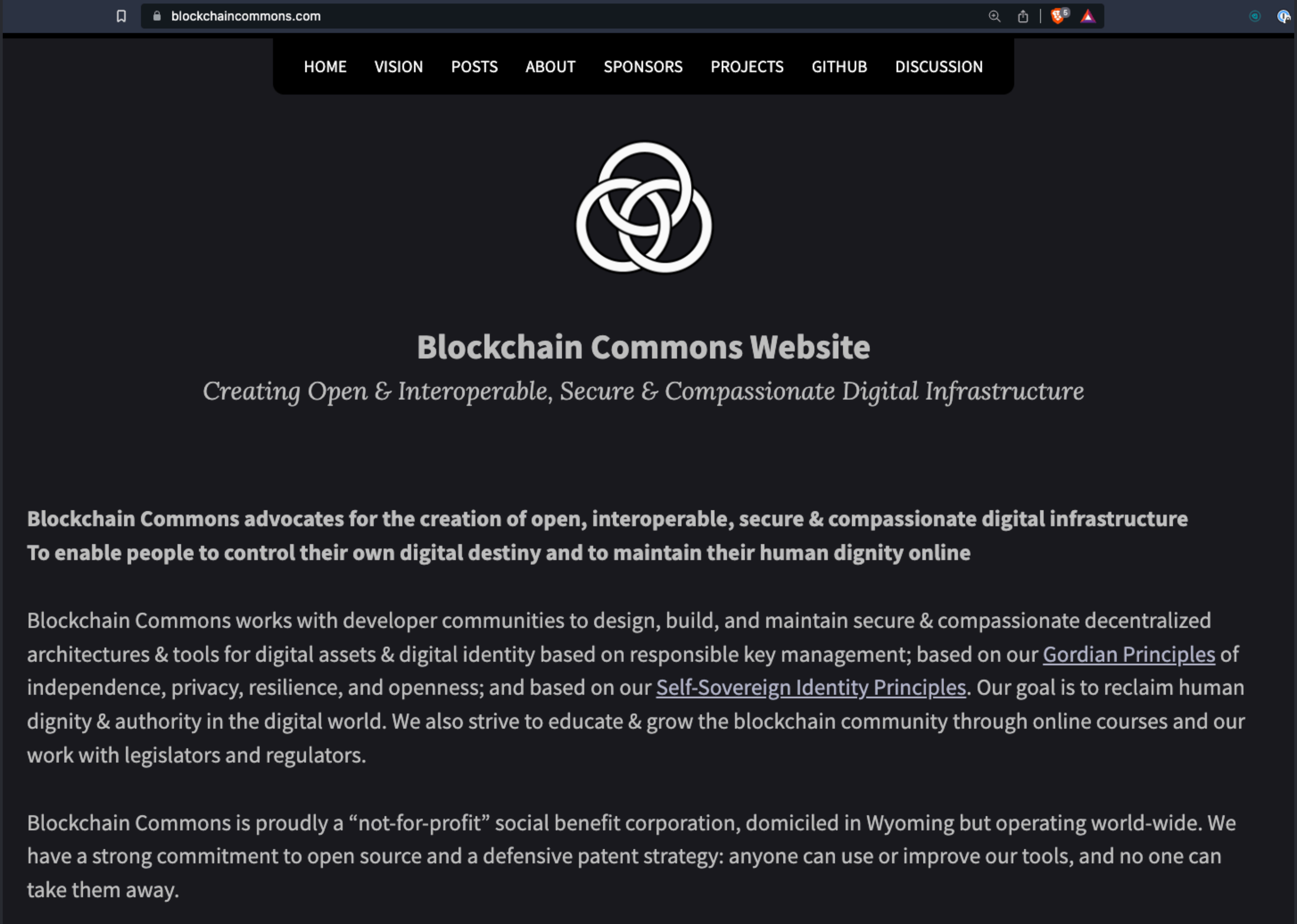
BLOCKCHAIN COMMONS

GORDIAN ENVELOPE & dCBOR IETF DISPATCH 27 MARCH 2023

BLOCKCHAIN COMMONS


blockchaincommons.com

HOME VISION POSTS ABOUT SPONSORS PROJECTS GITHUB DISCUSSION

Blockchain Commons Website

Creating Open & Interoperable, Secure & Compassionate Digital Infrastructure

Blockchain Commons advocates for the creation of open, interoperable, secure & compassionate digital infrastructure
To enable people to control their own digital destiny and to maintain their human dignity online

Blockchain Commons works with developer communities to design, build, and maintain secure & compassionate decentralized architectures & tools for digital assets & digital identity based on responsible key management; based on our Gordian Principles of independence, privacy, resilience, and openness; and based on our Self-Sovereign Identity Principles. Our goal is to reclaim human dignity & authority in the digital world. We also strive to educate & grow the blockchain community through online courses and our work with legislators and regulators.

Blockchain Commons is proudly a “not-for-profit” social benefit corporation, domiciled in Wyoming but operating world-wide. We have a strong commitment to open source and a defensive patent strategy: anyone can use or improve our tools, and no one can take them away.

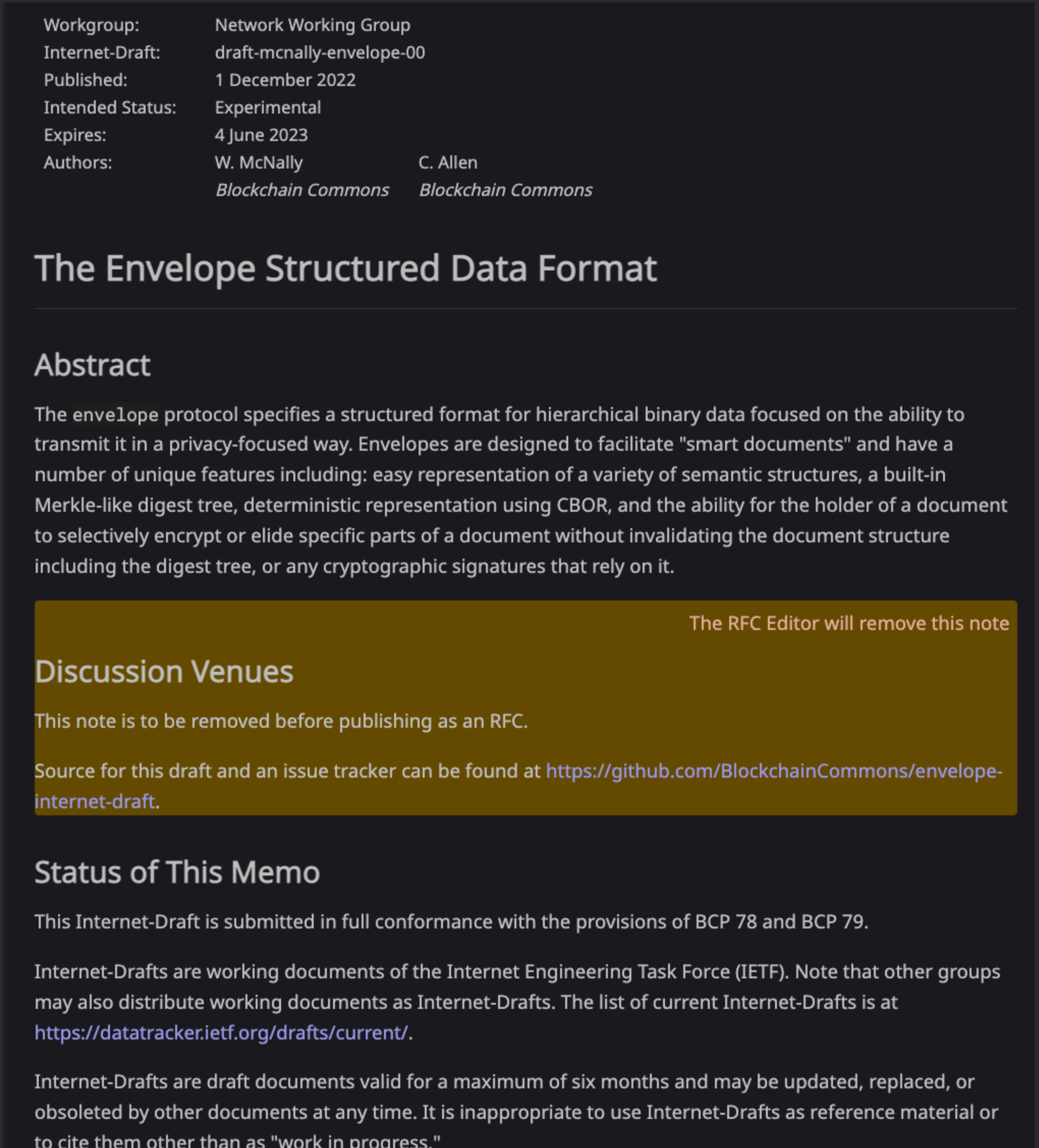
WHAT IS BLOCKCHAIN COMMONS?

- We are a community that brings together stakeholders to collaboratively build open & interoperable, secure & compassionate infrastructure.
- We design decentralized solutions where everyone wins.
- We are a neutral “not-for-profit” that enables people to control their own digital destiny.

GORDIAN ENVELOPE

INTERNET DRAFT

datatracker.ietf.org/doc/draft-mcnally-envelope



Workgroup:	Network Working Group	
Internet-Draft:	draft-mcnally-envelope-00	
Published:	1 December 2022	
Intended Status:	Experimental	
Expires:	4 June 2023	
Authors:	W. McNally	C. Allen
	Blockchain Commons	*Blockchain Commons*

The Envelope Structured Data Format

Abstract

The `envelope` protocol specifies a structured format for hierarchical binary data focused on the ability to transmit it in a privacy-focused way. Envelopes are designed to facilitate "smart documents" and have a number of unique features including: easy representation of a variety of semantic structures, a built-in Merkle-like digest tree, deterministic representation using CBOR, and the ability for the holder of a document to selectively encrypt or elide specific parts of a document without invalidating the document structure including the digest tree, or any cryptographic signatures that rely on it.

The RFC Editor will remove this note

Discussion Venues

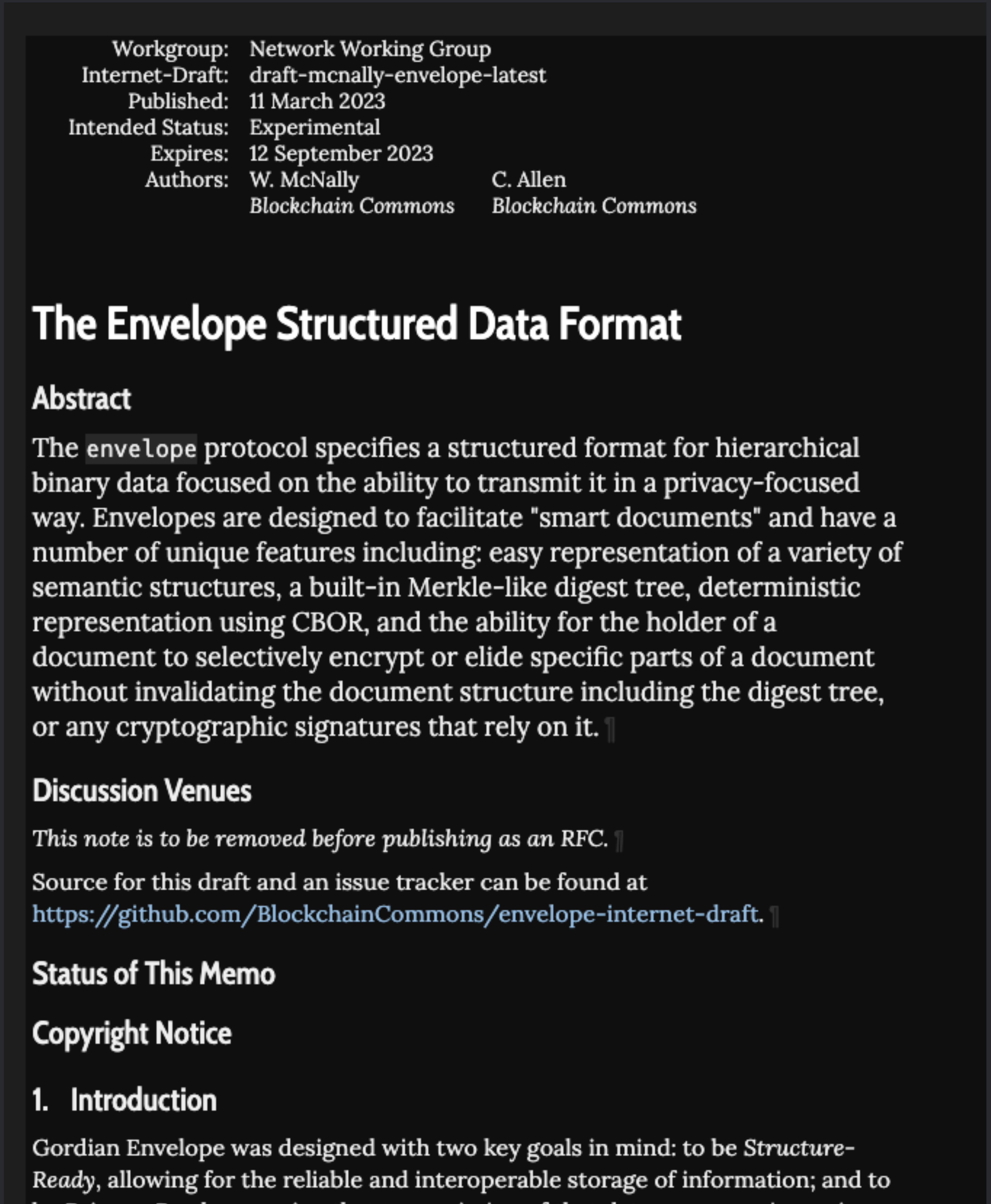
This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/BlockchainCommons/envelope-internet-draft.

Status of This Memo

INTERNET DRAFT

blockchaincommons.github.io/WIPs-IETF-draft-envelope/draft-mcnally-envelope.html



Workgroup:	Network Working Group
Internet-Draft:	draft-mcnally-envelope-latest
Published:	11 March 2023
Intended Status:	Experimental
Expires:	12 September 2023
Authors:	W. McNally *Blockchain Commons* · C. Allen *Blockchain Commons*

The Envelope Structured Data Format

Abstract

The `envelope` protocol specifies a structured format for hierarchical binary data focused on the ability to transmit it in a privacy-focused way. Envelopes are designed to facilitate "smart documents" and have a number of unique features including: easy representation of a variety of semantic structures, a built-in Merkle-like digest tree, deterministic representation using CBOR, and the ability for the holder of a document to selectively encrypt or elide specific parts of a document without invalidating the document structure including the digest tree, or any cryptographic signatures that rely on it. ¶

Discussion Venues

This note is to be removed before publishing as an RFC. ¶

Source for this draft and an issue tracker can be found at https://github.com/BlockchainCommons/envelope-internet-draft. ¶

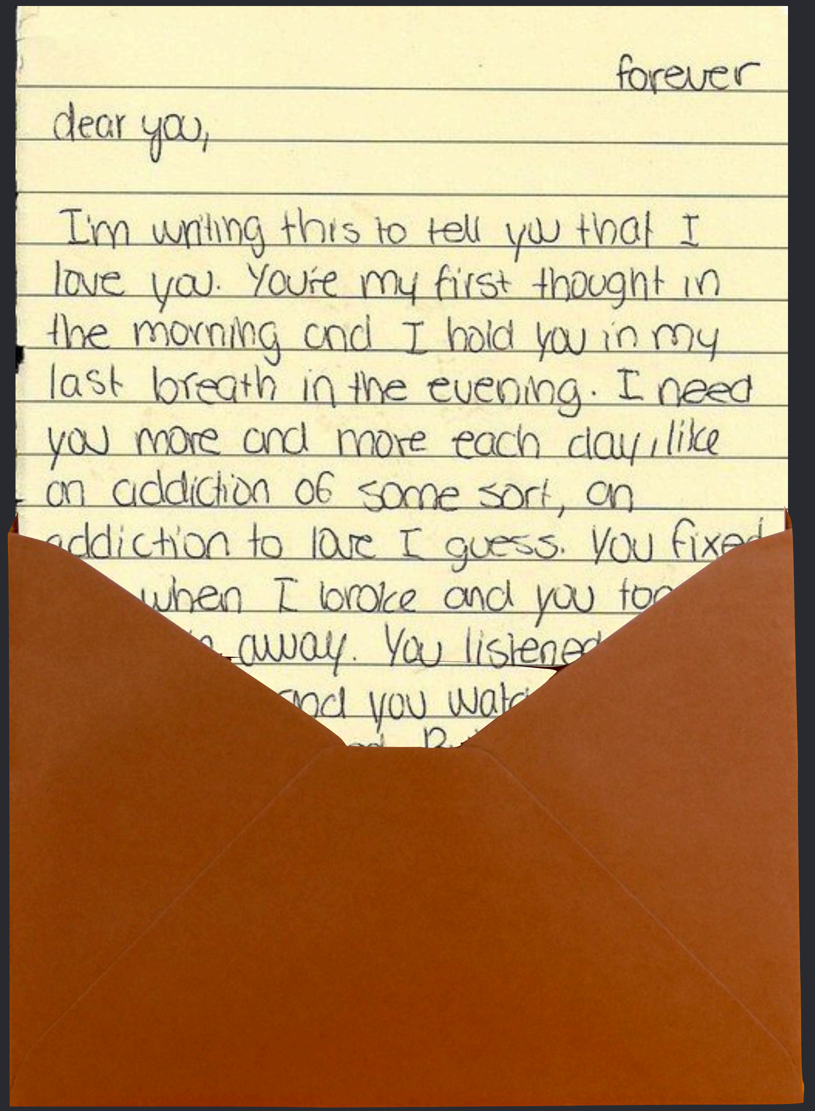
Status of This Memo

Copyright Notice

1. Introduction

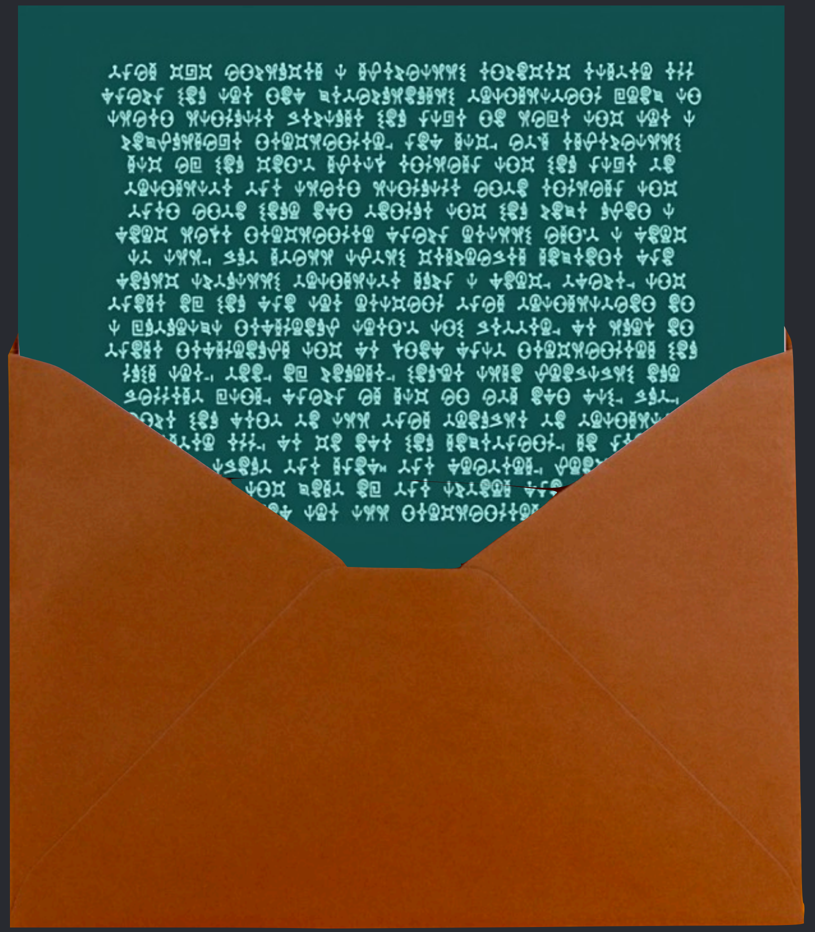
Gordian Envelope was designed with two key goals in mind: to be *Structure-Ready*, allowing for the reliable and interoperable storage of information; and to

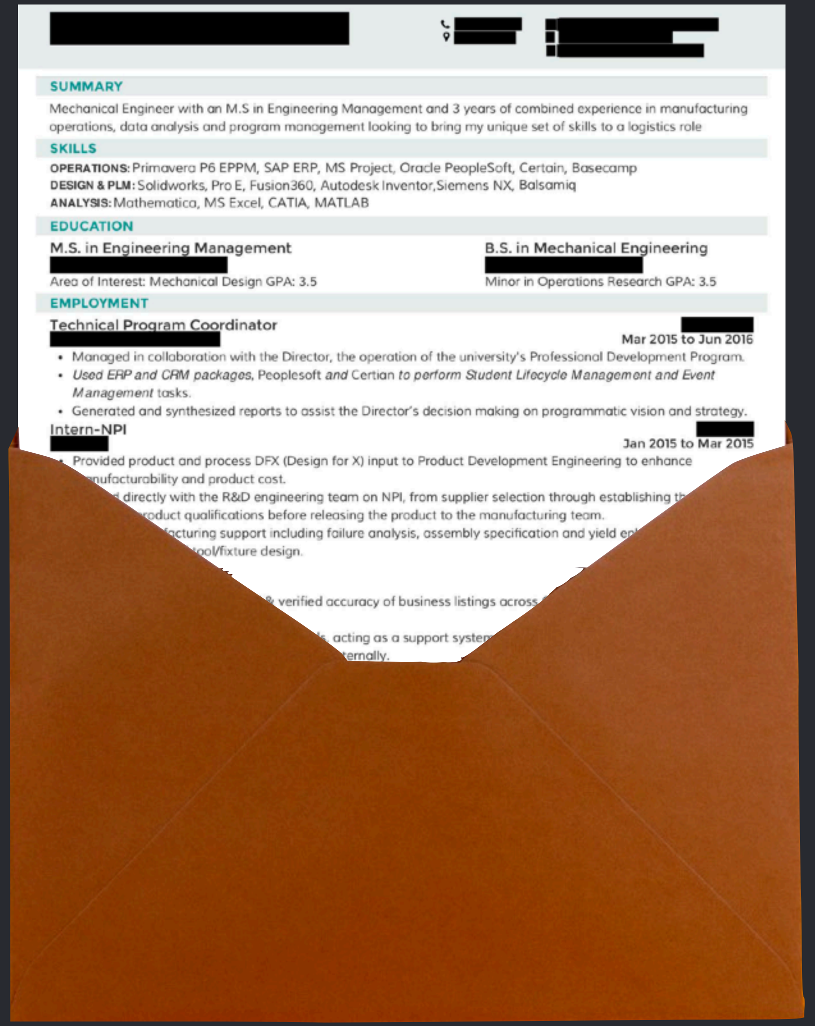
ENVELOPES HOLD MANY KINDS OF THINGS


forever

dear you,

I'm writing this to tell you that I love you. You're my first thought in the morning and I hold you in my last breath in the evening. I need you more and more each day, like an addiction of some sort, an addiction to love I guess. You fixed when I broke and you too away. You listened and you watc




[redacted]

[redacted]

[redacted]

SUMMARY

Mechanical Engineer with an M.S in Engineering Management and 3 years of combined experience in manufacturing operations, data analysis and program management looking to bring my unique set of skills to a logistics role

SKILLS

OPERATIONS: Primavera P6 EPPM, SAP ERP, MS Project, Oracle PeopleSoft, Certain, Basecamp
DESIGN & PLM: Solidworks, Pro E, Fusion360, Autodesk Inventor,Siemens NX, Balsamiq
ANALYSIS: Mathematica, MS Excel, CATIA, MATLAB

EDUCATION

M.S. in Engineering Management
[redacted]
Area of Interest: Mechanical Design GPA: 3.5

B.S. in Mechanical Engineering
[redacted]
Minor in Operations Research GPA: 3.5

EMPLOYMENT

Technical Program Coordinator [redacted]
[redacted] Mar 2015 to Jun 2016

- Managed in collaboration with the Director, the operation of the university's Professional Development Program.
- *Used ERP and CRM packages,* Peoplesoft *and* Certian *to perform Student Lifecycle Management and Event Management* tasks.
- Generated and synthesized reports to assist the Director's decision making on programmatic vision and strategy.

Intern-NPI [redacted]
[redacted] Jan 2015 to Mar 2015

- Provided product and process DFX (Design for X) input to Product Development Engineering to enhance nufacturability and product cost.
- d directly with the R&D engineering team on NPI, from supplier selection through establishing th roduct qualifications before releasing the product to the manufacturing team.
- facturing support including failure analysis, assembly specification and yield en tool/fixture design.

& verified accuracy of business listings across

s, acting as a support system ternally.

CERTIFICATE

OF ACHIEVEMENT

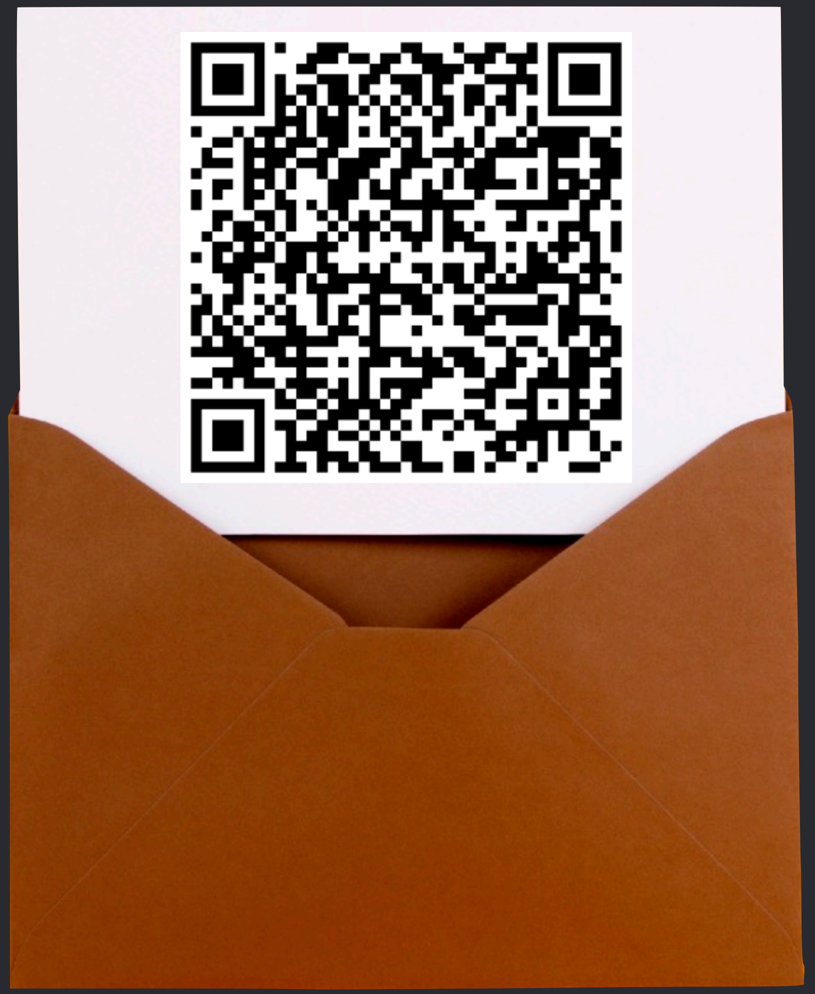
THIS CERTIFICATE IS PROUDLY PRESENTED TO

John Doe

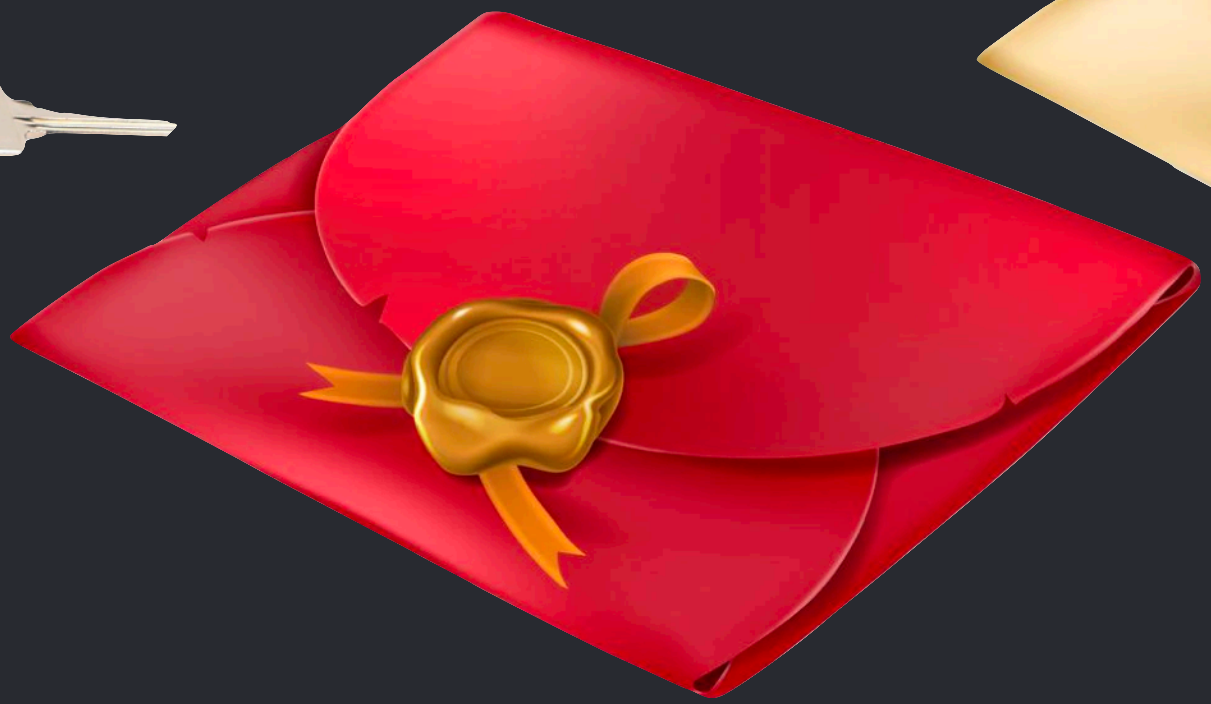
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sit amet porta leo. Mauris nec tortor ut erat venenatis porta. Integer pulvinar aliquet purus sit amet venenatis. Nulla lacus risus, sodales bibendum dolor at

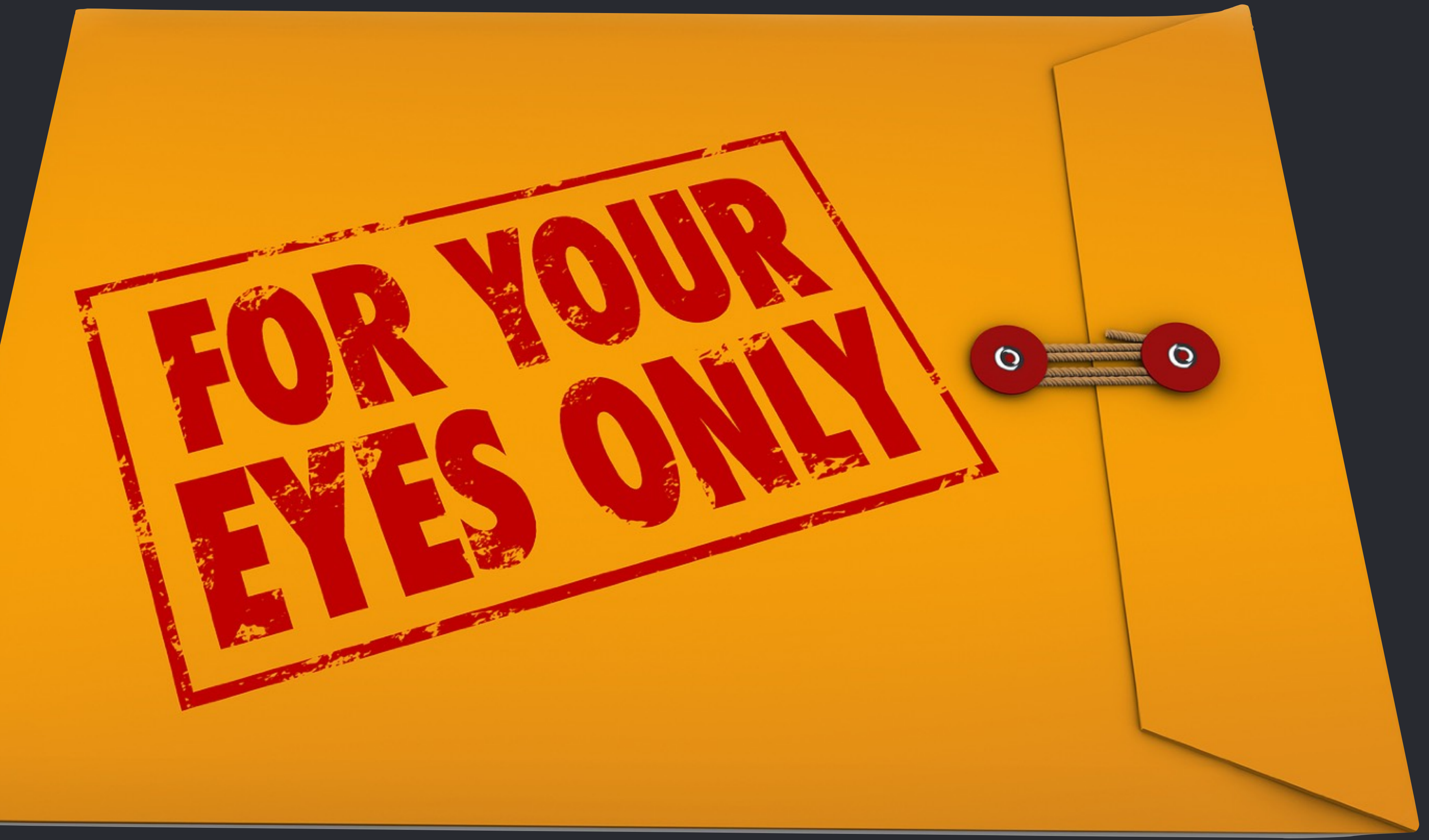
DATE

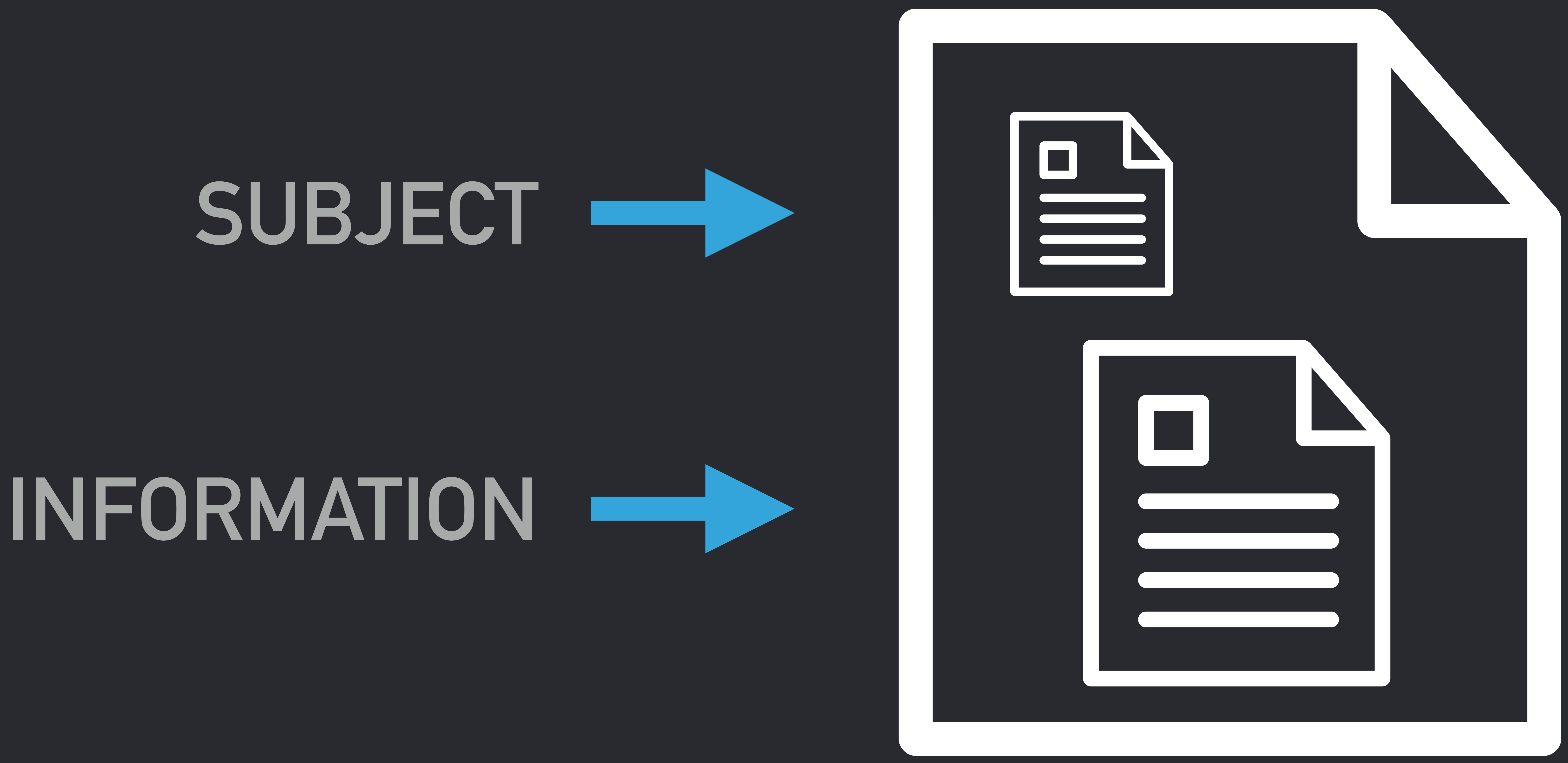
SIGNATURE

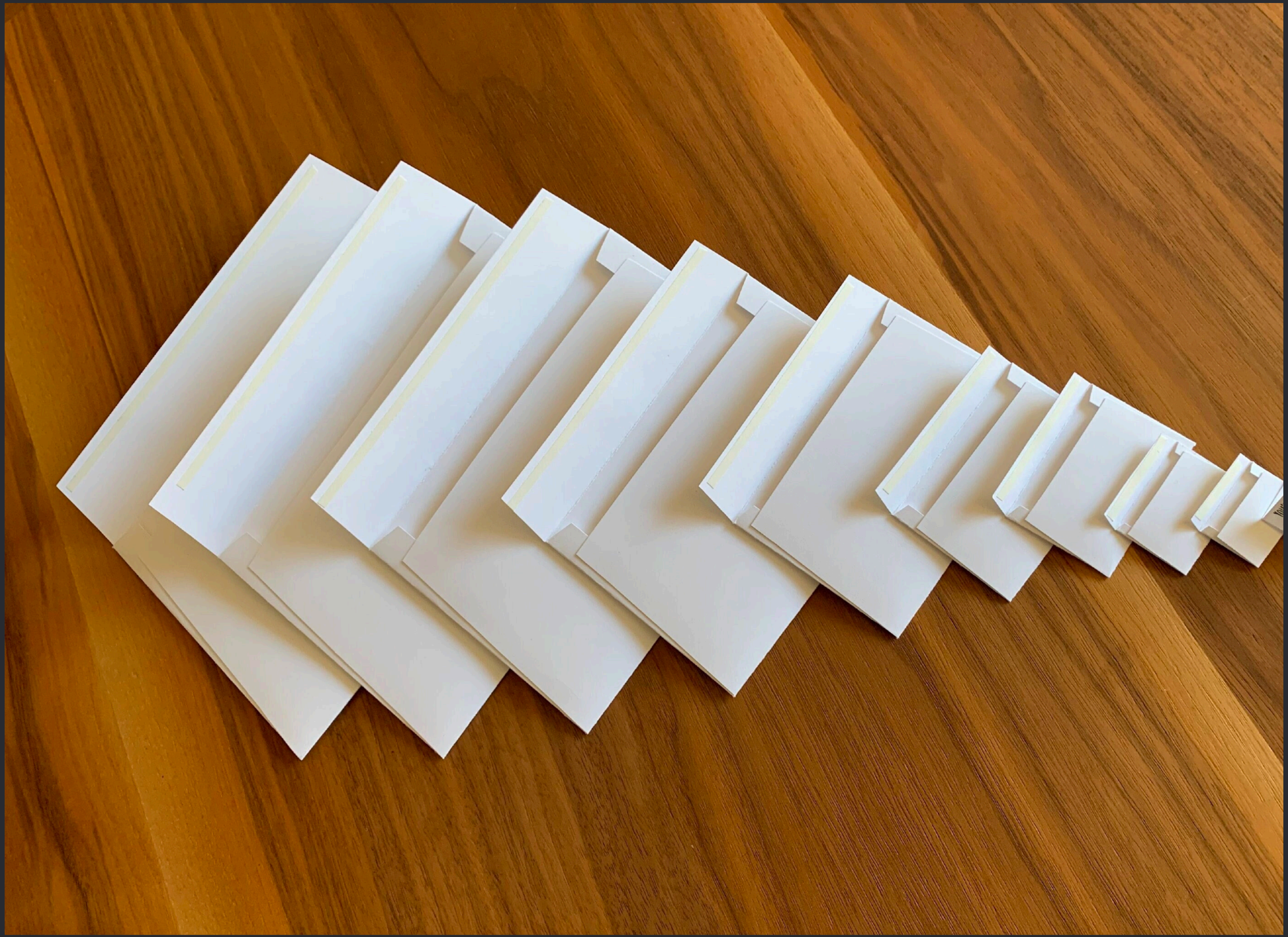


ENVELOPES CAN DO MANY THINGS






SUBJECT
INFORMATION



```swift
struct Envelope {
    let subject: Envelope
    let assertions: [Assertion]
}

struct Assertion {
    let predicate: Envelope
    let object: Envelope
}
```

```
struct Envelope {
    let subject: Envelope
    let assertions: [Assertion]
}

struct Assertion {
    let predicate: Envelope
    let object: Envelope
}
```

```
"Alice" [
    "knows": "Bob"
]
```

```swift
struct Envelope {
    let subject: Envelope
    let assertions: [Assertion]
}

struct Assertion {
    let predicate: Envelope
    let object: Envelope
}
```

```swift
enum Envelope {
    case node(subject: Envelope, assertions: [Envelope])
    case leaf(CBOR)
    case wrapped(Envelope)
    case knownValue(KnownValue)
    case assertion(Assertion)
    case encrypted(EncryptedMessage)
    case compressed(Compressed, Digest)
    case elided(Digest)
}
```

```
"Alice" [
    "knows": "Bob"
]
```

```swift
struct Envelope {
    let subject: Envelope
    let assertions: [Assertion]
}

struct Assertion {
    let predicate: Envelope
    let object: Envelope
}
```

```swift
enum Envelope {
    case node(subject: Envelope, assertions: [Envelope])
    case leaf(CBOR)
    case wrapped(Envelope)
    case knownValue(KnownValue)
    case assertion(Assertion)
    case encrypted(EncryptedMessage)
    case compressed(Compressed, Digest)
    case elided(Digest)
}
```

```
"Alice" [
    "knows": "Bob"
]
```

```
"Alice"

"knows": "Bob"

ELIDED

ENCRYPTED

COMPRESSED
```

```
"Alice" [
    "knows": "Bob"
]
```

5 POSITIONS

```
"Alice" [
    "knows": "Bob"
]
```

```
"Alice" [
    "knows": "Bob"
]
```

```
"Alice" [
    "knows": "Bob"
]
```

1 – SUBJECT

```
"Alice" [
    "knows": "Bob"
]
```

```
ELIDED [
    "knows": "Bob"
]
```

```
"Alice" [
    "knows": "Bob"
    "01": "P1"
]
```

2 – PREDICATE

```
"Alice" [
    "knows": "Bob"
]
```

```
"Alice" [
    ELIDED: "Bob"
]
```

```
"Alice" [
    "knows" [
        "O2": "P2"
    ] : "Bob"
]
```

3 - OBJECT

```
"Alice" [
    "knows": "Bob"
]
```

```
"Alice" [
    "knows": ELIDED
]
```

```
"Alice" [
    "knows": "Bob" [
        "03": "P3"
    ]
]
```

4 - ASSERTION

```
"Alice" [
    "knows": "Bob"
]
```

```
"Alice" [
    ELIDED
]
```

```
"Alice" [
    {
        "knows": "Bob"
    } [
        "04": "P4"
    ]
]
```

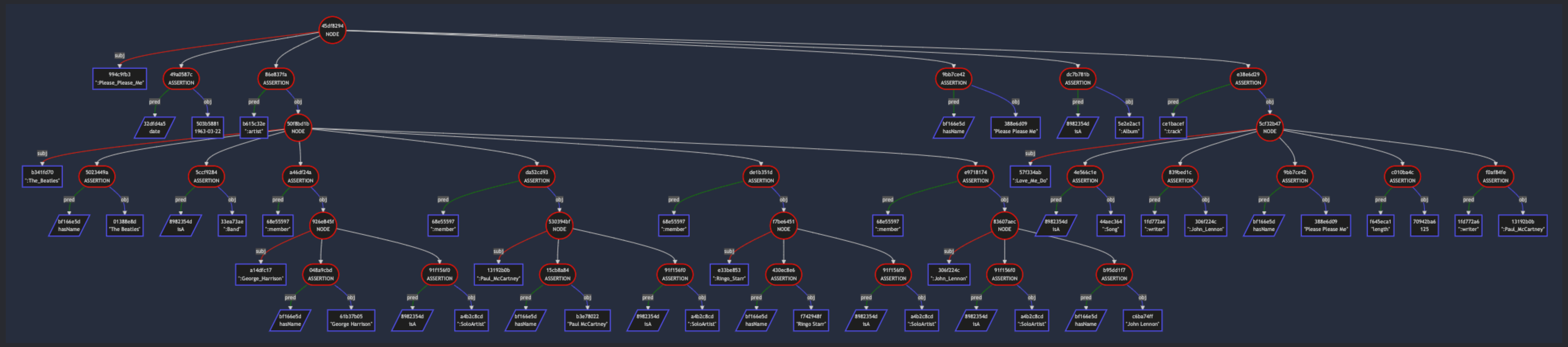
5 - ENVELOPE

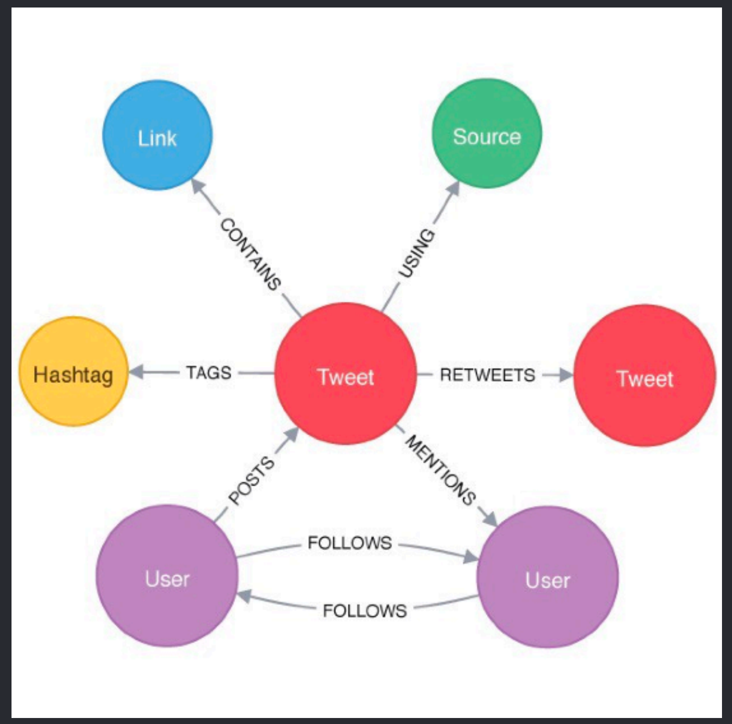
```
"Alice" [
    "knows": "Bob"
]
```

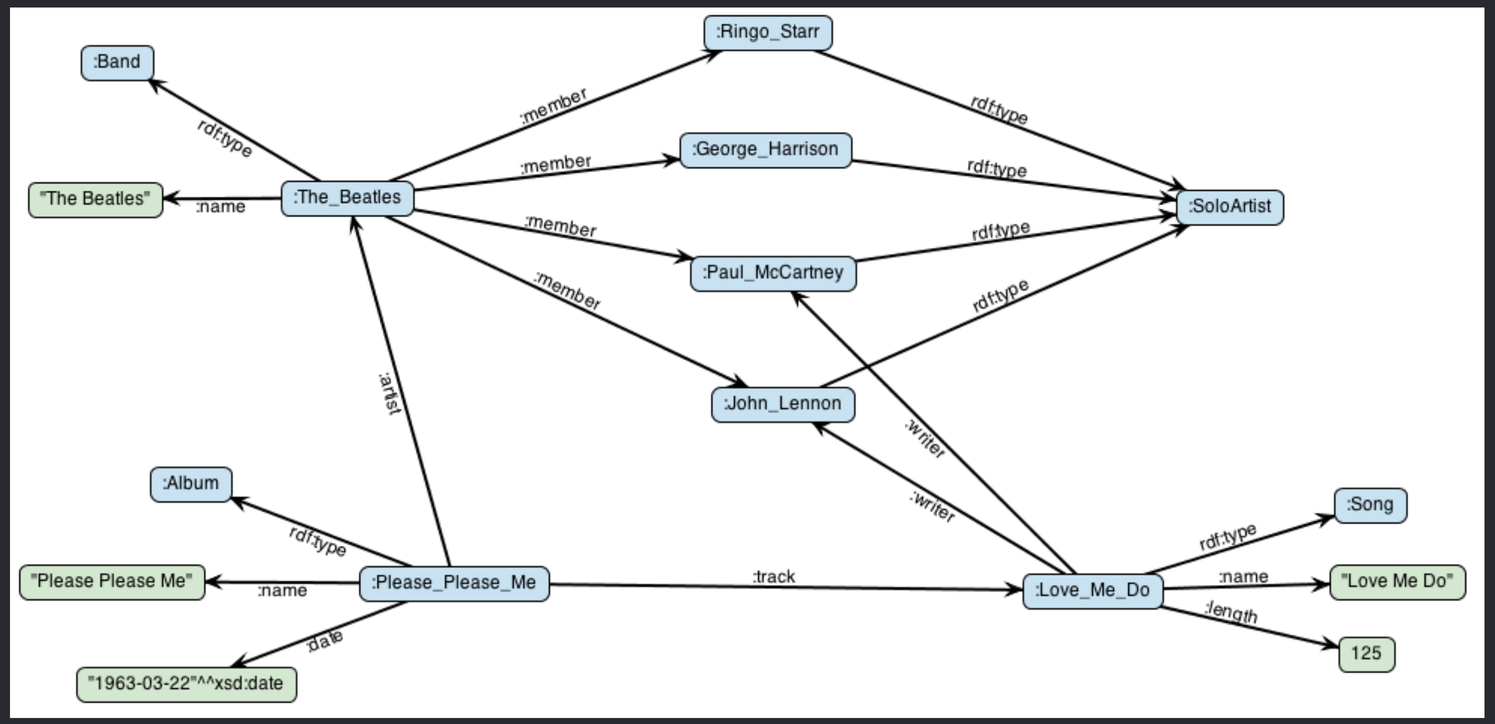
```
ELIDED
```

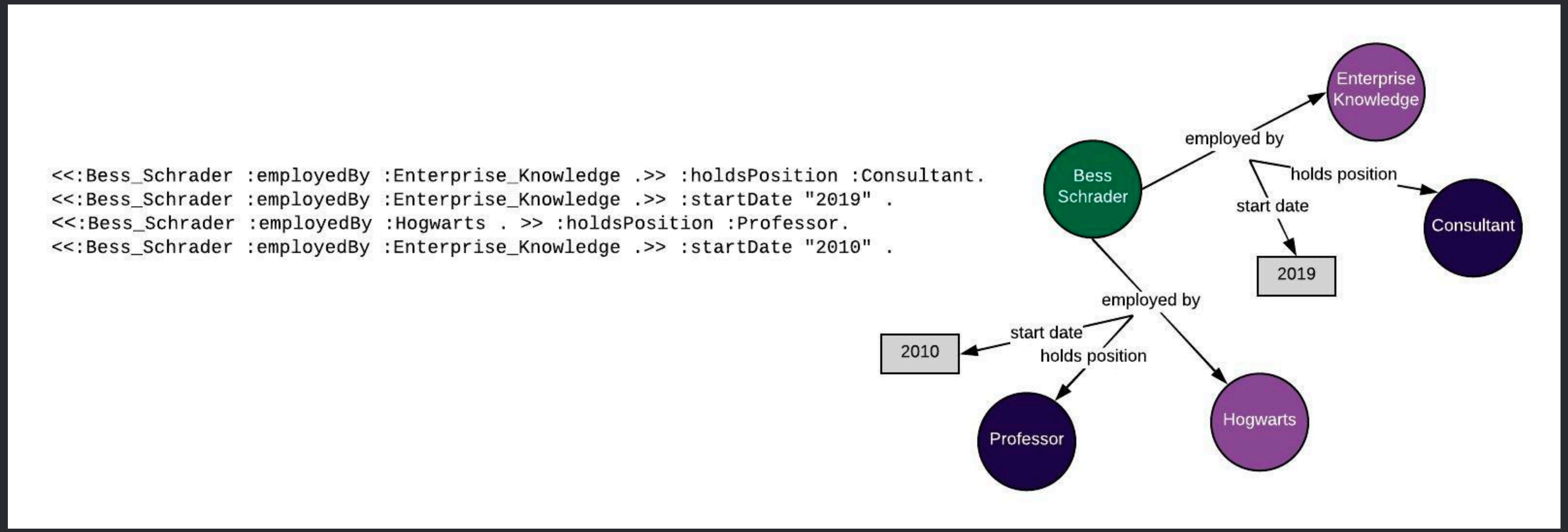
```
{
    "Alice" [
        "knows": "Bob"
    ]
} [
    "05": "P5"
]
```

STRUCTURE READY









```
<<:Bess_Schrader :employedBy :Enterprise_Knowledge .>> :holdsPosition :Consultant.
<<:Bess_Schrader :employedBy :Enterprise_Knowledge .>> :startDate "2019" .
<<:Bess_Schrader :employedBy :Hogwarts . >> :holdsPosition :Professor.
<<:Bess_Schrader :employedBy :Enterprise_Knowledge .>> :startDate "2010" .
```

```
[Envelope("Alice"), Envelope("Bob"), Envelope("Carol")]
```

USE CASES

A user can elide their content:

https://github.com/BlockchainCommons/Gordian/blob/master/Envelope/Use-Cases/Educational.md#2-danika-restricts-her-revelations-elision

A user can elide and then later reveal their content:

https://github.com/BlockchainCommons/Gordian/blob/master/Envelope/Use-Cases/Software.md#7-amira-reveals-her-identity-progressive-trust

User-based herd privacy:

https://github.com/BlockchainCommons/Gordian/blob/master/Envelope/Use-Cases/Educational.md#6-paul-proves-proficiency-with-improved-privacy-herd-privacy-with-non-correlation

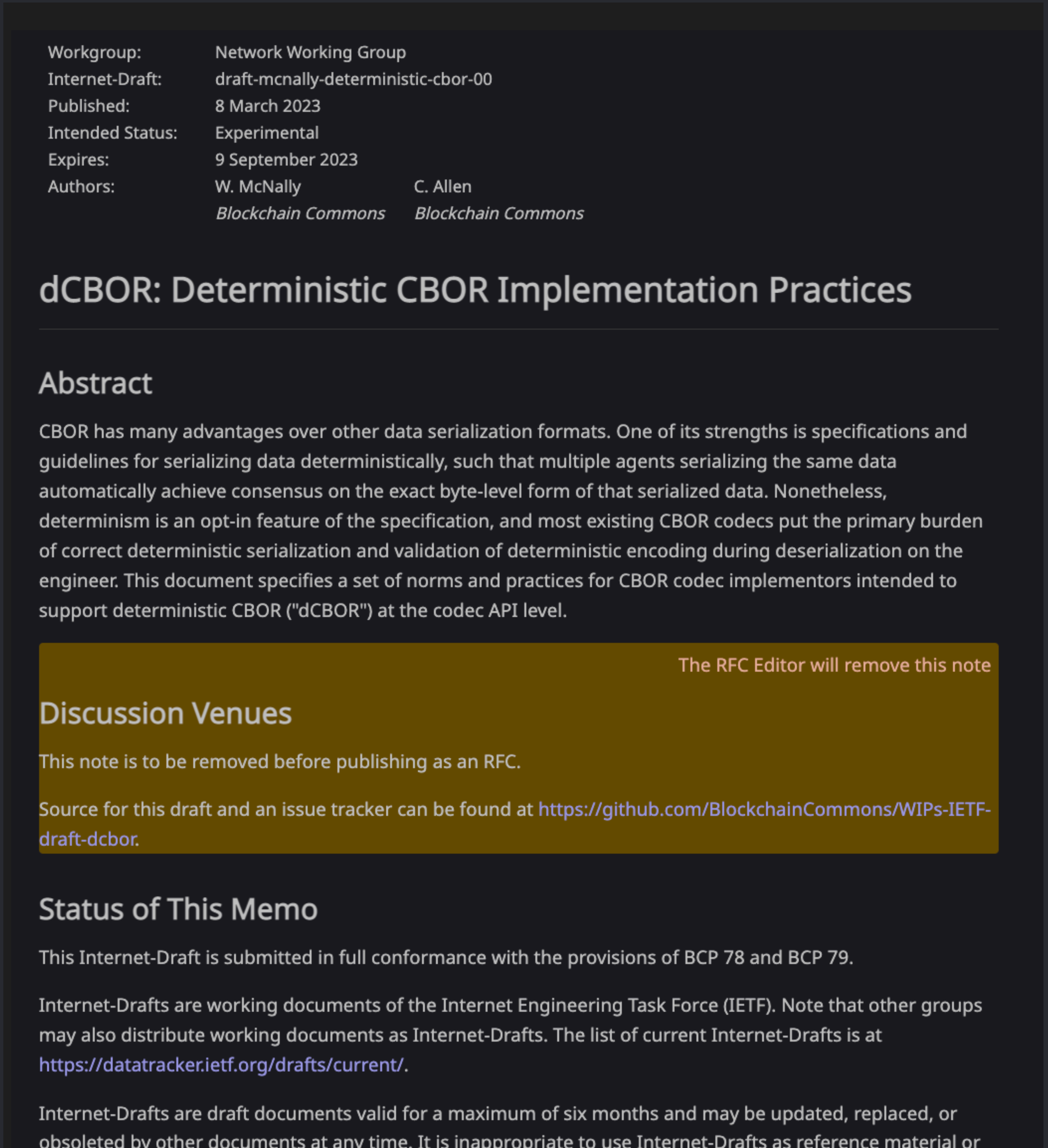
dCBOR

ASN.1 DER
PHP
Fast Infoset
JSON
Thrift
Pickle
Property list
XML-RPC
Protobuf
XML
CBOR
SDX
BSON
Bencode
CDR
Netstrings
OGDL
OPC-UA Binary
Java serialization
UBJSON
CSV
FlatBuffers
FHIR
EXI
Avro
SOAP
MessagePack
Parquet
Binn
Smile
S-expressions
YAML
Cap'n_Proto
XDR
Ion
D-Bus
OpenDDL

- BINARY
- CONCISE
- SELF-DESCRIBING
- CONSTRAINED ENVIRONMENTS
- PLATFORM/LANGUAGE AGNOSTIC
- STANDARDIZED
- DETERMINISTIC

INTERNET DRAFT

datatracker.ietf.org/doc/draft-mcnally-deterministic-cbor/



Workgroup:	Network Working Group
Internet-Draft:	draft-mcnally-deterministic-cbor-00
Published:	8 March 2023
Intended Status:	Experimental
Expires:	9 September 2023
Authors:	W. McNally *Blockchain Commons* / C. Allen *Blockchain Commons*

dCBOR: Deterministic CBOR Implementation Practices

Abstract

CBOR has many advantages over other data serialization formats. One of its strengths is specifications and guidelines for serializing data deterministically, such that multiple agents serializing the same data automatically achieve consensus on the exact byte-level form of that serialized data. Nonetheless, determinism is an opt-in feature of the specification, and most existing CBOR codecs put the primary burden of correct deterministic serialization and validation of deterministic encoding during deserialization on the engineer. This document specifies a set of norms and practices for CBOR codec implementors intended to support deterministic CBOR ("dCBOR") at the codec API level.

The RFC Editor will remove this note

Discussion Venues

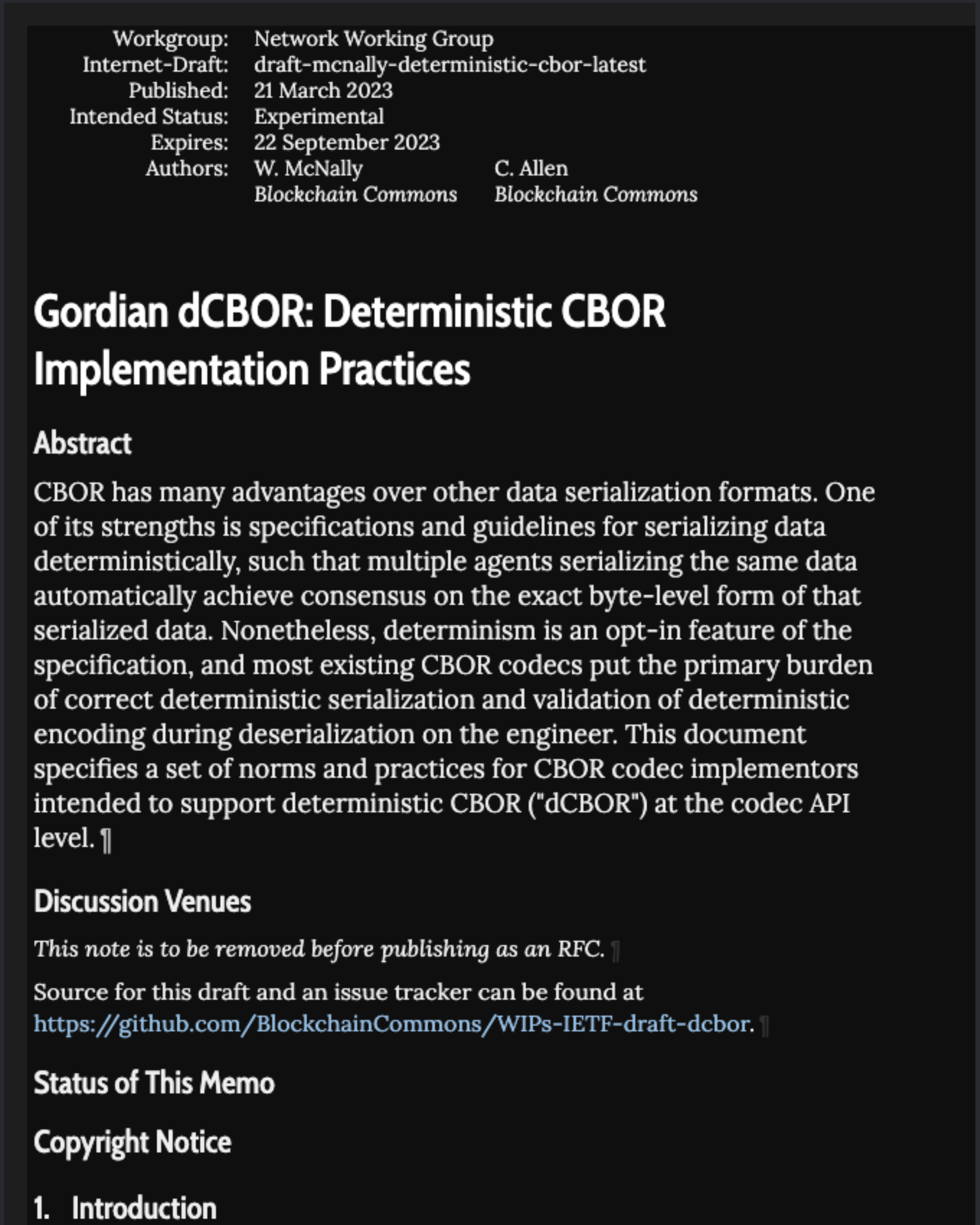
This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/BlockchainCommons/WIPs-IETF-draft-dcbor.

Status of This Memo

INTERNET DRAFT

blockchaincommons.github.io/WIPs-IETF-draft-deterministic-cbor/draft-mcnally-deterministic-cbor.html



Workgroup:	Network Working Group
Internet-Draft:	draft-mcnally-deterministic-cbor-latest
Published:	21 March 2023
Intended Status:	Experimental
Expires:	22 September 2023

Authors: W. McNally *Blockchain Commons* C. Allen *Blockchain Commons*

Gordian dCBOR: Deterministic CBOR Implementation Practices

Abstract

CBOR has many advantages over other data serialization formats. One of its strengths is specifications and guidelines for serializing data deterministically, such that multiple agents serializing the same data automatically achieve consensus on the exact byte-level form of that serialized data. Nonetheless, determinism is an opt-in feature of the specification, and most existing CBOR codecs put the primary burden of correct deterministic serialization and validation of deterministic encoding during deserialization on the engineer. This document specifies a set of norms and practices for CBOR codec implementors intended to support deterministic CBOR ("dCBOR") at the codec API level. ¶

Discussion Venues

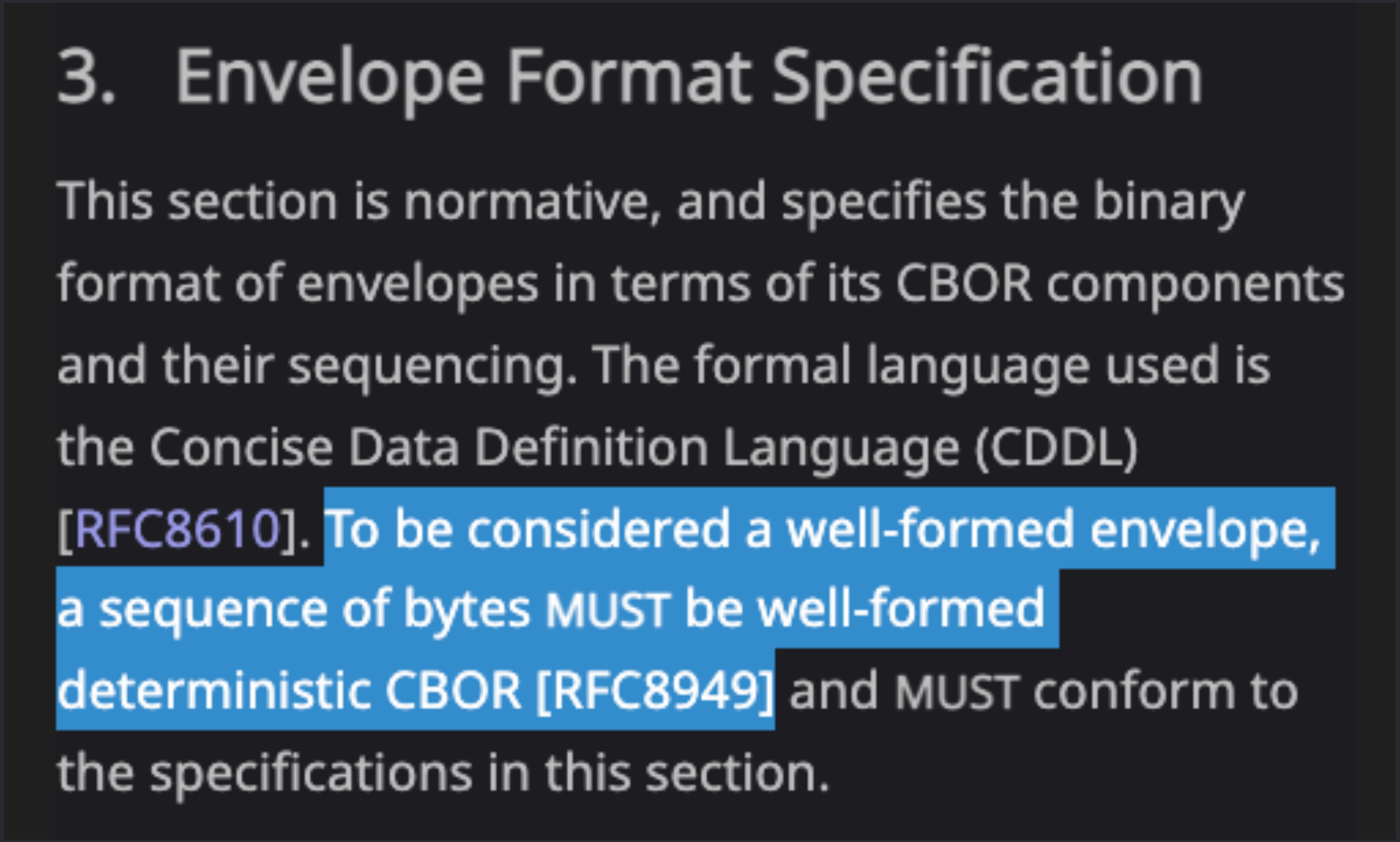
This note is to be removed before publishing as an RFC. ¶

Source for this draft and an issue tracker can be found at https://github.com/BlockchainCommons/WIPs-IETF-draft-dcbor. ¶

Status of This Memo

Copyright Notice

1. Introduction

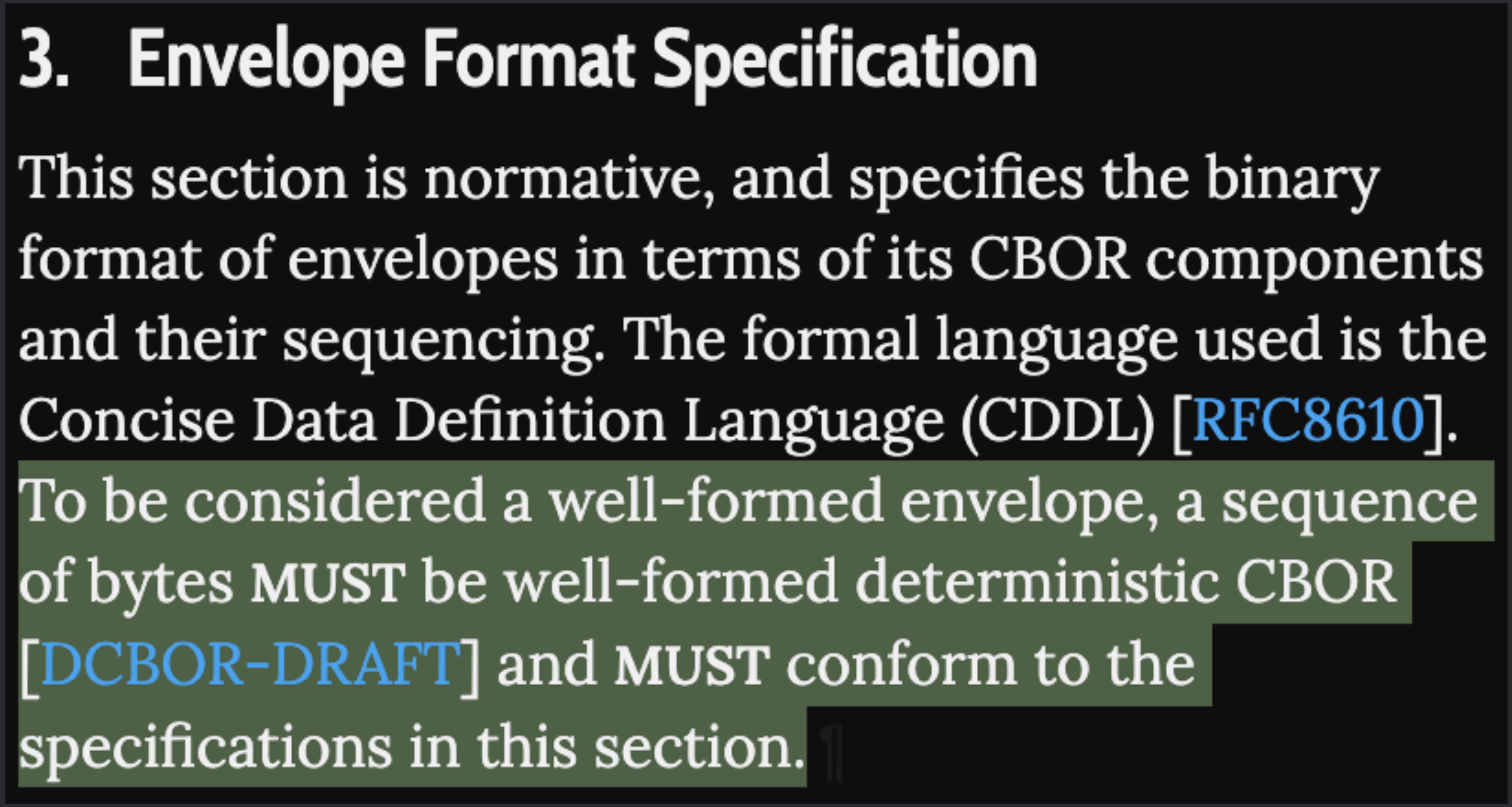


3. Envelope Format Specification

This section is normative, and specifies the binary format of envelopes in terms of its CBOR components and their sequencing. The formal language used is the Concise Data Definition Language (CDDL) [RFC8610]. To be considered a well-formed envelope, a sequence of bytes MUST be well-formed deterministic CBOR [RFC8949] and MUST conform to the specifications in this section.

- BINARY
- CONCISE
- SELF-DESCRIBING
- CONSTRAINED ENVIRONMENTS
- PLATFORM/LANGUAGE AGNOSTIC
- STANDARDIZED
- DETERMINISTIC

https://datatracker.ietf.org/doc/draft-mcnally-envelope/



3. Envelope Format Specification

This section is normative, and specifies the binary format of envelopes in terms of its CBOR components and their sequencing. The formal language used is the Concise Data Definition Language (CDDL) [RFC8610]. To be considered a well-formed envelope, a sequence of bytes **MUST** be well-formed deterministic CBOR [DCBOR-DRAFT] and **MUST** conform to the specifications in this section. ¶

- BINARY
- CONCISE
- SELF-DESCRIBING
- CONSTRAINED ENVIRONMENTS
- PLATFORM/LANGUAGE AGNOSTIC
- STANDARDIZED
- DETERMINISTIC

https://datatracker.ietf.org/doc/draft-mcnally-envelope/

THE POINT OF DETERMINISM

- Eliminate choices for how to serialize particular data.
- Where possible, the API enforces these standards.
- Where not possible, developers MUST specify how to serialize and how to validate on deserializing.
- Multiple agents serializing the same data should automatically achieve consensus on the exact form of that data.

WHAT DOES THE SPEC SAY?
RFC-8949

4.2. Deterministically Encoded CBOR

Some protocols may want encoders to only emit CBOR in a particular deterministic format; those protocols might also have the decoders check that their input is in that deterministic format. Those protocols are free to define what they mean by a "deterministic format" and what encoders and decoders are expected to do. This section defines a set of restrictions that can serve as the base of such a deterministic format.

WHAT DOES THE SPEC SAY?
RFC-8949

4.2.1. Core Deterministic Encoding Requirements

- Variable-length integers MUST be as short as possible.
- Floating-point values MUST use the shortest form that preserves the value.
- Indefinite-length arrays and maps MUST NOT be used.
- Map keys MUST be sorted in bytewise lexicographic order of their deterministic encodings.

WHAT DOES THE SPEC SAY?
RFC-8949

4.2.2. Additional Deterministic Encoding Considerations

- Protocols MUST specify the circumstances under which a data item MUST or MUST NOT be tagged.
- Protocols allowing the use of BigNums ≥ 2^{64} (tags 2 and 3) MUST specify whether values $<2^{64}$ MUST use regular integer encodings.
- Protocols allowing the use of floating-point numbers must decide how to encode values like –0.0, NaN/Signalling NaN, subnormal values, etc.

WHAT DOES BLOCKCHAIN COMMONS SAY?

- Deterministic encoding is essential for cryptographic "smart documents" like Gordian Envelope.
- All of our existing CBOR specs are already deterministic encoding-compliant.
- Being opinionated is good.
- Enforcing opinionated best practices at the software/API level is even better.

HOW MANY EXISTING CBOR IMPLEMENTATIONS DIRECTLY SUPPORT DETERMINISTIC ENCODING AS A CORE VALUE?

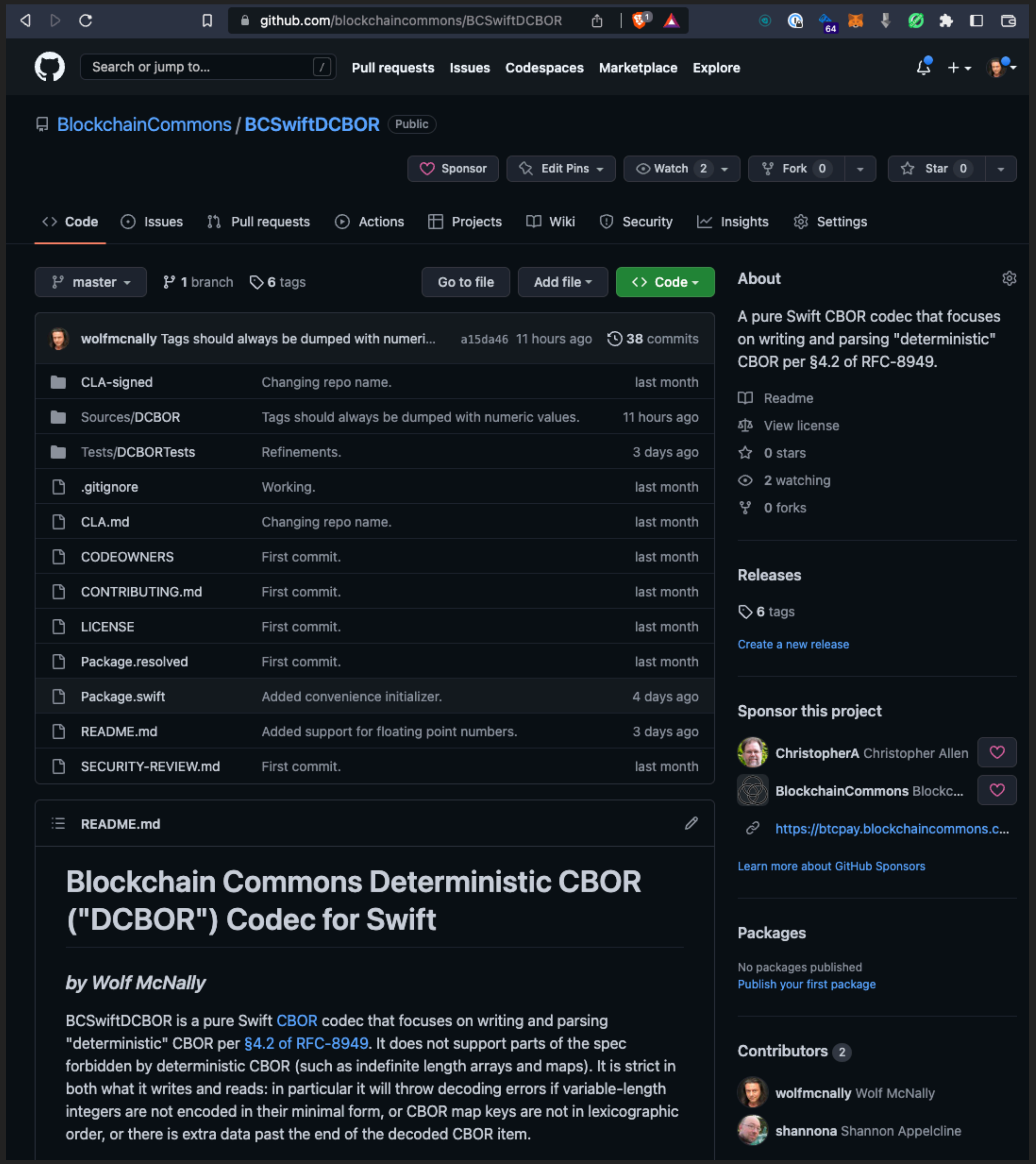
404

(none found)

SO NOW WE'VE BUILT TWO OF THEM…

dCBOR Swift

https://github.com/BlockchainCommons/BCSwiftDCBOR



github.com/blockchaincommons/BCSwiftDCBOR

Search or jump to… | Pull requests | Issues | Codespaces | Marketplace | Explore

BlockchainCommons / BCSwiftDCBOR Public

Sponsor | Edit Pins | Watch 2 | Fork 0 | Star 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | 1 branch | 6 tags | Go to file | Add file | Code

wolfmcnally Tags should always be dumped with numeri… a15da46 11 hours ago 38 commits

CLA-signed	Changing repo name.	last month
Sources/DCBOR	Tags should always be dumped with numeric values.	11 hours ago
Tests/DCBORTests	Refinements.	3 days ago
.gitignore	Working.	last month
CLA.md	Changing repo name.	last month
CODEOWNERS	First commit.	last month
CONTRIBUTING.md	First commit.	last month
LICENSE	First commit.	last month
Package.resolved	First commit.	last month
Package.swift	Added convenience initializer.	4 days ago
README.md	Added support for floating point numbers.	3 days ago
SECURITY-REVIEW.md	First commit.	last month

README.md

Blockchain Commons Deterministic CBOR ("DCBOR") Codec for Swift

by Wolf McNally

BCSwiftDCBOR is a pure Swift CBOR codec that focuses on writing and parsing "deterministic" CBOR per §4.2 of RFC-8949. It does not support parts of the spec forbidden by deterministic CBOR (such as indefinite length arrays and maps). It is strict in both what it writes and reads: in particular it will throw decoding errors if variable-length integers are not encoded in their minimal form, or CBOR map keys are not in lexicographic order, or there is extra data past the end of the decoded CBOR item.

About

A pure Swift CBOR codec that focuses on writing and parsing "deterministic" CBOR per §4.2 of RFC-8949.

Readme
View license
0 stars
2 watching
0 forks

Releases

6 tags

Create a new release

Sponsor this project

ChristopherA Christopher Allen
BlockchainCommons Blockc…
https://btcpay.blockchaincommons.c…
Learn more about GitHub Sponsors

Packages

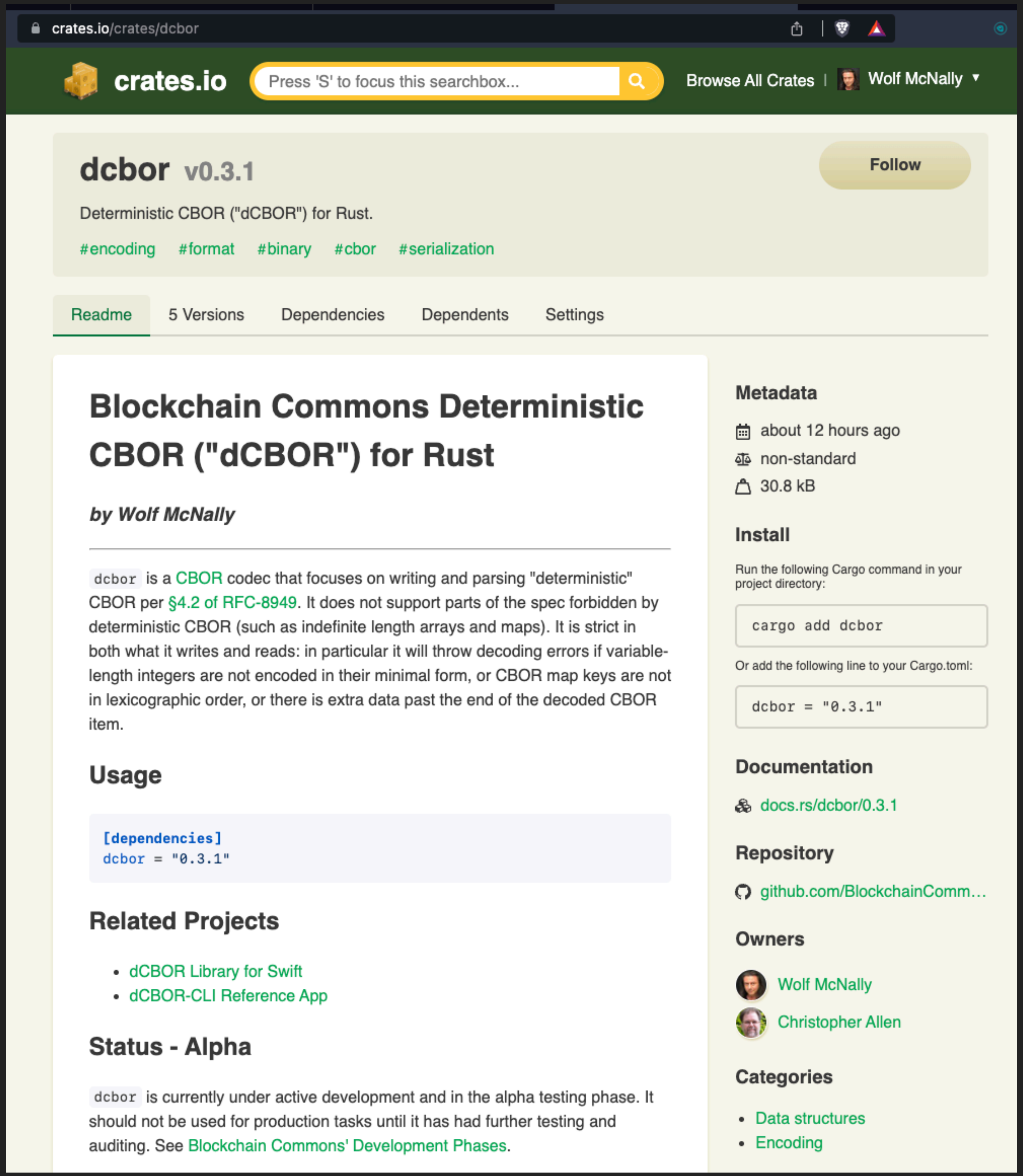
No packages published
Publish your first package

Contributors 2

wolfmcnally Wolf McNally
shannona Shannon Appelcline

dCBOR Rust

https://crates.io/crates/dcbor



crates.io/crates/dcbor

crates.io | Press 'S' to focus this searchbox… | Browse All Crates | Wolf McNally

dcbor v0.3.1

Follow

Deterministic CBOR ("dCBOR") for Rust.

#encoding #format #binary #cbor #serialization

Readme | 5 Versions | Dependencies | Dependents | Settings

Blockchain Commons Deterministic CBOR ("dCBOR") for Rust

by Wolf McNally

`dcbor` is a CBOR codec that focuses on writing and parsing "deterministic" CBOR per §4.2 of RFC-8949. It does not support parts of the spec forbidden by deterministic CBOR (such as indefinite length arrays and maps). It is strict in both what it writes and reads: in particular it will throw decoding errors if variable-length integers are not encoded in their minimal form, or CBOR map keys are not in lexicographic order, or there is extra data past the end of the decoded CBOR item.

Usage

```
[dependencies]
dcbor = "0.3.1"
```

Related Projects

- dCBOR Library for Swift
- dCBOR-CLI Reference App

Status - Alpha

`dcbor` is currently under active development and in the alpha testing phase. It should not be used for production tasks until it has had further testing and auditing. See Blockchain Commons' Development Phases.

Metadata

about 12 hours ago
non-standard
30.8 kB

Install

Run the following Cargo command in your project directory:

```
cargo add dcbor
```

Or add the following line to your Cargo.toml:

```
dcbor = "0.3.1"
```

Documentation

docs.rs/dcbor/0.3.1

Repository

github.com/BlockchainComm…

Owners

Wolf McNally
Christopher Allen

Categories

- Data structures
- Encoding

GOALS FOR dCBOR

- Make it easy to write and read deterministic CBOR (dCBOR) that complies with RFC-8949 §4.2.1. Core Deterministic Encoding Requirements.
- Be strict about what is written and read.
  - Make it hard to write non-compliant dCBOR.
  - Make it an error to read non-compliant dCBOR.
- Facilitate as much as possible the considerations in RFC-8949 §4.2.2. Additional Deterministic Encoding Considerations.

Encoding of Maps

- RFC-8949: Map keys MUST be sorted in bytewise lexicographic order of their deterministic encodings.
- dCBOR libraries provide a special-purpose `Map` structure that keeps key-value pairs in canonical sorted order as they are inserted or removed.
- Provides iteration through key-value pairs in canonical order.
- In some other ways they behave like a normal dictionary/map, but they are primarily intended for use during the serialization/deserialization process.
- Deserialization of out-of-order map keys is an error.

Encoding of Numeric Values

- All encoded numeric values use the shortest possible serialization.
  - Integers are 8, 16, 32, or 64 bits, floating point values 16, 32, or 64 bits.
- Floating point values with no fractional part are serialized as integers if possible.
  - This means that 0.0, -0.0, and 0 are all serialized exactly the same way.
  - Same semantics as JSON, JavaScript and Ruby.
- After deserialization, any numeric value can be extracted as a floating point value.
- Attempting to extract an integer from a numeric value with a fractional part is an error.
- Attempting to deserialize a dCBOR stream with any numeric values not in their canonical shortest form is an error.
- NaN is canonicalized to a single representation.

Provide protocols (Swift) and traits (Rust) to make structures CBOR-friendly.

- `CBORCodable` conformance adds serialization/deserialization to any type.
  - Many fundamental built-in types conform including integers, floating point values, strings, byte strings, arrays, booleans, and dates.
- `CBORTaggedCodable` adds a tag that is always written on serialization and expected on deserialization.
- No attempt has been made to make dCBOR compatible with either the `Codable` protocol (Swift) or the `SerDe` serialization framework (Rust).
  - A lot of work for little benefit, with many sharp edge/corner cases to deal with.
  - If this is something you desire, we welcome PRs!

Output of CBOR diagnostic notation and annotated hex dumps

- The deserialized `CBOR` type has `diagnostic()` and `hex()` methods.
- Can be provided with `knownTags` argument that provides names for tags.

```
304(   ; crypto-keypath
   {
      1:
      [23, false, 23, true, 33, false]
   }
)
```

```
d9 0130         # tag(304)   ; crypto-keypath
   a1           # map(1)
      01        # unsigned(1)
      86        # array(6)
         17     # unsigned(23)
         f4     # false
         17     # unsigned(23)
         f5     # true
         1821   # unsigned(33)
         f4     # false
```

Validations performed while decoding or extracting

```swift
/// An error decoding or parsing CBOR.
public enum CBORError: LocalizedError, Equatable {
    /// Early end of data.
    case underrun

    /// Unsupported value in CBOR header.
    ///
    /// The case includes the encountered header as associated data.
    case badHeaderValue(encountered: UInt8)

    /// A numeric value was encoded in non-canonical form.
    case nonCanonicalNumeric

    /// An invalidly-encoded UTF-8 string was encountered.
    case invalidString

    /// The decoded CBOR had extra data at the end.
    ///
    /// The case includes the number of unused bytes as associated data.
    case unusedData(Int)
```

```swift
    /// The decoded CBOR map has keys that are not in canonical order.
    case misorderedMapKey

    /// The decoded CBOR map has a duplicate key.
    case duplicateMapKey

    /// The numeric value could not be represented in the specified numeric type.
    case outOfRange

    /// The decoded value was not the expected type.
    case wrongType

    /// The decoded value did not have the expected tag.
    ///
    /// The case includes the expected tag and encountered tag as associated data.
    case wrongTag(expected: Tag, encountered: Tag)

    /// Invalid CBOR format. Frequently thrown by libraries depending on this one.
    case invalidFormat
}
```

CHRISTOPHER ALLEN
christophera@lifewithalacrity.com
@BlockchainComns

WOLF MCNALLY
wolf@wolfmcnally.com
@WolfMcNally

List of Envelope resource links:

https://www.blockchaincommons.com/introduction/Envelope-Intro/#envelope-links