CBOR (RFC 7049)

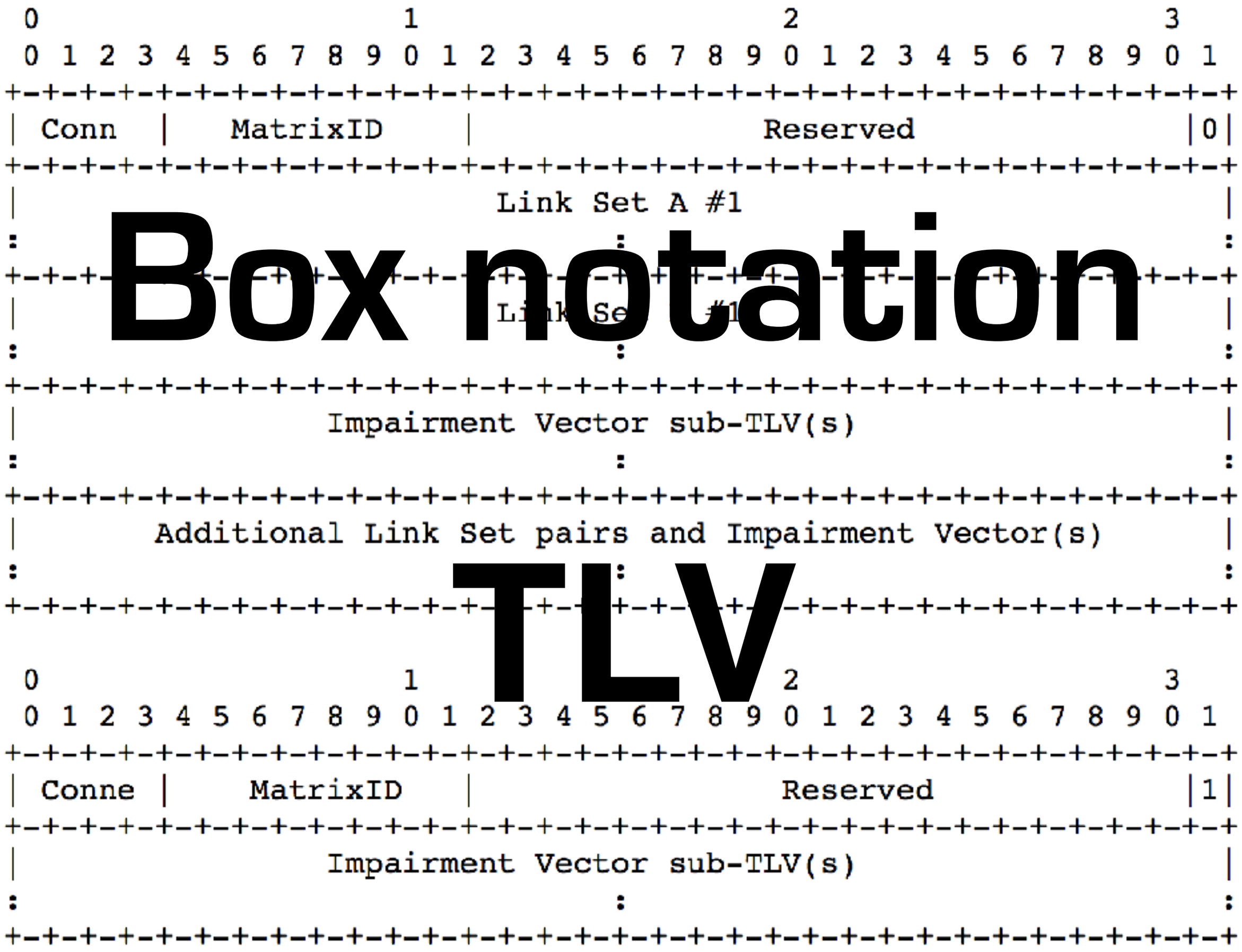
Concise Binary Object Representation

See also: IETF94 CBOR lightning tutorial
Carsten Bormann, 2015-11-01
http://www.tzi.de/~cabo/CBOR-2015-11-01.pdf

Slide stolen from Douglas Crockford

History of Data Formats

- **Ad Hoc**
- **Database Model**
- **Document Model**
- **Programming Language Model**


Box notation
TLV
0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Conn
MatrixID
Reserved
0
Link Set A #1
Impairment Vector sub-TLV(s)
Additional Link Set pairs and Impairment Vector(s)
Conne
1

XML

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
S:     epp-1.0.xsd">
S: <response>
S:  <result code="1000">
S:   <msg>Command completed successfully</msg>
S:  </result>
S:  <resData>
S:   <domain:infData
S:    xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
S:    xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0
S:    domain-1.0.xsd">
S:    <domain:name>3.8.0.0.6.9.2.3.6.1.4.4.e164.arpa</domain:name>
S:    <domain:roid>EXAMPLE1-REP</domain:roid>
S:    <domain:status s="ok"/>
S:    <domain:registrant>jd1234</domain:registrant>
S:    <domain:contact type="admin">sh8013</domain:contact>
S:    <domain:contact type="tech">sh8013</domain:contact>
S:    <domain:n[illegible]
S:     <domain:ho[illegible]ns1.[illegible]le.[illegible]/do[illegible]n:hostObj>
S:     <domain:hos[illegible]>ns2.[illegible]le[illegible]/do[illegible]n:hostObj>
S:    </domain:ns[illegible]
S:    <domain:ho[illegible]exam[illegible]e.[illegible]/d[illegible]in[illegible]st>
S:    <domain:h[illegible]>ns2[illegible]am[illegible]e.[illegible]/d[illegible]in[illegible]
S:    <domain:clID>ClientX</domain:clID>
S:    <domain:crID>ClientY</domain:crID>
S:    <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:    <domain:upID>ClientX</domain:upID>
S:    <domain:upDate>1999-12-03T09:00:00.0Z</domain:upDate>
S:    <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:    <domain:trDate>2000-04-08T09:00:00.0Z</domain:trDate>
S:    <domain:authInfo>
S:     <domain:pw>2fooBAR</domain:pw>
S:    </domain:authInfo>
S:   </domain:infData>
S:  </resData>
S:  <extension>
S:   <e164:infData xmlns:e164="urn:ietf:params:xml:ns:e164epp-1.0"
S:    xsi:schemaLocation="urn:ietf:params:xml:ns:e164epp-1.0
S:    e164epp-1.0.xsd">
S:    <e164:naptr>
S:     <e164:order>10</e164:order>
S:     <e164:pref>100</e164:pref>
S:     <e164:flags>u</e164:flags>
```

XSD

```
                 type="idmef:file-permission"
                 use="required" />
</xsd:complexType>

<xsd:complexType name="FileAccess">
  <xsd:sequence>
    <xsd:element name="UserId"
                 type="idmef:UserId" />
    <xsd:element name="permission"
                 type="idmef:Permission"
                 minOccurs="1"
                 maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Inode">
  <xsd:sequence>
    <xsd:element name="change-time"
                 type="xsd:string"
                 minOccurs="0"
                 [illegible]Occ[illegible]" />
    <xsd:s[illegible]en[illegible]minO[illegible]rs=[illegible]max[illegible]curs=[illegible]>
      <xsd:[illegible]t na[illegible]"number"
                   typ[illegible]tring" />
      <xsd:[illegible]t name="maj[illegible]evi[illegible]"
                   ty[illegible]"xsd:[illegible]ing" />
      <x[illegible]leme[illegible]name[illegible]devi[illegible]
                   type="xsd:string" />
    </xsd:sequence>
    <xsd:sequence minOccurs="0" maxOccurs="1">
      <xsd:element name="c-major-device"
                   type="xsd:string" />
      <xsd:element name="c-minor-device"
                   type="xsd:string" />
    </xsd:sequence>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Linkage">
  <xsd:choice>
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="path" type="xsd:string" />
    </xsd:sequence>
    <xsd:element name="File" type="idmef:File" />
  </xsd:choice>
  <xsd:attribute name="category"
                 type="idmef:linkage-category"
```

JSON data model

Primitive:

- null
- false, true
- numbers (decimal float)
- text string (UTF-8)

Container:

- “object” (map, with text string keys only)
- array

CBOR data model

Primitive:

- null (+ other "**simple**")
- false, true
- numbers:
  - **Integer**
  - **Float16, 32, 64**
- text string (UTF-8)
- **byte string**

Container:

- map (**any key**)
- array
- **Tag** (extension point)

JSON limitations

- No **binary** data (byte strings)
- Numbers are in **decimal**, some parsing required
- Format requires copying:
  - **Escaping** for strings
  - Base64 for binary
- **No extensibility** (e.g., date format?)
- Interoperability **issues**
  - I-JSON further reduces functionality (RFC 7493)

Data Formats

	Character-based	Concise Binary
Document-Oriented	XML	EXI
Data-Oriented	JSON	???

BSON and friends

- Lots of “binary JSON” proposals
- Often optimized for data at rest, not protocol use (BSON ➔ MongoDB)
- Most are **more** complex than JSON

Why a new binary object format?

- Different design goals from current formats
  – stated up front in the document
- Extremely **small code size**
  – for work on constrained node networks
- Reasonably **compact data size**
  – but no compression or even bit-fiddling
- Useful to any protocol or application that *likes* the design goals

Concise Binary Object Representation (CBOR)

“Sea Boar”

Graphics: Stefanie Gerdes

Data Formats

	Character-based	Concise Binary
Document-Oriented	XML	EXI
Data-Oriented	JSON	CBOR

Design goals (1 of 2)

1. unambiguously encode most **common data formats** (such as JSON-like data) used in Internet standards

2. **compact implementation** possible for encoder and decoder

3. able to parse **without a schema description**.

Design goals (2 of 2)

4. Serialization reasonably **compact**, but data compactness **secondary to** implementation compactness
5. applicable to both **constrained nodes** and **high-volume applications**
6. support all **JSON** data types, conversion to and from JSON
7. **extensible**, with the extended data being able to be parsed by earlier parsers

2013-09-13: CBOR RFC

- “Concise Binary Object Representation”:
  JSON equivalent for constrained nodes
    - start from JSON data model (no schema needed)
    - add binary data, extensibility (“tags”)
    - concise binary encoding (byte-oriented, counting objects)
    - add diagnostic notation
- Done without a WG (with APPSAWG support)

Overview | Specification | Implementations | Tools

http://cbor.io

CBOR

RFC 7049 Concise Binary Object Representation

"The Concise Binary Object Representation (CBOR) 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."

JSON data model

CBOR is based on the wildly successful JSON data model: numbers, strings, arrays, maps (called objects in JSON), and a few values such as false, true, and null.

No Schema needed

Embracing binary

Some applications that would like to use JSON need to transport binary data, such as encryption keys, graphic data, or sensor values. In JSON, these data need to be encoded (usually in base64 format), adding complexity and bulk.

Concise encoding

Stable format

CBOR is defined in an Internet Standards Document, RFC 7049. The format has been designed to be stable for decades.

Extensible

To be able grow with its applications and to

References to rfc7049

This is an experimental product. These dependencies are extracted using heuristics looking for strings with particular prefixes. Notably, this means that references to I-Ds by title only are not reflected here. If it's really important, please inspect the documents' references sections directly.

Showing RFCs and active Internet-Drafts, sorted by reference type, then document name.

Document	Title
draft-bormann-cbor-tags-oid	Concise Binary Object Representation (CBOR) Tags and Techniq Refs Ref'd by
draft-bormann-cbor-time-tag	Concise Binary Object Representation (CBOR) Tags for Time, Du Refs Ref'd by
draft-bormann-lpwan-cbor-template	Concise Binary Object Representation (CBOR) Tag for CBOR Ten Refs Ref'd by
draft-bormann-t2trg-sworn	SWORN: Secure Wake on Radio Nudging Refs Ref'd by
draft-carpenter-anima-asa-guidelines	Guidelines for Autonomic Service Agents Refs Ref'd by
draft-hartke-t2trg-cbor-forms	CBOR-encoded Form Data Refs Ref'd by
draft-hartke-t2trg-coral	The Constrained RESTful Application Language (CoRAL) Refs Ref'd by
draft-ietf-6tisch-minimal-security	Minimal Security Framework for 6TiSCH Refs Ref'd by
draft-ietf-ace-cbor-web-token	CBOR Web Token (CWT) Refs Ref'd by
draft-ietf-anima-grasp	A Generic Autonomic Signaling Protocol (GRASP) Refs Ref'd by
draft-ietf-core-links-json	Representing CoRE Formats in JSON and CBOR Refs Ref'd by
draft-ietf-core-object-security	Object Security of CoAP (OSCOAP) Refs Ref'd by
draft-ietf-core-senml	Media Types for Sensor Measurement Lists (SenML) Refs Ref'd by
draft-ietf-core-sid	YANG Schema Item iDentifier (SID) Refs Ref'd by
draft-ietf-core-yang-cbor	CBOR Encoding of Data Modeled with YANG Refs Ref'd by
draft-ietf-cose-msg	CBOR Object Signing and Encryption (COSE) Refs Ref'd by
draft-ietf-dtn-bpbis	Bundle Protocol Refs Ref'd by
draft-ietf-roll-mpl-forw-select	MPL Forwarder Select (MPLFS) Refs Ref'd by
draft-ietf-sacm-coswid	Concise Software Identifiers Refs Ref'd by
draft-jones-cose-rsa	Using RSA Algorithms with COSE Messages Refs Ref'd by
draft-jroatch-cbor-tags	Concise Binary Object Representation (CBOR) Tags for Typed Ar Refs Ref'd by
draft-keranen-t2trg-rest-iot	RESTful Design for Internet of Things Systems Refs Ref'd by
draft-navas-ace-secure-time-synchronization	Lightweight Authenticated Time (LATe) Synchronization Protoc Refs Ref'd by
draft-richardson-6tisch-minimal-rekey	Minimal Security rekeying mechanism for 6TiSCH Refs Ref'd by
draft-schaad-cose-x509	CBOR Encoded Message Syntax (COSE): Headers for carrying ar Refs Ref'd by
draft-selander-ace-cose-ecdhe	Ephemeral Diffie-Hellman Over COSE (EDHOC) Refs Ref'd by

References to rfc7049

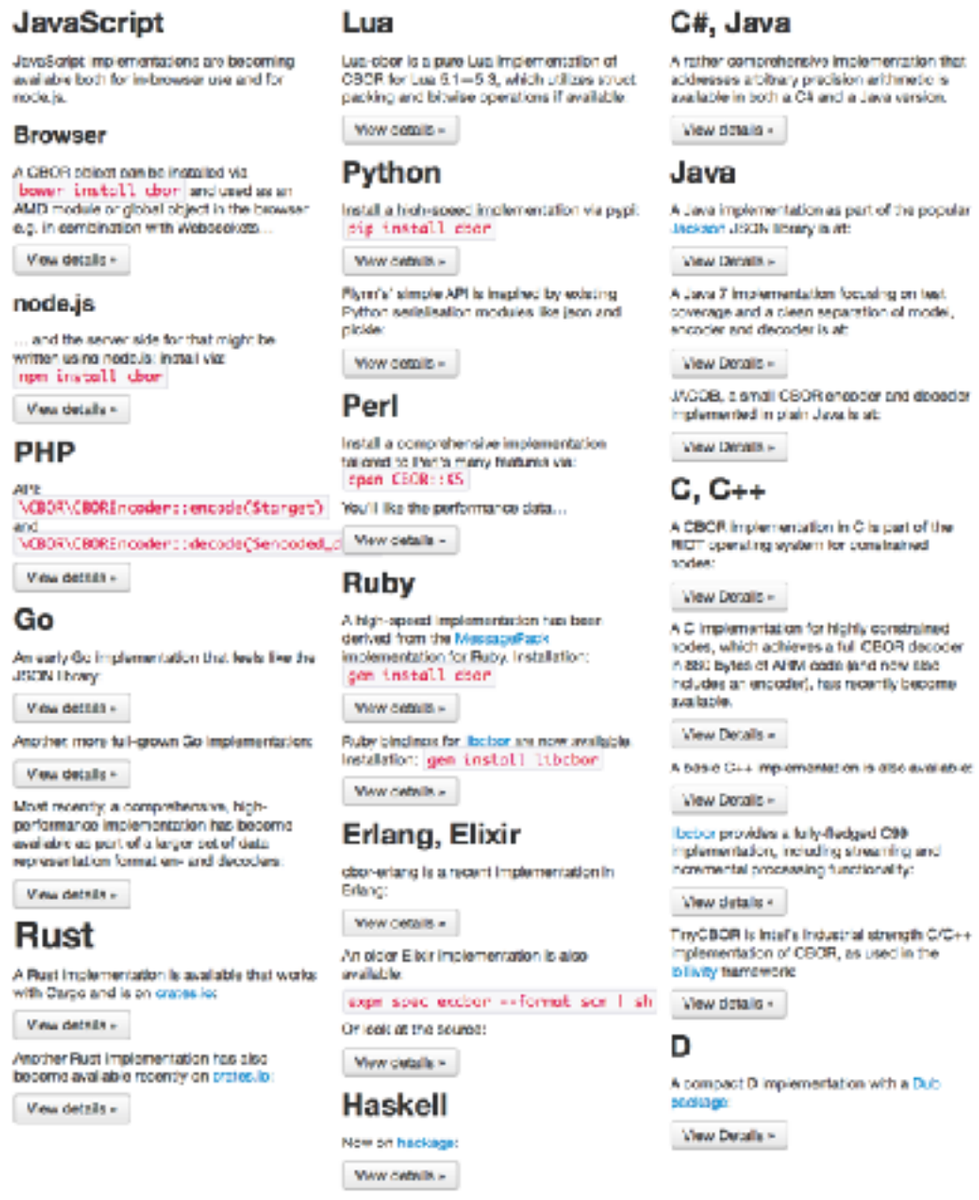
This is an experimental product. These dependencies are extracted using heuristics looking for strings with particular prefixes. Notably, this means that references to I-D

Showing RFCs and active Internet-Drafts, sorted by reference type, then document name.

Document	Title
draft-bormann-cbor-tags-oid	Concise Binary Object Representation (CBOR) Tags and Techniques for Object Identifiers, UUIDs, Enumerati Refs Ref'd by
draft-bormann-cbor-time-tag	Concise Binary Object Representation (CBOR) Tags for Time, Duration, and Period Refs Ref'd by
draft-bormann-lpwan-cbor-template	Concise Binary Object Representation (CBOR) Tag for CBOR Templates Refs Ref'd by
draft-bormann-t2trg-sworn	SWORN: Secure Wake on Radio Nudging Refs Ref'd by
draft-carpenter-anima-asa-guidelines	Guidelines for Autonomic Service Agents Refs Ref'd by
draft-hartke-t2trg-cbor-forms	CBOR-encoded Form Data Refs Ref'd by
draft-hartke-t2trg-coral	The Constrained RESTful Application Language (CoRAL) Refs Ref'd by
draft-ietf-6tisch-minimal-security	Minimal Security Framework for 6TiSCH Refs Ref'd by
draft-ietf-ace-cbor-web-token	CBOR Web Token (CWT) Refs Ref'd by
draft-ietf-anima-grasp	A Generic Autonomic Signaling Protocol (GRASP) Refs Ref'd by
draft-ietf-core-links-json	Representing CoRE Formats in JSON and CBOR Refs Ref'd by
draft-ietf-core-object-security	Object Security of CoAP (OSCOAP) Refs Ref'd by
draft-ietf-core-senml	Media Types for Sensor Measurement Lists (SenML) Refs Ref'd by
draft-ietf-core-sid	YANG Schema Item iDentifier (SID) Refs Ref'd by
draft-ietf-core-yang-cbor	CBOR Encoding of Data Modeled with YANG Refs Ref'd by
draft-ietf-cose-msg	CBOR Object Signing and Encryption (COSE) Refs Ref'd by
draft-ietf-dtn-bpbis	Bundle Protocol Refs Ref'd by
draft-ietf-roll-mpl-forw-select	MPL Forwarder Select (MPLFS) Refs Ref'd by
draft-ietf-sacm-coswid	Concise Software Identifiers Refs Ref'd by
draft-jones-cose-rsa	Using RSA Algorithms with COSE Messages Refs Ref'd by
draft-jroatch-cbor-tags	Concise Binary Object Representation (CBOR) Tags for Typed Arrays Refs Ref'd by
draft-keranen-t2trg-rest-iot	RESTful Design for Internet of Things Systems Refs Ref'd by
draft-navas-ace-secure-time-synchronization	Lightweight Authenticated Time (LATe) Synchronization Protocol Refs Ref'd by

Implementations

- Parsing/generating CBOR easier than interfacing with application
  - Minimal implementation: 822 bytes of ARM code
- Different integration models, different languages
- > 25 implementations (after first two years)



http://cbor.io

Batteries included

- RFC 7049 predefines 18 Tags
  - Time, big numbers (bigint, float, decimal), various converter helpers, URI, MIME message
- Easy to register your own CBOR Tags
  - 19 more tags: 6 for COSE; UUIDs, binary MIME, Perl support, language tagged string, compression

2015-06-03: COSE WG

- CBOR Object Signing and Encryption:
  Object Security for the IoT

- Based on **JOSE**: JSON Web Token, JWS, JWE, …
  - Data structures for signatures, integrity, encryption…
  - Derived from on OAuth JWT
  - Encoded in JSON, can encrypt/sign other data

- **COSE: use CBOR instead of JSON**
  - Can directly use binary encoding (no base64)
  - Optimized for constrained devices

So, why a WG?

Take CBOR to STD

RFC 6410:

- independent interoperable implementations ✔
- no errata (oops)
- no unused features
- (if patented: licensing process)

Take CBOR to STD

- **Do not**: futz around
- Document interoperability
- Make needed improvements in specification quality
  - At least fix the errata :-)
- Are all tags implemented interoperably?

Next steps

- Create a 7049bis repo on github.com/cbor-wg
  - Leading to draft-ietf-cbor-7049bis shortly
- Start the git-based issues/PR/merge process
- Start a separate feature interoperability list (wiki?)

CDDL

Henk Birkholz, Christoph Vigano, Carsten Bormann

FDT in the IETF

- Formal description techniques helped kill OSI
- Takeup of FDT in IETF reluctant
  - A few notable exceptions: e.g. RFC 4997
  - Island of FDT: Management — SMIv2, YANG
- Widely used: ABNF
  (RFC 5234 = STD 68, updated by RFC 7405 (PS))

ABNF

- BNF: grammars for strings
  - RFC40 (1970): first RFC with BNF
- “Internet” BNF: Augmented BNF (ABNF)
  - RFC 733 (1977): “Ken L. Harrenstien, of SRI International, was responsible for re-coding the BNF into an augmented BNF which compacts the specification and allows increased comprehensibility.”

ABNF in the IETF

- 752 RFCs and I-Ds reference RFC 5234 (the most recent version of ABNF) [cf. YANG: 160]

- Tool support (e.g., BAP, abnf-gen; antlr support)

- Pretty much standard for text-based protocols that aren’t based on XML or JSON

ABNF is composed of productions

```
addr-spec      = local-part "@" domain
local-part     = dot-atom / quoted-string / obs-local-part
domain         = dot-atom / domain-literal / obs-domain
domain-literal = [CFWS] "[" *([FWS] dtext) [FWS] "]" [CFWS]
dtext          = %d33-90 /          ; Printable US-ASCII
                 %d94-126 /         ;  characters not including
                 obs-dtext          ;  "[", "]", or “\"
```

- **Names** for sublanguages
- **Compose** using
  - Concatenation
  - Choice: /
- **Literals** terminate nesting

From ABNF to CDDL

- Build **trees** of data items, not **strings** of characters
- Add literals for primitive types
- Add constructors for containers (arrays, maps)
  - Inspiration: Relax-NG (ISO/IEC 19757-2)

Rule names are **types**

```
bool = false / true
label = text / int
int = uint / nint
```

- Types are **sets** of potential values
- Even literals are (very small) types

```
participants = 1 / 2 / 3
participants = 1..3
msgtype = "PUT"
msgtype = 1
```

Groups: building containers

- Containers contain sequences (array) or sets (maps) of entries

- Entries are types (array) or key/value type pairs (maps)

- Unify this into **group**:

  - sequenced (ignored within maps)

  - labeled (ignored within arrays)

How **RFC 7071** would have looked like in CDDL

```
reputation-object = {              ; This is a map (JSON object)
  application: text                ;  text string (vs. binary)
  reputons: [* reputon]            ;  Array of 0-∞ reputons
}

reputon = {                        ; Another map (JSON object)
  rater: text
  assertion: text
  rated: text
  rating: float16                  ; OK, float16 is a CBORism
  ? confidence: float16            ; optional…
  ? normal-rating: float16
  ? sample-size: uint              ; unsigned integer
  ? generated: uint
  ? expires: uint
  * text => any                    ; 0-∞, express extensibility
}
```

Named groups

```
header_map = {
    Generic_Headers,
    * label => values
}
Generic_Headers = (
    ? 1 => int / tstr,  ; algorithm identifier
    ? 2 => [+label],    ; criticality
    ? 3 => tstr / int,  ; content type
    ? 4 => bstr,        ; key identifier
    ? 5 => bstr,        ; IV
    ? 6 => bstr,        ; Partial IV
    ? 7 => COSE_Signature / [+COSE_Signature]
)
```

- Named groups allow re-use of parts of a map/array
- Inclusion instead of inheritance

GRASP

- Generic Autonomic Signaling Protocol (GRASP)
- For once, try not to invent another TLV format: just use CBOR
- Messages are arrays, with type, id, option:

```
message /= [MESSAGE_TYPE, session-id, *option]
MESSAGE_TYPE = 123 ; a defined constant
session-id = 0..16777215
; option is one of the options defined below
```

- Options are arrays, again:

```
option /= waiting-time-option
waiting-time-option = [O_WAITING, waiting-time]
O_WAITING = 456 ; a defined constant
waiting-time = 0..4294967295  ; in milliseconds
```

References to draft-greevenbosch-appsawg-cbor-cddl

This is an experimental product. These dependencies are extracted using heuristics looking for strings with particular prefixes. Notably, this means that references to I-Ds by title only are not reflected here. If it's really important, please inspect the documents' refere

Showing RFCs and active Internet-Drafts, sorted by reference type, then document name.

Document	Title	Status	Type
draft-bormann-cbor-time-tag	**Concise Binary Object Representation (CBOR) Tags for Time, Duration, and Period** Refs Ref'd by		normatively references
draft-bormann-lpwan-cbor-template	**Concise Binary Object Representation (CBOR) Tag for CBOR Templates** Refs Ref'd by		normatively references
draft-bormann-t2trg-sworn	**SWORN: Secure Wake on Radio Nudging** Refs Ref'd by		normatively references
draft-carpenter-anima-ani-objectives	**Technical Objective Formats for the Autonomic Network Infrastructure** Refs Ref'd by		normatively references
draft-jroatch-cbor-tags	**Concise Binary Object Representation (CBOR) Tags for Typed Arrays** Refs Ref'd by		normatively references
draft-mandyam-tokbind-attest	**Attested TLS Token Binding** Refs Ref'd by		normatively references
draft-birkholz-tuda	**Time-Based Uni-Directional Attestation** Refs Ref'd by		informatively references
draft-hartke-t2trg-cbor-forms	**CBOR-encoded Form Data** Refs Ref'd by		informatively references
draft-hartke-t2trg-coral	**The Constrained RESTful Application Language (CoRAL)** Refs Ref'd by		informatively references
draft-ietf-ace-cbor-web-token	**CBOR Web Token (CWT)** Refs Ref'd by		informatively references
draft-ietf-anima-grasp	**A Generic Autonomic Signaling Protocol (GRASP)** Refs Ref'd by	Proposed Standard	informatively references
draft-ietf-anima-prefix-management	**Autonomic IPv6 Edge Prefix Management in Large-scale Networks** Refs Ref'd by	Informational	informatively references
draft-ietf-core-links-json	**Representing CoRE Formats in JSON and CBOR** Refs Ref'd by	Proposed Standard	informatively references
draft-ietf-core-object-security	**Object Security of CoAP (OSCOAP)** Refs Ref'd by		informatively references
draft-ietf-core-senml	**Media Types for Sensor Measurement Lists (SenML)** Refs Ref'd by	Proposed Standard	informatively references
draft-ietf-cose-msg	**CBOR Object Signing and Encryption (COSE)** Refs Ref'd by	Proposed Standard	informatively references
draft-ietf-dnsop-dns-capture-format	**C-DNS: A DNS Packet Capture Format** Refs Ref'd by	Proposed Standard	informatively references
draft-ietf-sacm-coswid	**Concise Software Identifiers** Refs Ref'd by		informatively references
draft-liu-opentrustprotocol-cbor	**Open Trust Protocol CBOR Encoding** Refs Ref'd by		informatively references
draft-schaad-cose-x509	**CBOR Encoded Message Syntax (COSE): Headers for carrying and referencing X.509 certificates** Refs Ref'd by		informatively references
draft-selander-ace-cose-ecdhe	**Ephemeral Diffie-Hellman Over COSE (EDHOC)** Refs Ref'd by		informatively references
rfc8007	**Content Delivery Network Interconnection (CDNI) Control Interface / Triggers** Refs Ref'd by	Proposed Standard	informatively references

SDOs outside of IETF

- CDDL is being used for specifying both CBOR and JSON in W3C, ___, and _____ ___
- Data in flight in a variety of protocols, e.g.
  - Access to specific features in wireless radios
  - Aggregation of metadata,
    enabling visualization of network topologies

From draft to RFC

- **Do not**: break it
- Editorial improvements required
- Any additional language features needed?
  - Should stay in the “tree grammar” envelope
- What can we take out?

computed literals?

- integers relative to an offset

```
base = 400
a = base + 1
b = base + 2
```

- string concatenation/interpolation
    - e.g., to build long regexes out of parts

unpack/inclusion operator?

```
foo-basic = { foo-guts }
foo-guts = (a: int, b: uint)
foo-extended = { foo-guts, c: text }
```

- →

```
foo-basic = { a: int, b: uint }
foo-extended = { <foo-basic, c: text }
```

representation constraints

- definite vs. indefinite
- Float16, float32, float64
- …
- (These often can be done on a global level)

cuts (better error messages)

```
a = ant / cat / elk
ant = ["ant", ^ uint]
cat = ["cat", ^ text]
ant = ["elk", ^ float]

["ant", 47.11]
```

- tool will not tell you "can't match a",
  but "can't match rest of ant"

modules

```
;;< module fritz
;;< export foo, bar
foo = [baz, ant, cat]
bar = uint
```

```
;;< module animals
;;< from fritz import foo
```

- (This is completely unthought-through)
- Proposal: make these a layer on top of CDDL

interchange as JSON

```
a = b / c
```

- →

```
[":rule", "a", [":typechoice", "b", "c"]]
```

- Define standard mapping for tools that want to
    - pretty print CDDL
    - reason about CDDL
    - transform CDDL (e.g., for parser generators)

Avoid the kitchen sink

- This is not a Christmas wish list
- Each feature has a cost
  - specification complexity
  - learning effort
  - implementation effort

Next steps

- cddl draft already at github.com/core-wg

- Start the git-based issues/PR/merge process

More tags

draft-jroatch-cbor-tags-05

- Provide tags for homogeneous arrays represented in byte strings
  - Inspired by JavaScript
  - Both LSB and MSB first
  - Reserves 24 tags in 1-byte space

uint	sint	float
uint8	sint8	binary16
uint16	sint16	binary32
uint32	sint32	binary64
uint64	sint64	binary128

- Provide a tag for other homogeneous arrays
- Provide a tag for multidimensional arrays

Unchartered Work

draft-bormann-cbor-time-tag-00

- Nobody knew that time could be so complicated!

draft-bormann-cbor-time-tag-00

- Limits of CBOR Tag 0/1:
  - Limited resolution
  - Only Posix Time as time scale
  - “Intent” information and other metadata cannot be included
- Start with defining a kitchen sink
  - Then see whether we want to keep all of that
  - Make sure simple things stay simple

draft-bormann-lpwan-cbor-template

- **variable:** placeholder CBOR data item included in a larger data item (the "CBOR template")
- Relevant for LPWAN SCHC
- But can be used in a general way

Status of Tags drafts

- OID: On charter, kitchen sink
- Array: On charter, ready for adoption
- Time: Off charter
- Template: Off charter
  (will likely be done with SCHC anyway)

Tutorial

CBOR: Agenda

- What is it, and when might I want it?
- How does it work?
- How do I work with it?

CBOR vs. “binary JSONs”

- Encoding [1, [2, 3]]:

	compact	stream
ASN.1 BER*	30 0b 02 01 01 30 06 02 01 02 02 01 03	30 80 02 01 01 30 06 02 01 02 02 01 03 00 00
MessagePack	92 01 92 02 03	
BSON	22 00 00 00 10 30 00 01 00 00 00 04 31 00 13 00 00 00 10 30 00 02 00 00 00 10 31 00 03 00 00 00 00 00	
UBJSON	61 02 42 01 61 02 42 02 42 03	61 ff 42 01 61 02 42 02 42 03 45*
CBOR	82 01 82 02 03	9f 01 82 02 03 ff

Very quick overview of the format

- Initial byte: **major type** (3 bits) and **additional information** (5 bits: immediate value or length information)
- Eight major types:
  - unsigned (0) and negative (1) **integers**
  - **byte** strings (2), **UTF-8** strings (3)
  - **arrays** (4), **maps** (5)
  - optional **tagging** (6) and **simple types** (7) (floating point, Booleans, etc.)

Additional information

- 5 bits
  - 0..23: immediate value
  - 24..27: 1, 2, 4, 8 bytes value follow
  - *28..30: reserved*
  - 31: indefinite length
    - terminated only by 0xFF in place of data item

- Generates unsigned integer:
  - **Value** for mt 0, 1 (unsigned/neg integers), 7 (“simple”)
  - **Length** (in bytes) for mt 2, 3 (byte/text strings)
  - **Count** (in items) for mt 4, 5 (array, map)
  - **Tag** value for mt 6

Major types 6 and 7

- mt 7:
  - **special** values for ai = 0..24
    - false, true, null, undef
    - IANA registry for more
  - ai = 25, 26, 27: IEEE **floats**
    - in 16 (“half”), 32 (“single”), and 64 (“double”) bits
- mt 6: semantic **tagging** for things like dates, arbitrary-length bignums, and decimal fractions

Tags

- A Tag contains **one** data item
- 0: RFC 3339 (~ ISO 8601) **text string** date/time
- 1: UNIX time (**number** relative to 1970-01-01)
- 2/3: bignum (**byte string** encodes unsigned)
- 4: **[exp, mant]** (decimal fraction)
- 5: **[exp, mant]** (binary fraction, “bigfloat”)
- 21..23: expected conversion of **byte string**
- 24: nested CBOR data item in **byte string**
- 32…: URI, base64[url], regexp, mime (**text strings**)

New Tags

- Anyone can register a tag (IANA)
  - 0..23: Standards action
  - 24..255: Specification required
  - 256..18446744073709551615: FCFS
- 25/256: stringref for simple compression
- 28/29: value sharing (beyond trees)
- 26/27: constructed object (Perl/generic)
- 22098: Perl reference (“indirection”)

Examples

- Lots of examples in RFC (making use of JSON–like “**diagnostic notation**”)
- 0 ➔ 0x00, 1 ➔ 0x01, 23 ➔ 0x17, 24 ➔ 0x1818
- 100 ➔ 0x1864, 1000 ➔ 0x1903e8, 1000000 ➔ 0x1a000f4240
- 18446744073709551615 ➔ 0x1bffffffffffffff, 18446744073709551616 ➔ 0xc249010000000000000000
- –1 ➔ 0x20, –10 ➔ 0x29, –100 ➔ 0x3863, –1000 ➔ 0x3903e7
- 1.0 ➔ 0xf93c00, 1.1 ➔ 0xfb3ff199999999999a, 1.5 ➔ 0xf93e00
- Infinity ➔ 0xf97c00, NaN ➔ 0xf97e00, –Infinity ➔ 0xf9fc00
- false ➔ 0xf4, true ➔ 0xf5, null ➔ 0xf6
- h'' ➔ 0x40, h'01020304' ➔ 0x4401020304
- "" ➔ 0x60, ”a" ➔ 0x6161, ”IETF" ➔ 0x6449455446
- [] ➔ 0x80, [1, 2, 3] ➔ 0x83010203, [1, [2, 3], [4, 5]] ➔ 0x830182020382040 5
- {} ➔ 0xa0, {1: 2, 3: 4} ➔ 0xa201020304, {"a": 1, "b": [2, 3]} ➔ 0xa26161016162820203

CBOR: Agenda

- What is it, and when might I want it?
- How does it work?
- How do I work with it?

http://cbor.me: CBOR playground

- Convert back and forth between **diagnostic notation** (~JSON) and binary encoding

CBOR

Diagnostic →

```
[1, [2, 3]]
```

← 5 Bytes

```
82        # array(2)
   01     # unsigned(1)
   82     # array(2)
      02  # unsigned(2)
      03  # unsigned(3)
```

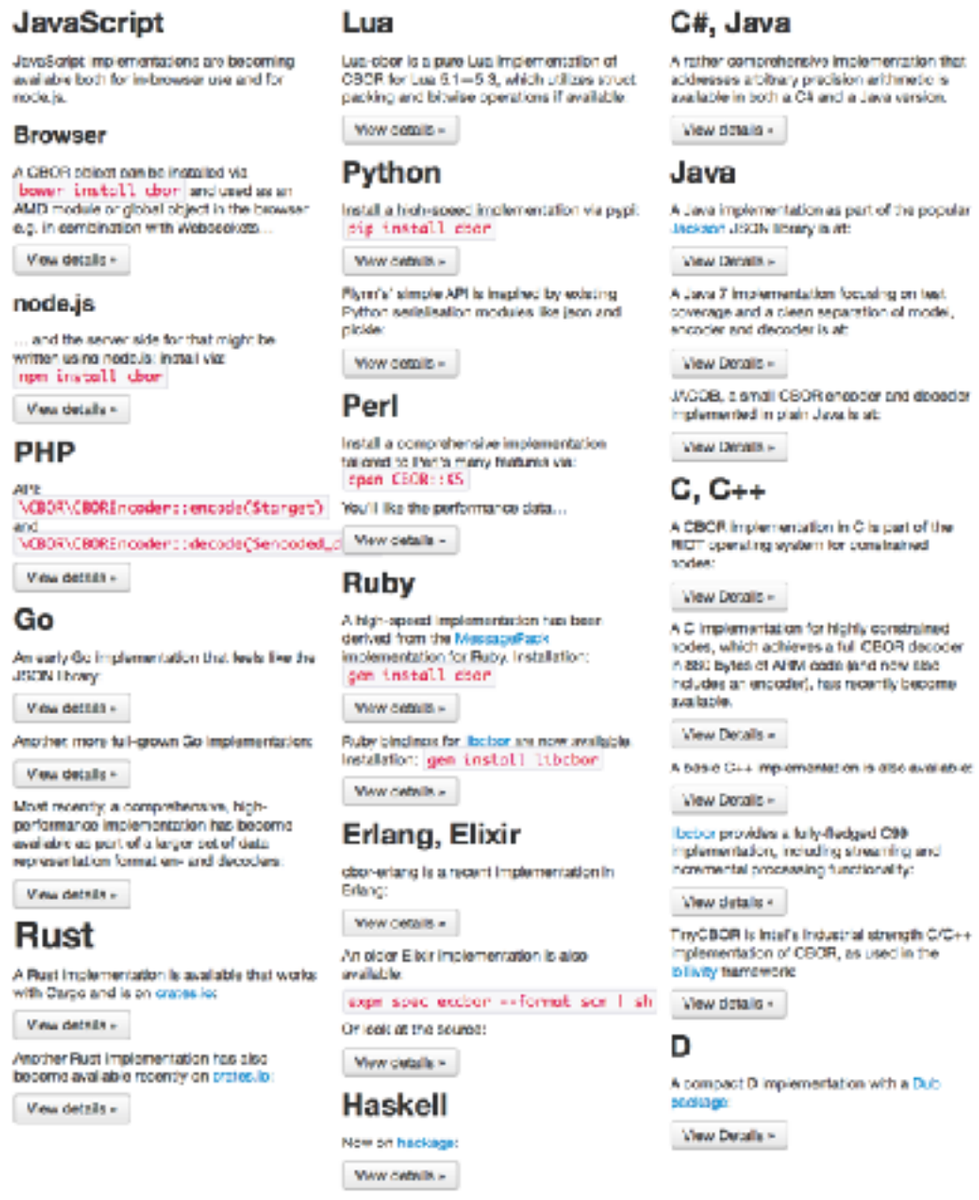
Offline tools (gem install)

- cbor-diag:
  offline (command line) version of cbor.me

- cddl: generate examples from CDDL, verify instances against CDDL, extract code definitions from CDDL

Implementations

- Parsing/generating CBOR easier than interfacing with application
  - Minimal implementation: 822 bytes of ARM code
- Different integration models, different languages
- > 25 implementations (after first two years)



http://cbor.io

Resources

- RFC 7049
- http://cbor.io and http://cbor.me; gem install cbor-diag
- cbor@ietf.org
- http://tools.ietf.org/html/cddl
- gem install cddl