CBOR

From Wikipedia, the free encyclopedia

CBOR (Concise Binary Object Representation) is a binary data serialization format loosely based on JSON. It is defined in IETF RFC 7049 (https://tools.ietf.org/html/rfc7049).[1]

Amongst other uses, it is the recommended data serialization layer for the CoAP Internet of Things protocol suite.[2]

Contents

Specification of the CBOR encoding

A CBOR encoded data is seen as a stream of data items. E.g.

CBOR Data	**Data Item 1**				**Data Item 2**				**Data Item X...**
Byte Count	1 Byte (CBOR Data Item Header)		Variable	Variable	1 Byte (CBOR Data Item Header)		Variable	Variable	etc...
Structure	Major Type	Additional Information	Payload Length (optional)	Data Payload (optional)	Major Type	Additional Information	Payload Length (optional)	Data Payload (optional)	etc...
Bit Count	3 Bits	5 Bits	8 Bits × Variable	8 Bits × Variable	3 Bits	5 Bits	8 Bits × Variable	8 Bits × Variable	etc..

How Major Type and Additional Type is handled in each data item?

Each data item behaviour is defined by the Major Type and Additional Type. The major type is used for selecting the main behaviour or type of each data item.

The additional type is additional information whose exact behaviour is dependent on the Major Type value.

CBOR Data Item Header

Below table illustrate how the CBOR data item header works

Major Type	Major Type Value		Additional Type Value (unsigned)	Additional Type Meaning	Payload Length Exist	Data Payload Exist
Unsigned Integer	0	0b000	**The 5-bit additional information is either the integer itself (for additional information values 0 through 23) or the length of additional data.**			
			0 to 23 (0x0 to 0x17) (0b00000 to 0b10111)	Used directly as the data payload. Thus keeping a compact size.	No	No
			24	Next Byte is uint8_t in Data Payload Section	No	Yes
			25	Next 2 Bytes uint16_t in Data Payload Section	No	Yes
			26	Next 4 Bytes is uint32_t in Data Payload Section	No	Yes
			27	Next 8 Bytes is uint64_t in Data Payload Section	No	Yes
Negative Integer	1	0b001	**The encoding follows the rules for unsigned integers (major type 0), except that the value is then -1 minus the encoded unsigned integer.**			
			0 to 23 (0x0 to 0x17) (0b00000 to 0b10111)	Used directly as the data payload. Thus keeping a compact size.	No	No
			24	Next 1 Byte in Data Payload	No	Yes
			25	Next 2 Bytes uint16_t in Data Payload	No	Yes
			26	Next 4 Bytes is uint32_t in Data Payload	No	Yes
			27	Next 8 Bytes is uint64_t in Data Payload	No	Yes
Byte String	2	0b010	**The string's length in bytes is represented following the rules for positive integers (major type 0).**			
			0 to 23 (0x0 to 0x17) (0b00000 to 0b10111)	Used directly as the data length specifier. Thus keeping a compact size.	No	Yes
			24	Next Byte is uint8_t for Payload Length	Yes	Yes
			25	Next 2 Bytes uint16_t for Payload Length	Yes	Yes
			26	Next 4 Bytes is uint32_t for Payload Length	Yes	Yes
			27	Next 8 Bytes is uint64_t for Payload Length	Yes	Yes
Text String	3	0b011	**A text string, specifically a string of Unicode characters that is encoded as UTF-8 [RFC3629].**			
			0 to 23 (0x0 to 0x17) (0b00000 to 0b10111)	Used directly as the data length specifier. Thus keeping a compact size.	No	Yes
			24	Next Byte is uint8_t for Payload Length	Yes	Yes
			25	Next 2 Bytes uint16_t for Payload Length	Yes	Yes
			26	Next 4 Bytes is uint32_t for Payload Length	Yes	Yes
			27	Next 8 Bytes is uint64_t for Payload Length	Yes	Yes
			Arrays are also called lists, sequences, or tuples. The length denotes the number of data items in array			

Array of Data Items	4	0b100	**rather than the byte length.**			
			0 to 23 (0x0 to 0x17) (0b00000 to 0b10111)	Used directly as the data length specifier. Thus keeping a compact size.	No	No
			24	Next Byte is uint8_t for Payload Length	Yes	No
			25	Next 2 Bytes uint16_t for Payload Length	Yes	No
			26	Next 4 Bytes is uint32_t for Payload Length	Yes	No
			27	Next 8 Bytes is uint64_t for Payload Length	Yes	No
			...	...	...	...
			31	Start of Indefinite Array till next corresponding "Break" Code.	No	No
Map of pairs of data items	5	0b101	**A map of pairs of data items. Maps are also called tables, dictionaries, hashes, or objects (in JSON).** **The length denotes the number of data items in array rather than the byte length.** **Every map entry takes two data items in sequential order, a key data item and a value data item.**			
			0 to 23 (0x0 to 0x17) (0b00000 to 0b10111)	Used directly as the data length specifier. Thus keeping a compact size.	No	No
			24	Next Byte is uint8_t for Payload Length	Yes	No
			25	Next 2 Bytes uint16_t for Payload Length	Yes	No
			26	Next 4 Bytes is uint32_t for Payload Length	Yes	No
			27	Next 8 Bytes is uint64_t for Payload Length	Yes	No
			...	...	...	...
			31	Start of Indefinite Map till next corresponding "Break" Code.	No	No
Semantic Tag	6	0b110	**Used for optional semantic tagging of other major types**			
			Tag ID	**Refer to https://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml for the semantic meaning of each tag.**		
			0-23	Standard Actions https://tools.ietf.org/html/rfc7049#section-2.4	No	No
			24-255	Specification Required	No	No
			256-18446744073709551615	First Come First Served	No	No
Primitives e.g. Break, Float,	7	0b111	**floating-point numbers and simple data types that need no content, as well as the "break" stop code**			
			0..23	Simple value (value 0..23 in Additional Type Value)	No	No
			24	Simple value (value 32..255 in following byte)	No	Yes
			25	IEEE 754 Half-Precision Float (16 bits follow)	No	Yes
			26	IEEE 754 Single-Precision Float (32 bits follow)	No	Yes
			27	IEEE 754 Double-Precision Float (64 bits follow)	No	Yes

Simple Values			28	Unassigned		
			29			
			30			
			31	"break" stop code for indefinite-length items	No	No

- Byte = 8bits

Primitives (Major Type = 7)

The primitives major type has a major type value of 7. It is used for Simple Data types, common complex float types, as well as control code.

	Major Type	**Additional Value**	**Extra Bytes (If Required)**							
Bytes	**0**		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Bit Size	**3 bits**	**5 bits**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**
Simple Value 0 to 23 (Value X)	7	X=0...23	Not Used							
Simple Value 24 to 255 (Value X)	7	24	X=24...255	Not Used						
IEEE 754 Half-Precision Float (16 bits follow)	7	25	16 bits IEEE 754	Not Used						
IEEE 754 Single-Precision Float (32 bits follow)	7	26	32 bits IEEE 754				Not Used			
IEEE 754 Double-Precision Float (64 bits follow)	7	27	64 bits IEEE 754							
Break From Indefinite Array Or Map	7	31	Not Used							

Break control code (Additional Type Value = 31)

This is a meta value, that is used in conjunction with arrays and maps set to indefinite length mode. This indicates to the CBOR parser to close the corresponding map or array level.

IEEE 754 Floats (Additional Type Value = 25 or 26 or 27)

This allows for storing floats, encoded as IEEE 754 float values.

Simple Value

Most simple values are either unassigned or reserved for future improvements.

However these are defined.

Simple Value	**Semantic**
20	Boolean False
21	Boolean True
22	Null
23	Undefined

Semantic Tag Registration

IANA has created the CBOR Tags registry, located in https://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml . Registration must contain these template.

Semantic Tag Type	Range	Template			
		Data Item	Semantic Description (Short Form)	Point Of Contact	Description Of Semantics (URL) ()
Standard Actions	0-23	Required	Required	N/A	N/A
Specification Required	24-255	Required	Required	N/A	N/A
First Come First Served	256-18446744073709551615	Required	Required	Required	Description is optional. The URL can point to an Internet-Draft or a web page.

https://tools.ietf.org/html/rfc7049#section-7.2

Implementations

Name	Primary author	Language	License	Source	Remarks
cbor-js	Patrick Gansterer	JavaScript	MIT	https://github.com/paroga/cbor-js	
node-cbor	Joe Hildebrand	JavaScript	MIT	https://github.com/hildjj/node-cbor	
CBOREncode	Pavel Gulbin	PHP	PHP	https://github.com/2tvenom/CBOREncode	
cbor	Pavel Gulbin	Go	WTFPL	https://github.com/2tvenom/cbor	
cbor_go	Brian Olson	Go	APL 2.0	https://github.com/brianolson/cbor_go	
go-codec	Ugorji Nwoke	Go	MIT	https://godoc.org/github.com/ugorji/go/codec	Also handles JSON, MsgPack and BinC.
rust-cbor	Andrew Gallant	Rust	MIT or Unlicense	https://github.com/BurntSushi/rust-cbor	
cbor-codec	Toralf Wittner	Rust	MPL 2.0	https://twittner.gitlab.io/cbor-codec/cbor/	
SwiftCBOR	greg@unrelenting.technology	Swift	Unlicense	https://github.com/myfreeweb/SwiftCBOR	
CBOR.jl	Saurav Sachidanand	Julia	MIT	https://github.com/saurvs/CBOR.jl	
Lua-CBOR	Kim Alvefur	Lua	MIT	https://www.zash.se/lua-cbor.html	
org.conman.cbor	Sean Conner	Lua	LGPL-3	https://github.com/spc476/CBOR	
cbor_py	Brian Olson	Python	APL 2.0	https://github.com/brianolson/cbor_py	
flynn	Fritz Conrad Grimpen	Python	MIT	https://github.com/fritz0705/flynn	

cbor2	Alex Grönholm	Python	MIT	https://github.com/agronholm/cbor2	
CBOR::XS	Marc Lehmann	Perl	GPL-3	http://software.schmorp.de/pkg/CBOR-XS.html	
cbor-ruby	Sadayuki Furuhashi Carsten Bormann	Ruby	APL 2.0	https://github.com/cabo/cbor-ruby	
libcbor-ruby	Pavel Kalvoda	Ruby	MIT	https://github.com/PJK/libcbor-ruby	Binding to libcbor.
cbor-erlang	Jihyun Yu	Erlang	BSD-3-clause	https://github.com/yjh0502/cbor-erlang	
excbor	Carsten Bormann	Elixir	not specified, ask the author	https://github.com/cabo/excbor	
CBOR	R. Kyle Murphy	Haskell	LGPL-3	https://github.com/orclev/CBOR	
borc	Joe Hildebrand Friedel Ziegelmayer	JavaScript	MIT	https://github.com/dignifiedquire/borc	Fork of node-cbor.
borc-refs	Joe Hildebrand Friedel Ziegelmayer Sandro Hawke	JavaScript	MIT	https://github.com/sandhawke/borc-refs	Fork of borc.
CBOR	Peter Occil	C#	Public domain software	https://github.com/peteroupc/CBOR	Also handles JSON.
Jackson	Tatu Saloranta	Java	APL-2.0	https://github.com/FasterXML/jackson-dataformats-binary/tree/master/cbor	Also handles other formats.
cbor-java	Constantin Rack	Java	APL-2.0	https://github.com/c-rack/cbor-java	
jacob	J.W. Janssen	Java	APL-2.0	https://github.com/jawi/jacob	
RIOT	Kevin Funk Jana Cavojska	C	LGPL-2.1	https://github.com/RIOT-OS/RIOT/blob/master/sys/cbor/cbor.c	Part of RIOT operating system.
cn-cbor	Joe Hildebrand Carsten Bormann	C	MIT	https://github.com/cabo/cn-cbor	
cbor-cpp	Stanislav Ovsyannikov	C++	APL-2.0	https://github.com/naphaso/cbor-cpp	
libcbor	Pavel Kalvoda	C	MIT	https://github.com/PJK/libcbor	
tinycbor	Intel	C	MIT	https://github.com/01org/tinycbor	
cbor-d	Andrey Penechko	D	Boost 1.0	https://github.com/MrSmith33/cbor-d	

clj-cbor	Greg Look	Clojure	Unlicense	https://github.com/greglook/clj-cbor	
JSON for Modern C++	Niels Lohmann	C++	MIT	https://github.com/nlohmann/json	Also handles JSON and MsgPack.
borabora	Christoph Engelbert	Java	APL-2.0	https://github.com/noctarius/borabora	
lua-ConciseSerialization	François Perrad	Lua	MIT	https://fperrad.github.io/lua-ConciseSerialization/	
flunn	Fritz Conrad Grimpen Sokolov Yura	Python	MIT	https://pypi.python.org/pypi/flunn	
cbor-qt	Anton Dutov	C++	Public domain	https://github.com/anton-dutov/cbor-qt	
cbor11	Jakob Varmose Bentzen	C++	Public domain	https://github.com/jakobvarmose/cbor11	
cborcpp	Alex Nekipelov	C++	MIT	https://github.com/nekipelov/cborcpp	
GoldFish	Vincent Lascaux	C++	MIT	https://github.com/OneNoteDev/GoldFish	
Library-Arduino-Cbor	Juanjo Tara	C++	APL-2.0	https://github.com/jjtara/Library-Arduino-Cbor	
serde_cbor	Pyfisch	Rust	Apache-2.0/MIT	https://github.com/pyfisch/cbor	

References

1. http://cbor.io/
2. http://coap.technology/

External links

- Online tool to convert from CBOR binary to Textual Representation and back. (http://cbor.me/)