References	35
List of Figures	40
List of Tables	40
Acknowledgements	40
Contributors	40
Author's Address	41

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. The CBOR Tags Registry

CBOR tags are an extension point of CBOR, and that extension point is managed through an IANA registry ([IANA.cbor-tags] as set up in Section 9.2 of RFC 8949 [STD94]), not through publishing documents.

Some tag ranges require a stable specification to be publicly available under the "Specification Required" policy (Section 4.6 of RFC 8126 [BCP26]). That stable availability can be provided by any suitable means, such as a (specific revision of a) GitHub repository, a document published by an SDO or other entity, or even by a specific revision of an Internet-Draft, or an RFC if that happens to be in the pipeline. (In the latter case, while the RFC is being developed, the allocation will probably be made when that RFC still is in Internet-Draft state, which is then referenced in the registry and is usually updated to the RFC number when that is published.)

An example for this is provided by Section 9.1 of the present document. A few allocations were made for these tags (101, 121-127, 1280-1400) that are now a permanent part of the CBOR Tags registry. These allocations do reference revision 07 of the present Internet-Draft, because that contains an explanation of what these tags are supposed to be used for. That document revision 07 is part of the permanent record of the IETF and is not going away, so we are able to use it as a stable part of the information provided with the allocation. (In more recent revisions of the present document, some glue text was updated and a clerical error in illustrative CDDL code was fixed; eventually, there may be some incentive to update the registry entry to a newer revision of the present document.)

As with other stable specifications referenced this way, there is no need for the present document to be a published RFC before the tags become useful; i.e., the publication action that makes the tag numbers available for wide use in interchange is the acceptance of the tags into the IANA CBOR Tags registry.

One example of an Internet-Draft-based specification document that eventually did get published as an RFC is given by the CBOR Tags for Time, Duration, and Period [RFC9581]. These were originally registered around 2017 (under the original policy for these registrations) with a reference to [I-D.draft-bormann-cbor-time-tag-01]. Over time, the document evolved into a WG document without a need to update the reference. The registration was finally updated to point to [RFC9581] when that was published, which was also the time additional registries were created for some of the inner workings of the time tags.

1.2. 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 scalar values (code points that can be part of a string of Unicode characters), 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 (Section 12), 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 encoded CBOR data item.

- * Tag 108, registered by this document (Section 12), is a parallel to tag 23, with the single difference that the hexadecimal form uses lowercase instead of uppercase letters.

Note that tag 23 has a serialization that is one byte shorter than tag 108, so if all else is equal, tag 23 (and thus upper case hex) would be chosen as it is slightly more efficient than tag 108. However, designers of CBOR protocols that use one of these tags often have reason to prefer lowercase hex in the application they are trying to be compatible with, in which case 108 provides a solution that is only one byte more expensive.

- * Tag 257, registered by Peter Occil with a specification in <http://peteroupc.github.io/CBOR/binarymime.html> (<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].
- * Tag 21065, having been registered by this document (Section 12), is a parallel to tag 35, with the difference that its text string tag content carries an I-Regexp regular expression [RFC9485] instead of a regexp of a more unspecified flavor. Companion tag 21066, having been registered by Joe Hildebrand with a specification in <https://github.com/hildjj/cbor-specs/blob/main/regexp.md> (<https://github.com/hildjj/cbor-specs/blob/main/regexp.md>), is the equivalent for JavaScript (ECMA262), but besides the regular expression itself also can include the regular expression flags as a separate item.

2.2. Tags from RFC 7049 not listed in RFC 8949

Appendix G.3 of RFC 8949 [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. See also Tag 21065 and 21066 above.

3. Security

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

3.1. COSE

CBOR Object Signing and Encryption (COSE) is defined in a number of RFCs. [RFC8152] was the initial specification, set up the registries, and populated them with an initial set of assignments. A revision split this specification into the data structure definitions (RFC9052), an Internet Standard [STD96], and a separate document defining the representation for the algorithms employed [RFC9053], which is expected to be updated more frequently than the COSE format itself. [RFC9054] added a separate set of algorithms for cryptographic hash functions (Hash functions have been a component of some [RFC9053] combined algorithms but weren't originally assigned separate codepoints themselves). A revised COSE counter signature structure was defined in RFC9338, another part of [STD96]; this also defines a tag for these.

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
19	COSE_Countersignature	COSE standalone V2 countersignature (RFC9338)
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 RFC9052, COSE, and RFC 9338

3.1.1. Tags for Bare Hash Values

[RFC9054] does not define CBOR tags for cryptographic Hash values; it rightly notes that Hash values are often used in structures that are application-specific and should be defined with those applications.

However, there are many cases where just a bare hash value is required, and for these cases common tags are useful. In one use case, these tags occur in a data structure that is specified to indicate elision by using one of these tags as an alternative to some other data structure. To enable agility, tags need to indicate the hash function used, preferably using the COSE algorithms registry as populated by [RFC9054].

(Note that there is another registry, "Named Information Hash Algorithm Registry" [IANA.named-information], that also defines numbers for some hash algorithms. We are not using this registry here, as more recent entries seem to have stopped assigning numbers. If desired, tags that employ this registry could be added later.)

The codepoint range available for the COSE algorithms registry is large, but the most likely range to be used for standard Hash functions is "Integer values between -256 and 255", which have the registry policy "Standards Action With Expert Review" (Section 16.4 of [RFC8152], Registry "COSE Algorithms" [IANA.cose]).

To this end, the present document registers a range of 512 tags from 18300 to 18811 (inclusive), paralleling the algorithm identifier range of -256 .. 255 (inclusive). The tag number for COSE algorithm number N is then defined to be 18556+N, except for N = 0 (see below). The tag value is a CBOR byte string, with the exception N = 0.

For example, in [IANA.cose] SHA-256 has the COSE algorithm identifier -16. This is in the range -256 .. 255 (inclusive range). Therefore, tag 18540 (= 18556 + (-16)) is the tag for a byte string containing a SHA-256 hash.

As a special case, there is one exception: Tag 18556 (= 18556 + 0) stands for the combination of a an explicit numeric COSE algorithm identifier with a hash value in an array, analogous to the use of COSE_CertHash in [RFC9360]:

```
Standard_COSE_Hash<alg, value> =
    #6.<hashmiddle .plus (alg .within directhash)>(value)
General_COSE_Hash<alg, value> = #6.<hashmiddle>([
    hashAlg: alg .within (int .ne directhash / tstr),
    hashValue: value .within bstr ])
hashmiddle = 18556
directhash = (-256 .. -1) / (1 .. 255)
```

Figure 1: Generic CDDL for Tags for Bare Hash Values

An example for the SHA-256 hash of "hello world" in CBOR diagnostic notation:

```
18540(
  h'b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9')
```

The same in CBOR pretty printed hex:

===== NOTE: '\ ' line wrapping per RFC 8792 =====

```
d9 486c                                # tag(18540)
 58 20                                # bytes(32)
  \
  b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9
```

As none has been registered, no real example can be given for a hash algorithm with an identifier not in the standard range, but if -4711 were such an identifier, a hash with an explicit algorithm number could look like:

```
18556([-4711, h'12341234123412341234123412341234123412341234'])
```

Note that not all tags assigned in this section do parallel an algorithm that is a cryptographic hash algorithm. Where this is not the case, there currently is no defined semantics for this tag, but the tags are assigned anyway. The semantics of tags that parallel algorithm assignments other than for cryptographic hash functions could be defined by a future version of this specification.

Note also that the cryptographic hashes in the content of the tag are not protected; any further protection (confidentiality, integrity) needs to be provided in the surrounding data structure, storage system, or communication channel.

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 [RFC9254] 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	YANG enumeration datatype	6.6
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 numbers defined for YANG-CBOR

[I-D.bormann-cbor-yang-standin] proposes to employ additional tags to enable the use of efficient binary encodings for certain frequently used YANG data types [RFC9911].

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 "雨" (U+96E8) in UTF-8, which represents rain in a number of languages.)

6. Datatypes

6.1. Advanced arithmetic

General information about the representation of numbers in CBOR can be found in [STD94] as well as [I-D.bormann-cbor-numbers].

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 the column "Reference" of 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
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 RFC 8949 [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 RFC 8949 [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 2 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 2: 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
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 ([RFC9562], Section 4)	https://github.com/lucas-clemente/cbor-specs/blob/master/uuid.md	Lucas_Clemente
48	byte string	IEEE MAC Address	[RFC9542]	
52	byte string or array	IPv4, [prefixlen, IPv4], [IPv4, prefixpart]	[RFC9164]	
54	byte string or array	IPv6, [prefixlen, IPv6], [IPv6, prefixpart]	[RFC9164]	
257	byte string	Binary MIME message	http://peteroupc.github.io/CBOR/binarymime.html	Peter Occil
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
1048	byte	IEEE OUI/CID	[RFC9542]	

	string		
+-----+	+-----+	+-----+	+-----+

Table 7: Select Domain-Specific Tags

Notes:

- * In the registration for Tag 37, the reference for UUID has been updated from [RFC4122] to [RFC9562]. The new RFC has a somewhat different internal structure; the definition of the layout of a binary UUID is now distributed over Section 5 of [RFC9562].
- * Tags 48, 52, and 54 are recommended for new designs that need to represent MAC addresses, IP addresses, and IP address prefixes; they essentially replace tags 260 and 261, which continue to be available for their existing applications.
- * Tag 263 describes a different kind of Hexadecimal string (sequence of hex digit pairs, same layout as tag 23) from the hex-string defined by the YANG data type hex-string (sequence of hex digit pairs separated by colons, Section 3 of [RFC9911]).

7.1. Human-readable Text

Tag	Data Item	Semantics	Reference
38	array	Language-tagged string	Appendix A of [RFC9290]

Table 8: Select Tags for Human-readable Text

Tag 38 was originally registered by Peter Occil in <http://peteroupc.github.io/CBOR/langtags.html> (<http://peteroupc.github.io/CBOR/langtags.html>); it has since been adopted and extended in Appendix A of [RFC9290], where a detailed definition of the tag and a few simple examples for its use are provided.

The problem that this tag was designed to solve is that text strings often need additional information to be properly presented to a human. While Unicode (and the UTF-8 form of Unicode used in CBOR) define the characters, additional information about the human language in use and the writing direction appropriate for the text given are often required.

The need to provide language information with text has been well-known for a while and led to a common form for this information, the language tag, defined in [BCP47].

Less well-known is the need to provide separate directionality information as well. The need for this information is demonstrated in [W3C-STRINGS-BIDI], which points out that it is "actually a bad idea to rely on language information to apply direction" and points out further reference information on this. [W3C-BIDI-USE-CASES] shows more examples for language tags and directionality, while [W3C-UBA-BASICS] provides an introduction to the way browsers, where "the order of characters in memory (logical) is not the same as the order in which they are displayed (visual)", "produce the correct order at the time of display" (Unicode Bidirectional Algorithm).

Tag 38 meets the requirements of its specific application in [RFC9290], which could be summarized as: Supplying the necessary information to present isolated, linear, comparatively small pieces of human-readable text. It neither addresses more complex requirements of specific languages such as [W3C-SIMPLE-RUBY], nor does it address requirements for more complex structure in texts such as emphasis, lists, or tables. These more complex requirements are typically met by specific media types such as HTML [HTML].

7.2. 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	[RFC9581]
1002	map	duration	[RFC9581]
1003	map	period	[RFC9581]

Table 9: 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 3 into application-specific CDDL as needed.

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

Figure 3: 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 RFC 8949 [STD94].)

These are all documented under <http://cbor.schmorp.de/>; the last pathname component is given in Table 10.

(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
29	unsigned integer	reference nth marked value	value-sharing
22098	multiple	hint that indicates an additional level of indirection	indirection

Table 10: 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>). This is an analog to tag 24 for embedded Encoded CBOR Data Items.

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 identified as to which of these variants is used, 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 11: Tag numbers for UTF-8 variants

9. Application-specific

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

	Tag	Tag Author	Short Description	Reference
	number	content		
	39	multiple	Identifier	https://github.com/lucas-clemente/cbor-
lucas-	Lucas			specs/blob/master/id.md (https://github.com/
	Clemente			clemente/cbor-specs/blob/master/id.md)
	42	byte	IPLD content identifier	https://github.com/ipld/cid-cbor/
	Volker			
		string		(https://github.com/ipld/cid-cbor/)
	Mische			
	103	array	Geographic Coordinates	https://github.com/allthingstalk/cbor/blob/m
aster/	Danilo			CBOR-Tag103-Geographic-Coordinates.md
	Vidovic			(https://github.com/allthingstalk/cbor/blob/
master/				CBOR-Tag103-Geographic-Coordinates.md)
	104	multiple	Geographic Coordinate	[I-D.clarke-cbor-crs]
			Reference System WKT or	
			EPSG number	
	120	multiple	Internet of Things Data	https://github.com/allthingstalk/cbor/blob/m
aster/	Danilo			CBOR-Tag120-Internet-of-Things-Data-Points.m
			Point	
	Vidovic			(https://github.com/allthingstalk/cbor/blob/
master/				CBOR-Tag120-Internet-of-Things-Data-Points.m
d)				
	258	array	Mathematical finite set	https://github.com/input-output-hk/cbor-sets
-	Alfredo			spec/blob/master/CBOR_SETS.md (https://github/
b.com/	Di			input-output-hk/cbor-sets-spec/blob/master/
	Napoli			CBOR_SETS.md)
	259	map	Map datatype with key-	https://github.com/shanewholloway/js-cbor-
	Shane			

```

it- | | | | value operations (e.g. | codec/blob/master/docs/CBOR-259-spec--explic
s-cbor- | | | | |.get()/.set()/.delete())| maps.md (https://github.com/shanewholloway/j
it- | | | | | codec/blob/master/docs/CBOR-259-spec--explic
| | | | | maps.md)
+-----+-----+-----+-----+
-----+-----+-----+-----+

```

Table 12: Application-specific Tags

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 (Section 12) for encoding data composed of enumerated alternatives:

Tags	Data Item	Meaning
121..127	any	alternatives 0..6, 1+1 encoding
1280..1400	any	alternatives 7..127, 1+2 encoding
101	array [uint, any]	alternatives as given by the uint + 128

Table 13: 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 RFC 8949 [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.

For cases 0..6 and 7..127, the tag value indicates the value of the alternative. For cases 128+, a single tag number is used with an enclosed two-element array that contains the case number and the value of the alternative.

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, structs 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 = #6.121(int) ; integer literal
/ #6.122([expr, expr]) ; addition
/ #6.123([expr, expr]) ; subtraction
/ #6.124(expr) ; unary negation
/ #6.125([expr, expr]) ; multiplication
/ #6.126([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.

10.2. Programming Aid for Simple Values

In a similar way, the present document also registers tag number 21334 as a fourth Invalid Tag. This tag is set aside specifically for use by CBOR implementations that have no natural way to represent Simple Values (Section 3.3 of RFC 8949 [STD94]) beyond false, true, or null in their programming interface. For instance, such implementations can represent simple(123) as 21334(123) in the programming interface.

10.3. Test Tag

CBOR implementations should handle tags that they do not understand, usually by creating an instance of an object that contains the tag number and the associated data item. In order to test this code, there needs to be a tag number that will never be allocated for some new semantics. This property also makes the tag number safe to use as a placeholder in documentation, such as in illustrative examples that show a hypothetical tag, without creating expectations for future registration.

This section specifies a CBOR tag for testing purposes:

Tag	TBD (Proposed: 1413829460)
Data item	Any
Semantics	Explicitly none.
Point of contact	Joe Hildebrand joe-ietf@cursive.net <mailto:joe-ietf@cursive.net>
Description of semantics	draft-bormann-cbor-notable-tags, Section 10.3

Table 14: Test Tag

10.3.1. Semantics

This tag is always intended to be represented in the same manner that an unimplemented tag would be in a given implementation.

10.3.2. Sample code

```
import {encode, decode, Tag} from 'cbor2';
const t = new Tag(1413829460, 'TEST');
const bytes = encode(t); // 0xda544553546454455354
const t2 = decode(bytes); // 1413829460('TEST')
```

11. Managing redundancy

A CBOR data item often has appreciable internal redundancy, for instance by serializing certain strings again and again.

Data compression specifications such as deflate [RFC1951], gzip [RFC1952], brotli [RFC7932], zstd [RFC8878], and dcb/dcz [RFC9842] provide powerful ways to provide compressed representations of encoded data items that exhibit internal redundancy. While these can provide very good compression ratios, the disadvantage of applying traditional data compression is that the serialized data need to undergo a separate compression pass at the producer and a decompression pass at the consumer before the data items can be accessed again.

An alternative is to perform some management of redundancy not at the level of serialized data, but at the data model level, i.e., within the data items interchanged. This can enable a CBOR library to perform redundancy reduction ("packing") and unpacking on the fly. This section discusses tag sets that can be used for this.

11.1. Packed CBOR

A comprehensive approach to reducing redundancy by packing is provided by the set of tags and simple values defined in [I-D.ietf-cbor-packed], which see.

11.2. Stringref

A very simple mechanism that is focused just on the redundancy of repeated (byte or text) strings is `_stringref_` [STRINGREF], originally designed in the context of the Perl platform (Section 8.1), but now implemented on other platforms, too.

With some additional care, the stringref mechanism can be used in a general fashion; this section will briefly introduce stringref and ways to use it that are fully interoperable.

11.2.1. Stringref Mechanism

Stringref works by silently entering (byte or text) strings in a table that maps between unsigned integers ("index" values) and the string value (including whether it is a text or a byte string). On the producer side, an entry is added to the table when the specific string value is encoded for the first time. This allows all further copies of the same string to be expressed by just referencing the table entry by its index (sequence number). This reference is expressed via tag 25, carrying the index as an unsigned integer. The table entry is only added if representing the string itself would be longer than a reference via tag 25, i.e., the string needs to have a minimum length in bytes as per Table 15 to get a table entry.

string index	minimum string length (bytes)
0 .. 23	3
24 .. 255	4
256 .. 65535	5
65536 .. 4294967295	7
4294967296 ..	11

Table 15: Stringref: Minimum String Length for
Creating a New Table Entry

Note that there is no requirement for the producer of the encoded CBOR data item to actually make use of a table entry; this means that inexact ("lossy") representations of the table can be used at the producer, as long as the highest index is properly incremented on any string actually encoded as such. (If an actual string is indeed emitted again, the new copy gets a new table entry.)

The consumer side follows the decisions of the producer, i.e., any (text or byte) string encountered in the encoded data item that is minimum size or larger leads to a new entry in the table at the lowest current unfilled index and thus to incrementing the highest index in use.

11.2.2. Stringref Namespaces

Employing stringref (tag 25) places an onus on CBOR libraries both during emission and during ingestion: Both have to manage their own mapping table and examine it repeatedly, possibly adding an entry for another index/string mapping to it, each time a string is processed.

To incur this overhead only when needed, stringref references are only valid within the tag content of a tag with tag number 256, `_stringref-namespace_` (tag content: any). A library that implements stringref ingestion can switch on stringref processing when that tag is encountered, and switch it off again when its processing leaves the tag 256 tag content. A producer can localize the effort needed at both ends by placing the tag at a position in the hierarchy that merits the additional processing.

Stringref namespaces also provide a basic form of composability: If another tag 256 is encountered within the content of a tag 256, a new mapping table ("namespace") is created and the processing of strings and tags 25 within the tag content of the inner tag operates starting at index 0, independently of the processing of the tag content of the outer tag outside the inner tag. (After leaving the inner tag 256, the processing continues using the table generated by the outer tag 256.)

11.2.3. Serialization Considerations

As defined, stringref has a dependency on serialization order. Like JSON objects, CBOR maps do not have a defined order in processing, and both encoders and decoders might employ their own preferred order when that is beneficial on the specific platform. Stringref's implicit filling of the mapping table only works correctly when producer and consumer agree on this order.

So far, this looming interoperability problem has kept stringref confined to some niche applications. However, stringref can be used without a danger of disagreeing map orders in conjunction with a deterministic encoding serialization constraint, such as CDE, which binds both sides to the same ordering of entries in a map [I-D.ietf-cbor-cde].

Using CDE also solves another problem with stringref as defined: The tags' specification says that it only applies to definite-length-encoded strings. Here as well, the component doing the stringref processing may have no control over which serialization is used in the encoder and no information about the serialization used from the decoder. Since CDE incorporates the serialization constraint "definite length only" (DLO), no disagreement can happen when those serialization constraints are followed.

The present specification therefore recommends using stringref only in environments where a deterministic encoding with a DLO serialization constraint, such as CDE (or LDE), is followed.

12. IANA Considerations

In the registry "CBOR Tags" [IANA.cbor-tags], IANA has allocated the first to third tag in Table 16 from the First Come First Served (FCFS) space, with the present document as the specification reference. IANA also has allocated the tags in the next seven rows from the Specification Required space, with the present document as the specification reference, as well as the tag in the final row from the FCFS range.

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	(none valid)	always invalid	draft-bormann-cbor-notable-tags, Section 10.1
63	byte string	Encoded CBOR Sequence [RFC8742]	draft-bormann-cbor-notable-tags, Section 2.1
21065	text string	I-Regexp	draft-bormann-cbor-notable-tags, Section 2.1; [RFC9485]
18300 to 18555 (inclusive)	byte string	Bare Hash value (COSE algorithm -256 to -1)	draft-bormann-cbor-notable-tags, Section 3.1.1
18556	array	[COSE algorithm identifier, Bare Hash value]	draft-bormann-cbor-notable-tags, Section 3.1.1
18557 to 18811 (inclusive)	byte string	Bare Hash value (COSE algorithm 1 to 255)	draft-bormann-cbor-notable-tags, Section 3.1.1
108	byte string	Expected conversion to	draft-bormann-cbor-

		base16 encoding (lowercase)	notable-tags, Section 2.1, Paragraph 2, Item 2
21334	uint	(always invalid in interchange) programming aid for simple values	draft- bormann-cbor- notable-tags, Section 10.2
1413829460	any	explicitly none	draft- bormann-cbor- notable-tags, Section 10.3

Table 16: Values for Tags

In addition, IANA has allocated the tags from Table 13, with a reference to the present document.

13. 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.

14. References

14.1. Normative References

[I-D.ietf-cbor-cde]

Bormann, C., "CBOR Common Deterministic Encoding (CDE)", Work in Progress, Internet-Draft, draft-ietf-cbor-cde-13, 13 October 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-cbor-cde-13>>.

[IANA.cbor-tags]

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

[IANA.cose]

IANA, "CBOR Object Signing and Encryption (COSE)", <<https://www.iana.org/assignments/cose>>.

`[IANA.named-information]`

IANA, "Named Information",
<<https://www.iana.org/assignments/named-information>>.

[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/rfc/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/rfc/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/rfc/rfc8746>>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE):
Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053,
August 2022, <<https://www.rfc-editor.org/rfc/rfc9053>>.

[RFC9054] Schaad, J., "CBOR Object Signing and Encryption (COSE):
Hash Algorithms", RFC 9054, DOI 10.17487/RFC9054, August
2022, <<https://www.rfc-editor.org/rfc/rfc9054>>.

[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/rfc/rfc9132>>.

[RFC9360] Schaad, J., "CBOR Object Signing and Encryption (COSE):
Header Parameters for Carrying and Referencing X.509
Certificates", RFC 9360, DOI 10.17487/RFC9360, February
2023, <<https://www.rfc-editor.org/rfc/rfc9360>>.

[STD63] Internet Standard 63,
<<https://www.rfc-editor.org/info/std63>>.
At the time of writing, this STD comprises the following:

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>>.

[STD94] Internet Standard 94,
<<https://www.rfc-editor.org/info/std94>>.

At the time of writing, this STD comprises the following:

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>>.

- [STD96] Internet Standard 96, <<https://www.rfc-editor.org/info/std96>>.
At the time of writing, this STD comprises the following:

Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <<https://www.rfc-editor.org/info/rfc9052>>.

Schaad, J., "CBOR Object Signing and Encryption (COSE): Countersignatures", STD 96, RFC 9338, DOI 10.17487/RFC9338, December 2022, <<https://www.rfc-editor.org/info/rfc9338>>.

14.2. Informative References

- [BCP26] Best Current Practice 26, <<https://www.rfc-editor.org/info/bcp26>>.
At the time of writing, this BCP comprises the following:

Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

- [BCP47] Best Current Practice 47, <<https://www.rfc-editor.org/info/bcp47>>.
At the time of writing, this BCP comprises the following:

Phillips, A., Ed. and M. Davis, Ed., "Matching of Language Tags", BCP 47, RFC 4647, DOI 10.17487/RFC4647, September 2006, <<https://www.rfc-editor.org/info/rfc4647>>.

Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <<https://www.rfc-editor.org/info/rfc5646>>.

- [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 JTC1 SC22 WG21
N4860, March 2020,
<<https://isocpp.org/files/papers/N4860.pdf>>.

[HTML]

WHATWG, "HTML — Living Standard",
<<https://html.spec.whatwg.org>>.

[I-D.bormann-cbor-numbers]

Bormann, C., "On Numbers in CBOR", Work in Progress,
Internet-Draft, draft-bormann-cbor-numbers-02, 7 July
2025, <<https://datatracker.ietf.org/doc/html/draft-bormann-cbor-numbers-02>>.

[I-D.bormann-cbor-yang-standin]

Bormann, C. and M. Matjka, "Stand-in Tags for YANG-CBOR",
Work in Progress, Internet-Draft, draft-bormann-cbor-yang-
standin-02, 30 August 2025,
<<https://datatracker.ietf.org/doc/html/draft-bormann-cbor-yang-standin-02>>.

[I-D.clarke-cbor-crs]

Clarke, T., "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://datatracker.ietf.org/doc/html/draft-clarke-cbor-crs-02>>.

[I-D.draft-bormann-cbor-time-tag-01]

Bormann, C., Gamari, B., and H. Birkholz, "Concise Binary
Object Representation (CBOR) Tags for Time, Duration, and
Period", Work in Progress, Internet-Draft, draft-bormann-
cbor-time-tag-01, 14 June 2017,
<<https://datatracker.ietf.org/doc/html/draft-bormann-cbor-time-tag-01>>.

[I-D.ietf-cbor-packed]

Bormann, C. and M. Gtschow, "Packed CBOR", Work in
Progress, Internet-Draft, draft-ietf-cbor-packed-19, 2
February 2026, <<https://datatracker.ietf.org/doc/html/draft-ietf-cbor-packed-19>>.

- [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://datatracker.ietf.org/doc/html/draft-trammell-rains-protocol-05>>.
- [IEEE754] IEEE, "IEEE Standard for Floating-Point Arithmetic", IEEE Std 754-2019, DOI 10.1109/IEEESTD.2019.8766229, <<https://ieeexplore.ieee.org/document/8766229>>.
- [RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, DOI 10.17487/RFC1951, May 1996, <<https://www.rfc-editor.org/rfc/rfc1951>>.
- [RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, DOI 10.17487/RFC1952, May 1996, <<https://www.rfc-editor.org/rfc/rfc1952>>.
- [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/rfc/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/rfc/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/rfc/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/rfc/rfc7049>>.
- [RFC7322] Flanagan, H. and S. Ginoza, "RFC Style Guide", RFC 7322, DOI 10.17487/RFC7322, September 2014, <<https://www.rfc-editor.org/rfc/rfc7322>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/rfc/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/rfc/rfc7519>>.
- [RFC7932] Alakuijala, J. and Z. Szabadka, "Brotli Compressed Data Format", RFC 7932, DOI 10.17487/RFC7932, July 2016, <<https://www.rfc-editor.org/rfc/rfc7932>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/rfc/rfc7950>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/rfc/rfc7951>>.
- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/rfc/rfc8152>>.
- [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/rfc/rfc8259>>.
- [RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <<https://www.rfc-editor.org/rfc/rfc8742>>.
- [RFC8878] Collet, Y. and M. Kucherawy, Ed., "Zstandard Compression and the 'application/zstd' Media Type", RFC 8878, DOI 10.17487/RFC8878, February 2021, <<https://www.rfc-editor.org/rfc/rfc8878>>.
- [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/rfc/rfc8943>>.
- [RFC9164] Richardson, M. and C. Bormann, "Concise Binary Object Representation (CBOR) Tags for IPv4 and IPv6 Addresses and Prefixes", RFC 9164, DOI 10.17487/RFC9164, December 2021, <<https://www.rfc-editor.org/rfc/rfc9164>>.

- [RFC9254] Veillette, M., Ed., Petrov, I., Ed., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", RFC 9254, DOI 10.17487/RFC9254, July 2022, <<https://www.rfc-editor.org/rfc/rfc9254>>.
- [RFC9290] Fossati, T. and C. Bormann, "Concise Problem Details for Constrained Application Protocol (CoAP) APIs", RFC 9290, DOI 10.17487/RFC9290, October 2022, <<https://www.rfc-editor.org/rfc/rfc9290>>.
- [RFC9485] Bormann, C. and T. Bray, "I-Regexp: An Interoperable Regular Expression Format", RFC 9485, DOI 10.17487/RFC9485, October 2023, <<https://www.rfc-editor.org/rfc/rfc9485>>.
- [RFC9542] Eastlake 3rd, D., Abley, J., and Y. Li, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 9542, DOI 10.17487/RFC9542, April 2024, <<https://www.rfc-editor.org/rfc/rfc9542>>.
- [RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <<https://www.rfc-editor.org/rfc/rfc9562>>.
- [RFC9581] Bormann, C., Gamari, B., and H. Birkholz, "Concise Binary Object Representation (CBOR) Tags for Time, Duration, and Period", RFC 9581, DOI 10.17487/RFC9581, August 2024, <<https://www.rfc-editor.org/rfc/rfc9581>>.
- [RFC9842] Meenan, P., Ed. and Y. Weiss, Ed., "Compression Dictionary Transport", RFC 9842, DOI 10.17487/RFC9842, September 2025, <<https://www.rfc-editor.org/rfc/rfc9842>>.
- [RFC9911] Schnwlder, J., Ed., "Common YANG Data Types", RFC 9911, DOI 10.17487/RFC9911, December 2025, <<https://www.rfc-editor.org/rfc/rfc9911>>.
- [STRINGREF] Lehmann, M. A., (Specification for Tags 256 and Tag 25), <<http://cbor.schmorp.de/stringref>>.
- [W3C-BIDI-USE-CASES] "Use cases for bidi and language metadata on the Web", 6 May 2021, <<https://www.w3.org/International/articles/lang-bidi-use-cases/>>.

[W3C-SIMPLE-RUBY]

"W3C Rules for Simple Placement of Japanese Ruby", W3C
First Public Working Draft, 9 June 2020,
<<https://www.w3.org/TR/simple-ruby/>>.

[W3C-STRINGS-BIDI]

"Strings and bidi", 31 July 2017,
<<https://www.w3.org/International/articles/strings-and-bidi/>>.

[W3C-UBA-BASICS]

"Unicode Bidirectional Algorithm basics", 9 August 2016,
<<https://www.w3.org/International/articles/inline-bidi-markup/uba-basics>>.

List of Figures

- Figure 1: Generic CDDL for Tags for Bare Hash Values
- Figure 2: CDDL for extended arithmetic tags
- Figure 3: CDDL for calendar date tags (RFC8943)

List of Tables

- Table 1: Tag numbers defined in RFC 7049
- Table 2: Tag numbers defined in RFC9052, COSE, and RFC 9338
- Table 3: Tag number defined for RFC 8392 CBOR Web Token (CWT)
- Table 4: Tag numbers defined for YANG-CBOR
- Table 5: Tags for advanced arithmetic
- Table 6: Tag numbers defined for Arrays
- Table 7: Select Domain-Specific Tags
- Table 8: Select Tags for Human-readable Text
- Table 9: Tag numbers for date and time
- Table 10: Tag numbers that aid the Perl platform
- Table 11: Tag numbers for UTF-8 variants
- Table 12: Application-specific Tags
- Table 13: Tags for Enumerated Alternative Data Items
- Table 14: Test Tag
- Table 16: Values for Tags

Acknowledgements

(Many, TBD)

Contributors

Peter Occil
Email: poccil14 at gmail dot com

Peter Occil registered tags 30, 264, 265, 268270 (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

Joe Hildebrand
Email: joe-ietf@cursive.net

Joe Hildebrand contributed the test tag and registered the JavaScript Regexp tag.

Jane Doe
To do

Further contributors will be listed here as text is added.

Please stay tuned.

Author's Address

Carsten Bormann
Universitt Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org