Network Working Group   J. Roatch
Internet-Draft
Intended status: Informational   C. Bormann
Expires: August 18, 2014   Universitaet Bremen TZI
                                                        February 14, 2014

Concise Binary Object Representation (CBOR) Tags for Typed Arrays
draft-jroatch-cbor-tags-00

Abstract

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

The present document makes use of this extensibility to define a number of CBOR tags for typed arrays of numeric data. It is intended as the reference document for the IANA registration of the CBOR tags defined.

Table of Contents

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 JavaScript 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. It is intended as the reference document for the IANA registration of the tags defined.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The term "byte" is used in its now customary sense as a synonym for "octet".

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 defined in Section 7.1 of [TypedArrayUpdate].

3. 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.

4. IANA Considerations

IANA is requested to reserve 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

Table 3: Values for Tags

5. Security Considerations

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

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.

6.2. Informative References

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

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

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

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