Implementations

CBOR is simple enough to implement from scratch for a specific application.
For applications where that is not desirable, generic implementations are available for a variety of platforms. Many of these implementations stay private, but some are published with liberal open-source licenses such as the Apache 2.0 or the MIT license.
Many implementations use a simple API of the form:

`encoded = CBOR.encode(data)` → `data = CBOR.decode(encoded)`

JavaScript

JavaScript implementations are available both for in-browser use and for node.js.

Browser

A CBOR object can be installed via `bower install cbor` and used as an AMD module or global object in the browser e.g. in combination with Websockets…

View details » (https://github.com/paroga/cbor-js)

node.js

… and the server side for that might be written using node.js; install via: `npm install cbor`

View details » (https://github.com/hildjj/node-cbor)

Also, an implementation based on the higher speed C library tinyCBOR is now available:

View details » (https://www.npmjs.com/package/tinycbor)

cbor-sync provides an extensible CBOR encoder/decoder:

View details » (https://github.com/ARMmbed/cbor-sync)

Duktape

A CBOR binding for both C and JavaScript:

View details » (https://github.com/svaarala/duktape/tree/master/extras/cbor)

Dart

A CBOR encoder/decoder suite with a test suite that incorporates the encode/decode tests from RFC7049 and more is now available in the Darts package ecosystem (https://pub.dartlang.org/packages/cbor):

View details » (https://github.com/shamblett/cbor)

PHP

API: `\CBOR\CBOREncoder::encode($target)` and `\CBOR\CBOREncoder::decode($encoded_data)`

View details » (https://github.com/2tvenom/CBOREncode)

Go

An early Go implementation that feels like the JSON library:

View details » (https://github.com/2tvenom/cbor)

Another, more full-grown Go implementation:

View details » (https://github.com/brianolson/cbor_go)

Most recently, a comprehensive, high-performance implementation has become available as part of a larger set of data representation format en- and decoders:

View details » (https://godoc.org/github.com/ugorji/go/codec#CborHandle)

Rust

A Rust implementation is available that works with Cargo and is on crates.io (https://crates.io/crates/cbor):

View details » (https://github.com/BurntSushi/rust-cbor)

Another Rust implementation has also become available recently on crates.io (https://crates.io/crates/cbor-codec):

View details » (https://gitlab.com/twittner/cbor-codec)

An implementation using Serde (https://docs.serde.rs/serde/index.html), a framework for serializing and deserializing Rust data structures efficiently and generically, is available on crates.io (https://crates.io/crates/serde_cbor):

View details » (https://github.com/pyfisch/cbor)

Swift

A Swift implementation without a Foundation dependency (cross-platform ready):

View details » (https://github.com/myfreeweb/SwiftCBOR)

Another, work-in-progress Swift implementation that is geared towards integration in iOS and macOS via Cocoapods:

View details » (https://github.com/Hassaniiii/CBORSwift)

Julia

`CBOR.jl` is a Julia package for working with the CBOR data format, providing straightforward encoding and decoding for Julia types. Install via:

Lua

Lua-cbor is a pure Lua implementation of CBOR for Lua 5.1—5.3, which utilizes struct packing and bitwise operations if available:

View details » (https://www.zash.se/lua-cbor.html)

“The most comprehensive CBOR module in the Lua universe” supports everything mentioned in RFC 7049 and the extensions registered with the IANA so far. It comes with parts implemented in C.

View details » (https://github.com/spc476/CBOR)

lua-ConciseSerialization is a pure Lua implementation of CBOR with variants for both 5.1 and 5.3; install via `luarocks install lua-conciseserialization`.

View details » (http://fperrad.github.io/lua-ConciseSerialization/)

Python

Install a high-speed implementation via pypi: `pip install cbor` (and/or possibly `pip3 install cbor`)

View details » (https://github.com/brianolson/cbor_py)

Flynn's' simple API is inspired by existing Python serialisation modules like json and pickle:

View details » (https://github.com/fritz0705/flynn)

Flunn is a fork of flynn, fixing some compatibility issues and with some refactoring:

View details » (https://pypi.python.org/pypi/flunn)

Install a high quality implementation that supports most CBOR tags, including those for representing cyclic (recursive) references, via `pip install cbor2`

View details » (https://pypi.io/project/cbor2/)

Perl

Install a comprehensive implementation tailored to Perl's many features via: `cpan CBOR::XS`

You'll like the performance data…

View details » (http://pod.tst.eu/http://cvs.schmorp.de/CBOR-XS/XS.pm)

Ruby

A high-speed implementation has been derived from the MessagePack (https://msgpack.org) implementation for Ruby. Installation: `gem install cbor`

View details » (https://github.com/cabo/cbor-ruby)

Ruby bindings for libcbor (http://libcbor.org) are now available. Installation: `gem install libcbor`

View details » (https://github.com/PJK/libcbor-ruby)

Erlang, Elixir

cbor-erlang is a recent implementation in Erlang:

View details » (https://github.com/yjh0502/cbor-erlang)

An older Elixir implementation is also available:

`expm spec excbor --format scm | sh`

Or look at the source:

View details » (https://github.com/cabo/excbor)

Haskell

An early implementation has been on hackage (https://hackage.haskell.org/package/CBOR) for a while:

View details » (https://github.com/orclev/CBOR)

cborg (https://github.com/well-typed/cborg), a more recent implementation, aims for higher speed and more features, and comes with tools (http://hackage.haskell.org/package/cbor-tool) and a JSON converter (http://hackage.haskell.org/package/cborg-json). It is also the basis for the Haskell serialise (http://hackage.haskell.org/package/serialise) package:

View details » (http://hackage.haskell.org/package/cborg)

Clojure

`clj-cbor` is an extensible, native Clojure implementation of the CBOR format:

View Details » (https://github.com/greglook/clj-cbor)

C#, Java

A rather comprehensive implementation that addresses arbitrary precision arithmetic is available in both a C# and a Java version.

View details » (https://github.com/peteroupc/CBOR)

Java

A Java implementation as part of the popular Jackson (https://github.com/FasterXML/jackson) JSON library is at:

View Details » (https://github.com/FasterXML/jackson-dataformats-binary)

A Java 7 implementation focusing on test coverage and a clean separation of model, encoder and decoder is at:

View Details » (https://github.com/c-rack/cbor-java)

JACOB, a small CBOR encoder and decoder implemented in plain Java is at:

View Details » (https://github.com/jawi/jacob)

borabora supports graph queries and lazy decoding of stream elements:

View Details » (https://github.com/noctarius/borabora)

Cyborg supports a stream parsing/generation API without the need to adapt your data to a specific model:

View Details » (https://github.com/dwaite/cyborg)

C, C++

A CBOR implementation in C is part of the RIOT operating system for constrained nodes:

View Details » (https://github.com/RIOT-OS/RIOT/blob/master/sys/cbor/cbor.c)

A C implementation for highly constrained nodes, which achieves a full CBOR decoder in 880 bytes of ARM code (and now also includes an encoder), has recently become available.

View Details » (https://github.com/cabo/cn-cbor)

A basic C++ implementation is also available:

View Details » (https://github.com/naphaso/cbor-cpp)

libcbor (http://libcbor.org) provides a fully-fledged C99 implementation, including streaming and incremental processing functionality:

View details » (https://github.com/PJK/libcbor)

TinyCBOR is Intel's industrial strength C/C++ implementation of CBOR, as used in the IoTivity (https://www.iotivity.org/) framework:

View details » (https://github.com/01org/tinycbor)

JSON for Modern C++ (https://github.com/nlohmann/json) — a header-only C++ library that aims to promote JSON to a first-class data type in C++11 — now supports CBOR as (de)serialization format.

View details » (https://github.com/nlohmann/json#binary-formats-cbor-and-messagepack)

A Qt Implementation is at:

View details » (https://github.com/anton-dutov/cbor-qt)

GoldFish is a fast, header-only C++11 library for CBOR and JSON that minimizes memory allocation by providing a SAX-like, but pull-oriented interface. Currently only ported to Visual C++.

View details » (https://github.com/OneNoteDev/GoldFish)

jsoncons is a C++11, header-only library for JSON construction that supports encode to/decode from CBOR.

View details » (https://github.com/danielaparker/jsoncons)

CBOR-lite is a simple Modern C++ header-only implementation designed for encoding and decoding of CBOR-based application level-protocols.

View details » (https://bitbucket.org/isode/cbor-lite)

cborg seems to be a popular name for a CBOR implementation, here for C++ with a fluent interface:

View details » (https://github.com/ARMmbed/cborg)

```
Pkg.add("CBOR")
```

View details »
(https://github.com/saurvs/CBOR.jl)

A simple low-level streamed callback-based CBOR push parser in C and C++:

View details »
(https://github.com/vi/simple_cbor_stream_parse)

D

A compact D implementation with a Dub package (https://code.dlang.org/packages/cbor-d):

View Details »
(https://github.com/MrSmith33/cbor-d)

Carsten Bormann (mailto:cabo@tzi.org?subject=cbor.io) • TZI (http://tzi.de) • 2014–2018

Feedback: Create a github issue (https://github.com/cbor/cbor.github.io/issues/new) (free github account required) or just send mail (mailto:cabo@tzi.org?subject=cbor.io)