Network Working Group J. Roatch
Internet-Draft
Intended status: Informational C. Bormann
Expires: February 9, 2018 Universitaet Bremen TZI
August 08, 2017

Concise Binary Object Representation (CBOR) Tags for Typed Arrays
draft-jroatch-cbor-tags-06

Abstract

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

The present document makes use of this extensibility to define a number of CBOR tags for typed arrays of numeric data, as well as two additional tags for multi-dimensional and homogeneous arrays. It is intended as the reference document for the IANA registration of the CBOR tags defined.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Terminology 3
2. Typed Arrays 3
2.1. Types of numbers 3
3. Additional Array Tags 4
3.1. Multi-dimensional Array 5
3.2. Homogeneous Array 5
4. Discussion 6
5. CDDL typenames 7
6. IANA Considerations 8
7. Security Considerations 9
8. References 10
8.1. Normative References 10
8.2. Informative References 10
Contributors 10
Acknowledgements 10
Authors' Addresses 11

1. Introduction

The Concise Binary Object Representation (CBOR, [RFC7049]) provides for the interchange of structured data without a requirement for a pre-agreed schema. RFC 7049 defines a basic set of data types, as well as a tagging mechanism that enables extending the set of data types supported via an IANA registry.

Recently, a simple form of typed arrays of numeric data have received interest both in the Web graphics community [TypedArray] and in the JavaScript specification [TypedArrayES6], as well as in corresponding implementations [ArrayBuffer].

Since these typed arrays may carry significant amounts of data, there is interest in interchanging them in CBOR without the need of lengthy conversion of each number in the array.

This document defines a number of interrelated CBOR tags that cover these typed arrays, as well as two additional tags for multi-dimensional and homogeneous arrays. It is intended as the reference document for the IANA registration of the tags defined.

1.1. Terminology

The term "byte" is used in its now customary sense as a synonym for "octet". Where bit arithmetic is explained, this document uses the notation familiar from the programming language C (including C++14's 0bnnn binary literals), except that the operator "**" stands for exponentiation.

2. Typed Arrays

Typed arrays are homogeneous arrays of numbers, all of which are encoded in a single form of binary representation. The concatenation of these representations is encoded as a single CBOR byte string (major type 2), enclosed by a single tag indicating the type and encoding of all the numbers represented in the byte string.

2.1. Types of numbers

Three classes of numbers are of interest: unsigned integers (uint), signed integers (twos' complement, sint), and IEEE 754 binary floating point numbers (which are always signed). For each of these classes, there are multiple representation lengths in active use:

Length ll	uint	sint	float
0	uint8	sint8	binary16
1	uint16	sint16	binary32
2	uint32	sint32	binary64
3	uint64	sint64	binary128

Table 1: Length values

Here, sintN stands for a signed integer of exactly N bits (for instance, sint16), and uintN stands for an unsigned integer of exactly N bits (for instance, uint32). The name binaryN stands for the number form of the same name defined in IEEE 754.

Since one objective of these tags is to be able to directly ship the ArrayBuffers underlying the Typed Arrays without re-encoding them, and these may be either in big endian (network byte order) or in little endian form, we need to define tags for both variants.

In total, this leads to 24 variants. In the tag, we need to express the choice between integer and floating point, the signedness (for integers), the endianness, and one of the four length values.

In order to simplify implementation, a range of tags is being allocated that allows retrieving all this information from the bits of the tag: Tag values from 64 to 87 (0x40 to 0x57).

The value is split up into 5 bit fields: 0b010_f_s_e_ll, as detailed in Table 2.

Field	Use
0b010	a constant '010'
f	0 for integer, 1 for float
s	0 for unsigned integer or float, 1 for signed integer
e	0 for big endian, 1 for little endian
ll	A number for the length (Table 1).

Table 2: Bit fields in the low 8 bits of the tag

The number of bytes in each array element can then be calculated by "2**(f + ll)" (or "1 << (f + ll)" in a typical programming language). (Notice that f and ll are the lsb of each nibble (4bit) in the byte.)

In the CBOR representation, the total number of elements in the array is not expressed explicitly, but implied from the length of the byte string and the length of each representation. It can be computed inversely to the previous formula: "bytelength >> (f + ll)".

For the uint8/sint8 values, the endianness is redundant. Only the big endian variant is used. As a special case, what would be the little endian variant of uint8 is used to signify that the numbers in the array are using clamped conversion from integers, as described in more detail in Section 7.1 of [TypedArrayUpdate].

3. Additional Array Tags

This specification defines two additional array tags. The Multi-dimensional Array tag can be combined with classical CBOR arrays as well as with Typed Arrays in order to build multi-dimensional arrays with constant numbers of elements in the sub-arrays. The Homogeneous Array tag can be used to facilitate the ingestion of homogeneous classical CBOR arrays, providing performance advantages even when a Typed Array does not apply.

3.1. Multi-dimensional Array

Tag: TBD40

Data Item: array (major type 4) of two arrays, one array (major type 4) of dimensions, and one array (major type 4, a Typed Array, or a Homogeneous Array) of elements

A multi-dimensional array is represented as a tagged array that contains two (one-dimensional) arrays. The first array defines the dimensions of the multi-dimensional array (in the sequence of outer dimensions towards inner dimensions) while the second array represents the contents of the multi-dimensional array. If the second array is itself tagged as a Typed Array then the element type of the multi-dimensional array is known to be the same type as that of the Typed Array. Data in the Typed Array byte string consists of consecutive values where the last dimension is considered contiguous (row-major order).

```
uint16_t a[2][3] = {
  {0, 1, 2},   /* row 0 */
  {3, 4, 5},
};

<Tag TBD40> # multi-dimensional array tag
   82       # array(2)
     82      # array(2)
       02     # unsigned(2) 1st Dimension
       03     # unsigned(3) 2nd Dimension
     d8 41   # uint16 array
       4a     # byte string(12)
         00 00 # unsigned(0)
         00 01 # unsigned(1)
         00 02 # unsigned(2)
         00 03 # unsigned(3)
         00 04 # unsigned(4)
         00 05 # unsigned(5)
```

Figure 1: Multi-dimensional array in C and CBOR

3.2. Homogeneous Array

Tag: TBD41

Data Item: array (major type 4)

This tag provides a hint to decoders that the array tagged by it has elements that are all of the same application type. The element type

of the array is thus determined by the application type of the first array element. This can be used by implementations in strongly typed languages while decoding to create native homogeneous arrays of specific types instead of ordered lists.

Which CBOR data items constitute elements of the same application type is specific to the application. However, type systems of programming languages have enough commonality that an application should be able to create portable homogeneous arrays.

```
bool boolArray[2] = { true, false };

<Tag TBD41>  # Homogeneous Array Tag
   82           #array(2)
      F5        # true
      F4        # false
```

Figure 2: Homogeneous array in C and CBOR

4. Discussion

Support for both little- and big-endian representation may seem out of character with CBOR, which is otherwise fully big endian. This support is in line with the intended use of the typed arrays and the objective not to require conversion of each array element.

This specification allocates a sizable chunk out of the single-byte tag space. This use of code point space is justified by the wide use of typed arrays in data interchange.

Applying a Homogeneous Array tag to a Typed Array would be redundant and is therefore not provided by the present specification.

5. CDDL typenames

For the use with CDDL [I-D.ietf-cbor-cddl], the typenames defined in Figure 3 are recommended:

```
ta-uint8 = #6.64(bstr)
ta-uint16be = #6.65(bstr)
ta-uint32be = #6.66(bstr)
ta-uint64be = #6.67(bstr)
ta-uint8-clamped = #6.68(bstr)
ta-uint16le = #6.69(bstr)
ta-uint32le = #6.70(bstr)
ta-uint64le = #6.71(bstr)
ta-sint8 = #6.72(bstr)
ta-sint16be = #6.73(bstr)
ta-sint32be = #6.74(bstr)
ta-sint64be = #6.75(bstr)
; reserved: #6.76(bstr)
ta-sint16le = #6.77(bstr)
ta-sint32le = #6.78(bstr)
ta-sint64le = #6.79(bstr)
ta-float16be = #6.80(bstr)
ta-float32be = #6.81(bstr)
ta-float64be = #6.82(bstr)
ta-float128be = #6.83(bstr)
ta-float16le = #6.84(bstr)
ta-float32le = #6.85(bstr)
ta-float64le = #6.86(bstr)
ta-float128le = #6.87(bstr)
homogeneous<array> = #6.TBD41(array)
multi-dim<dim, array> = #6.TBD40([dim, array])
```

Figure 3: Recommended typenames for CDDL

6. IANA Considerations

IANA is requested to allocate the tags in Table 3, with the present document as the specification reference.

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
TBD40	array of two arrays*	Multi-dimensional Array
TBD41	array	Homogeneous Array

Table 3: Values for Tags

*) TBD40 data item: second element of outer array in data item is native CBOR array (major type 4) or Typed Array (one of Tag 64..87)

RFC editor note: Please replace TBD40 and TBD41 by the tag numbers allocated by IANA throughout the document and delete this note.

7. Security Considerations

The security considerations of RFC 7049 apply; the tags introduced here are not expected to raise security considerations beyond those.

8. References

8.1. Normative References

[I-D.ietf-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR data structures", draft-ietf-cbor-cddl-00 (work in progress), July 2017.

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

8.2. Informative References

[ArrayBuffer]
Mozilla Developer Network, "JavaScript typed arrays", 2013, <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Typed_arrays>.

[TypedArray]
Vukicevic, V. and K. Russell, "Typed Array Specification", February 2011, <https://www.khronos.org/registry/typedarray/specs/1.0/>.

[TypedArrayES6]
"22.2 TypedArray Objects", in: ECMA-262 6th Edition, The ECMAScript 2015 Language Specification, June 2015, <http://www.ecma-international.org/ecma-262/6.0/#sec-typedarray-objects>.

[TypedArrayUpdate]
Herman, D. and K. Russell, "Typed Array Specification", July 2013, <https://www.khronos.org/registry/typedarray/specs/latest/>.

Contributors

Glenn Engel suggested the tags for multi-dimensional arrays and homogeneous arrays.

Acknowledgements

TBD

Authors' Addresses

Johnathan Roatch

Email: jroatch@gmail.com

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

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