CBOR
Internet-Draft
Intended status: Standards Track
Expires: 8 May 2026

W. McNally, Ed.
Blockchain Commons

4 November 2025

A CBOR Tag for Lossless Transport of IEEE-754 NaN Bit Patterns
draft-mcnally-cbor-nan-bstr-01

Abstract

IEEE-754 NaN formations are not numbers and have no equivalence class. They are not comparable to anything, including reflexively, and therefore each attribute - sign bit, signaling/quiet bit, payload bits, and representation width (binary16, binary32, binary64, binary128) - can carry meaning for an implementation. CBOR permits encoding NaNs as floating-point values, but generic processing, preferred serialization, and deterministic profiles may canonicalize or otherwise alter NaN encodings, losing information. This document defines a CBOR tag, colloquially "nan-bstr", whose content is a byte string carrying the exact IEEE-754 NaN bit pattern so that all attributes are preserved across encode/decode, enabling use cases like NaN boxing, platform-specific error signaling, diagnostics, and forensics.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-mcnally-cbor-nan-bstr/.

Discussion of this document takes place on the CBOR Working Group mailing list (mailto:cbor@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/cbor/. Subscribe at https://www.ietf.org/mailman/listinfo/cbor/.

Source for this draft and an issue tracker can be found at https://github.com/BlockchainCommons/draft-mcnally-cbor-nan-bstr.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 8 May 2026.

Copyright Notice

Table of Contents

1. Introduction

CBOR defines an extensible binary format with semantic tags (major type 6) that can ascribe additional meaning to enclosed data items ([RFC8949]). While CBOR's floating-point types can encode NaN values, encoders and application profiles commonly canonicalize NaNs or collapse them to a single preferred representation. For applications that rely on specific NaN formations, this behavior is unacceptable. This specification defines a single CBOR tag that wraps a CBOR byte string (bstr) containing 2, 4, 8, or 16 octets representing the big-endian bit pattern of a single IEEE-754 NaN (binary16/32/64/128). The tag enables exact round-tripping of NaN attributes independent of the policies that an ecosystem applies to floating-point numbers.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Motivation and Rationale

IEEE-754 purposefully leaves NaNs incomparable and allows implementations to use sign, signaling/quiet, and payload bits for implementation-defined purposes ([IEEE754]). When CBOR encoders perform preferred serialization or when deterministic profiles constrain encodings for predictability (see [RFC8949], Section 4.2), the precise NaN formation can be lost. A narrowly scoped tag that treats a NaN as an opaque bit pattern avoids interfering with numeric semantics while providing an explicit, interoperable mechanism to preserve all attributes. This mechanism addresses concrete needs: (1) NaN boxing schemes use NaN payload space to embed tagged values and pointers; preserving those bits across transports is essential ([JSC-NaNBox]). (2) Some profiles, such as dCBOR, intentionally allow only a single canonical NaN and reject others to maximize determinism; those ecosystems still benefit from an explicit escape hatch when a precise NaN must be preserved ([I-D.mcnally-deterministic-cbor]). (3) Other communities canonicalize NaNs for determinism (for example, WebAssembly discussions), illustrating the tension between reproducibility and information retention ([WASM-NaN]). See also prior CBOR WG threads that discussed NaN handling and determinism ([CBOR-WG-NaN-Threads]). A tag solves the transport problem without weakening profile policies

for ordinary numeric values.

4. The nan-bstr Tag

4.1. Semantics

The nan-bstr tag denotes that its content is a CBOR byte string whose bytes are, in network byte order (big-endian), the bit pattern of a single IEEE-754 NaN in one of the three interchange widths: 2 bytes (binary16), 4 bytes (binary32), 8 bytes (binary64), or 16 bytes (binary128). The tag does not change IEEE-754 NaN semantics: it does not define equality or ordering for NaNs. A tagged NaN is not a CBOR floating-point number; it is an opaque bit pattern that an application MAY convert to a native NaN when desired, acknowledging that the conversion could lose information if the platform cannot represent signaling NaNs or payload bits.

4.2. Validity

A decoder that understands this tag MUST enforce all of the following: (1) The enclosed bstr length is exactly 2, 4, 8, or 16. (2) The bytes are interpreted in network byte order, consistent with CBOR’s encoding of multi-byte values ([RFC8949]). (3) The bit pattern is a NaN for the indicated width (that is, exponent is all ones and the significand/fraction field is non-zero per [IEEE754]). (4) No normalization or canonicalization of the payload, sign bit, signaling/quiet bit, or width is performed by the tag processing itself. If any check fails, the decoder MUST treat the tagged value as invalid for this tag and handle it per application error policy.

4.3. Deterministic Encoding and Preferred Serialization

CBOR’s preferred serialization and deterministically encoded CBOR rules ([RFC8949], Section 4.2) apply to the tag and to the bstr’s container (for example, definite length), not to the content bytes themselves. The content of the bstr is application-defined and MUST be preserved exactly. When an application needs exact preservation of a NaN, the sender SHOULD use this tag in place of a floating-point NaN literal.

4.4. Interoperability with Deterministic Rules and dCBOR

Deterministic rules often restrict or canonicalize floating-point representations to ensure byte-for-byte stability. For example, the dCBOR rules allows only a single canonical NaN formation (half-width value with CBOR representation 0xf97e00) and rejects others ([I-D.mcnally-deterministic-cbor]). The output of deterministic rule sets can still carry non-canonical NaNs by allowing this tag:

encoders emit nan-bstr when exact NaN preservation is required and emit the canonical NaN otherwise. This keeps deterministic rules for numbers intact while providing a precise transport for exceptional cases.

5. Examples and Diagnostic Notation

The requested tag number for this specification is 102. Diagnostic notation shows tags in decimal by default. The following table presents examples for each supported width:

Width	Description	Diagnostic Notation	CBOR Encoding (hex)
binary16	Half-precision quiet NaN (0x7E00)	102(h'7E00')	D866 42 7E00
binary32	Single-precision quiet NaN with payload 0x000001	102(h'7FC0 0001')	D866 44 7FC0 0001
binary64	Double-precision signaling NaN with payload 0x00000000000001 and sign bit set	102(h'FFF0 0000 0000 0001')	D866 48 FFF0 0000 0000 0001
binary128	Quad-precision quiet NaN with payload 0x00000000000000000000000000000001	102(h'7FFF 8000 0000 0000 0000 0000 0000 0001')	D866 50 7FFF 8000 0000 0000 0000 0000 0000 0001

Table 1

In all cases, the content preserves sign, signaling/quiet, payload bits, and width exactly; applications that cannot natively represent a formation still retain the bit pattern for pass-through or later analysis.

6. CDDL

The following CDDL ([RFC8610]) defines the tagged value.

```
nan-bstr = #6.102 (bstr .size (2 / 4 / 8 / 16))
```

7. Security Considerations

Treat the bstr as untrusted input. A decoder MUST validate NaN well-formedness for the indicated width. Implementations MUST avoid triggering floating-point exceptions when merely transporting the bit pattern. Because payload bits can carry platform-specific metadata, applications should consider confidentiality and integrity requirements when transporting tagged NaNs. The tag restricts size to 2/4/8 bytes and introduces no variable-length resource concerns beyond those of CBOR bstr handling ([RFC8949]).

8. IANA Considerations

IANA is requested to register one new entry in the CBOR Tags registry ([IANA-CBOR-TAGS]) using the template of [RFC8949], Section 9.2.

* Tag: 102 (Specification Required range).
* Data item: byte string.
* Semantics: IEEE-754 NaN encoded as a 2, 4, 8, or 16 octet byte string (nan-bstr).
* Reference: This document.

9. Implementation Notes

This section is non-normative.

* Encoders SHOULD use definite-length bstrs.
* Decoders SHOULD expose APIs that surface width, sign, signaling/quiet, and payload without mutating bits.
* Implementations SHOULD avoid using nan-bstr values as map keys because NaN equivalence is undefined; generic CBOR guidance on key equivalence applies ([RFC8949]).
* Rules that otherwise canonicalize floating-point NaNs can retain those rules and treat nan-bstr as the explicit mechanism for exact preservation when needed.

10. Alternatives Considered

This section is non-normative.

The CBOR Tags registry defines tags 80-87 for various permutations of "IEEE 754 binary<length>, <endianness>, Typed Array" ([IANA-CBOR-TAGS]). These tags may be used to preserve exact bit patterns of floating-point values, including NaNs. However, they do not connote the special semantics of NaNs, such as incomparability and payload significance. Using those tags for NaNs could mislead implementers into treating tagged values as ordinary numbers, risking unintended comparisons or arithmetic. A dedicated nan-bstr tag clarifies intent and avoids semantic confusion.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

11.2. Informative References

[CBOR-WG-NaN-Threads] "CBOR WG mailing list threads on NaN and determinism", n.d., <https://mailarchive.ietf.org/arch/browse/cbor/?q=NaN>.

[I-D.mcnally-deterministic-cbor] McNally, W., Allen, C., Bormann, C., and L. Lundblade, "dCBOR: Deterministic CBOR", Work in Progress, Internet-

Draft, draft-mcnally-deterministic-cbor-14, 1 November 2025, <https://datatracker.ietf.org/doc/html/draft-mcnally-deterministic-cbor-14>.

[IANA-CBOR-TAGS] "Concise Binary Object Representation (CBOR) Tags Registry", n.d., <https://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml>.

[IEEE754] "IEEE Standard for Floating-Point Arithmetic", IEEE Std 754-2019, n.d., <https://ieeexplore.ieee.org/document/8766229>.

[JSC-NaNBox] "JavaScriptCore NaN Boxing exercise", n.d., <https://browser.training.ret2.systems/content/module_1/8_jsc_internals/exercises/jsc_nan_boxing>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[WASM-NaN] "WebAssembly issue", n.d., <https://github.com/WebAssembly/design/issues/1463>.

Appendix A. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of publication, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

A.1. Rust Implementation

Organization: Blockchain Commons

Name: cbor-nan-bstr-rust

Description: A Rust implementation providing encoding and decoding support for CBOR tag 102 (nan-bstr). The implementation includes functions to encode IEEE-754 NaN bit patterns as CBOR byte strings and decode them back, preserving all NaN attributes including sign, signaling/quiet bit, payload bits, and width.

Level of Maturity: Development

Coverage: The implementation covers encoding and decoding of all four supported widths (binary16, binary32, binary64, binary128) with full validation of NaN bit patterns.

Version Compatibility: This implementation is compatible with the current draft specification.

Licensing: BSD-2-Clause-Patent

Implementation Experience: No issues have been encountered during development. The implementation demonstrates that the tag provides a practical mechanism for exact NaN preservation across CBOR encode/decode cycles.

Contact: Wolf McNally (wolf@wolfmcnally.com)

URL: https://github.com/BlockchainCommons/cbor-nan-bstr-rust

Appendix B. Acknowledgments

Thanks to CBOR WG participants for discussion of determinism and floating-point edge cases, and to implementers who documented NaN canonicalization and processing behavior across platforms.

Author’s Address

Wolf McNally (editor)
Blockchain Commons
Email: wolf@wolfmcnally.com