

Concise Binary Object Representation (CBOR) Sequences
draft-ietf-cbor-sequence-01

Abstract

This document describes the Concise Binary Object Representation (CBOR) Sequence format and associated media type "application/cbor-seq". A CBOR Sequence consists of any number of encoded CBOR data items, simply concatenated in sequence.

Structured syntax suffixes for media types allow other media types to build on them and make it explicit that they are built on an existing media type as their foundation. This specification defines and registers "+cbor-seq" as a structured syntax suffix for CBOR Sequences.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 3, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	Conventions Used in This Document	3
2.	CBOR Sequence Format	3
3.	The "+cbor-seq" Structured Syntax Suffix	4
4.	Practical Considerations	4
4.1.	Specifying CBOR Sequences in CDDL	4
4.2.	Diagnostic Notation	5
4.3.	Optimizing CBOR Sequences for Skipping Elements	5
5.	Security Considerations	6
6.	IANA Considerations	6
6.1.	Media Type	6
6.2.	CoAP Content-Format Registration	7
6.3.	Structured Syntax Suffix	7
7.	References	9
7.1.	Normative References	9
7.2.	Informative References	9
	Acknowledgements	10
	Author's Address	10

[1.](#) Introduction

The Concise Binary Object Representation (CBOR) [[RFC7049](#)] can be used for serialization of data in the JSON [[RFC8259](#)] data model or in its own, somewhat expanded data model. When serializing a sequence of such values, it is sometimes convenient to have a format where these sequences can simply be concatenated to obtain a serialization of the concatenated sequence of values, or to encode a sequence of values that might grow at the end by just appending further CBOR data items.

This document describes the concept and format of "CBOR Sequences", which are composed of zero or more encoded CBOR data items. CBOR Sequences can be consumed (and produced) incrementally without requiring a streaming CBOR parser that is able to deliver substructures of a data item incrementally (or a streaming encoder able to encode from substructures incrementally).

This document defines and registers the "application/cbor-seq" media type in the media type registry, along with a CoAP Content-Format identifier. Media type structured syntax suffixes [[RFC6838](#)] were introduced as a way for a media type to signal that it is based on

Bormann

Expires March 3, 2020

[Page 2]

another media type as its foundation. CBOR [[RFC7049](#)] defines the "+cbor" structured syntax suffix. This document defines and registers the "+cbor-seq" structured syntax suffix in the "Structured Syntax Suffix Registry".

1.1. Conventions Used in This Document

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.

In this specification, the term "byte" is used in its now-customary sense as a synonym for "octet".

2. CBOR Sequence Format

Formally, a CBOR Sequence is a sequence of bytes that is recursively defined as either

- o an empty (zero-length) sequence of bytes
- o the sequence of bytes making up an encoded CBOR data item [[RFC7049](#)], followed by a CBOR Sequence.

In short, concatenating zero or more encoded CBOR data items generates a CBOR Sequence. (Consequently, concatenating zero or more CBOR Sequences also results in a CBOR Sequence.)

There is no end of sequence indicator. (If one is desired, CBOR-encoding an array of the CBOR data model values being encoded -- employing either a definite or an indefinite length encoding -- as a single CBOR data item may actually be the more appropriate representation.)

CBOR Sequences, unlike JSON Text Sequences [[RFC7464](#)], do not use a marker between items. This is possible because CBOR encoded data items are self-delimiting and the end can always be calculated. (Note that, while the early object/array-only form of JSON was self-delimiting as well, this stopped being the case when simple values such as single numbers were made valid JSON documents.)

Decoding a CBOR Sequence works as follows:

- o If the CBOR Sequence is an empty sequence of bytes, the result is an empty sequence of CBOR data model values.

- o Otherwise, decode a single CBOR data item from the bytes of the CBOR sequence, and insert the resulting CBOR data model value at the start of the result of decoding the rest of the bytes as a CBOR sequence. (A streaming decoder would therefore simply deliver zero or more CBOR data model values, each of which as soon as the bytes making it up are available.)

This means that if any data item in the sequence is not well-formed, it is not possible to reliably decode the rest of the sequence. (An implementation may be able to recover from some errors in a sequence of bytes that is almost, but not entirely a well-formed encoded CBOR data item. Handling malformed data is outside the scope of this specification.)

This also means that the CBOR Sequence format can reliably detect truncation of the bytes making up the last CBOR data item in the sequence, but not entirely missing CBOR data items at the end. A CBOR Sequence decoder that is used for consuming streaming CBOR Sequence data may simply pause for more data (e.g., by suspending and later resuming decoding) in case a truncated final item is being received.

3. The "+cbor-seq" Structured Syntax Suffix

The use case for the "+cbor-seq" structured syntax suffix is the same as for "+cbor": It SHOULD be used by a media type when parsing the bytes of the media type object as a CBOR Sequence leads to a meaningful result that is at least sometimes not just a single CBOR data item. (Without the qualification at the end, this sentence is trivially true for any +cbor media type, which of course should continue to use the "+cbor" structured syntax suffix.)

Applications encountering a "+cbor-seq" media type can then either simply use generic processing if all they need is a generic view of the CBOR Sequence, or they can use generic CBOR Sequence tools for initial parsing and then implement their own specific processing on top of that generic parsing tool.

4. Practical Considerations

4.1. Specifying CBOR Sequences in CDDL

In CDDL [[RFC8610](#)], CBOR sequences are already supported as contents of byte strings using the ".cborseq" control operator ([Section 3.8.4 of \[RFC8610\]](#)), by employing an array as the controller type:


```
my-embedded-cbor-seq = bytes .cborseq my-array
my-array = [* my-element]
my-element = my-foo / my-bar
```

CDDL currently does not provide for unadorned CBOR sequences as a top-level subject of a specification. For now, the suggestion is to use an array, as for the ".cborseq" control operator, for the top-level rule and add English text that explains that the specification is really about a CBOR sequence with the elements of the array:

```
; This defines an array, the elements of which are to be used
; in a CBOR sequence:
my-sequence = [* my-element]
my-element = my-foo / my-bar
```

(Future versions of CDDL may provide a notation for top-level CBOR sequences, e.g. by using a group as the top-level rule in a CDDL specification.)

4.2. Diagnostic Notation

CBOR diagnostic notation (see [Section 6 of \[RFC7049\]](#)) or extended diagnostic notation (Appendix G of [\[RFC8610\]](#)) also does not provide for unadorned CBOR Sequences at this time (the latter does provide for CBOR Sequences embedded in a byte string in [Appendix G.3 of \[RFC8610\]](#)).

In a similar spirit to the recommendation for CDDL above, this specification recommends enclosing the CBOR data items in an array. In a more informal setting, where the boundaries within which the notation is used are obvious, it is also possible to leave off the outer brackets for this array, as shown in these two examples:

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

```
1, 2, 3
```

Note that it is somewhat difficult to discuss zero-length CBOR Sequences in the latter form.

4.3. Optimizing CBOR Sequences for Skipping Elements

In certain applications, being able to efficiently skip an element without the need for decoding its substructure, or efficiently fanning out elements to multi-threaded decoding processes, is of the utmost importance. For these applications, byte strings (which carry length information in bytes) containing embedded CBOR can be used as the elements of a CBOR sequence:


```
; This defines an array of CBOR byte strings, the elements of which
; are to be used in a CBOR sequence:
my-sequence = [* my-element]
my-element = bytes .cbor my-element-structure
my-element-structure = my-foo / my-bar
```

Within limits, this may also enable recovering from elements that internally are not well-formed -- the limitation is that the sequence of byte strings does need to be well-formed as such.

5. Security Considerations

The security considerations of CBOR [RFC7049] apply. This format provides no cryptographic integrity protection of any kind, but can be combined with security specifications such as COSE [RFC8152] to do so.

As usual, decoders must operate on input that is assumed to be untrusted. This means that decoders must fail gracefully in the face of malicious inputs.

6. IANA Considerations

6.1. Media Type

Media types are registered in the media types registry [IANA.media-types]. IANA is requested to register the MIME media type for CBOR Sequence, application/cbor-seq, as follows:

Type name: application

Subtype name: cbor-seq

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: See RFCthis, [Section 5](#).

Interoperability considerations: Described herein.

Published specification: RFCthis.

Applications that use this media type: Data serialization and deserialization.

Fragment identifier considerations: N/A

Additional information:

- o Deprecated alias names for this type: N/A
- o Magic number(s): N/A
- o File extension(s): N/A
- o Macintosh file type code(s): N/A

Person & email address to contact for further information:
cbor@ietf.org

Intended usage: COMMON

Author: Carsten Bormann (cabo@tzi.org)

Change controller: IETF

6.2. CoAP Content-Format Registration

IANA is requested to assign a CoAP Content-Format ID for the media type "application/cbor-seq", in the CoAP Content-Formats subregistry of the core-parameter registry [[IANA.core-parameters](#)], from the "Expert Review" (0-255) range. The assigned ID is shown in Table 1.

Media type	Encoding	ID	Reference
application/cbor-seq	-	TBD63	RFCthis

Table 1: CoAP Content-Format ID

RFC editor: Please replace TBD63 by the number actually assigned and delete this paragraph.

6.3. Structured Syntax Suffix

Structured Syntax Suffixes are registered within the "Structured Syntax Suffix Registry" maintained at [[IANA.media-type-structured-suffix](#)]. IANA is requested to register the "+cbor-seq" structured syntax suffix in accordance with [[RFC6838](#)], as follows:

Name: CBOR Sequence

+suffix: +cbor-seq

References: RFCthis

Encoding considerations: binary

Fragment identifier considerations: The syntax and semantics of fragment identifiers specified for +cbor-seq SHOULD be as specified for "application/cbor-seq". (At publication of this document, there is no fragment identification syntax defined for "application/cbor-seq".)

The syntax and semantics for fragment identifiers for a specific "xxx/yyy+cbor-seq" SHOULD be processed as follows:

For cases defined in +cbor-seq, where the fragment identifier resolves per the +cbor-seq rules, then process as specified in +cbor-seq.

For cases defined in +cbor-seq, where the fragment identifier does not resolve per the +cbor-seq rules, then process as specified in "xxx/yyy+cbor-seq".

For cases not defined in +cbor-seq, then process as specified in "xxx/yyy+cbor-seq".

Interoperability considerations: n/a

Security considerations: See RFCthis, [Section 5](#)

Contact: CBOR WG mailing list (cbor@ietf.org), or any IESG-designated successor.

Author/Change controller: IETF

7. References

7.1. Normative References

- [IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE) Parameters",
<<http://www.iana.org/assignments/core-parameters>>.
- [IANA.media-type-structured-suffix]
IANA, "Structured Syntax Suffix Registry",
<<http://www.iana.org/assignments/media-type-structured-suffix>>.
- [IANA.media-types]
IANA, "Media Types",
<<http://www.iana.org/assignments/media-types>>.
- [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/info/rfc2119>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", [RFC 7049](#), DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [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/info/rfc8174>>.

7.2. Informative References

- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", [BCP 13](#), [RFC 6838](#), DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.
- [RFC7464] Williams, N., "JavaScript Object Notation (JSON) Text Sequences", [RFC 7464](#), DOI 10.17487/RFC7464, February 2015, <<https://www.rfc-editor.org/info/rfc7464>>.
- [RFC8091] Wilde, E., "A Media Type Structured Syntax Suffix for JSON Text Sequences", [RFC 8091](#), DOI 10.17487/RFC8091, February 2017, <<https://www.rfc-editor.org/info/rfc8091>>.

- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", [RFC 8152](#), DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/info/rfc8152>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, [RFC 8259](#), DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [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/info/rfc8610>>.

Acknowledgements

This draft has mostly been generated from [[RFC7464](#)] by Nico Williams and [[RFC8091](#)] by Erik Wilde, which do a similar, but slightly more complicated exercise for JSON [[RFC8259](#)]. Laurence Lundblade raised an issue on the CBOR mailing list that pointed out the need for this document. Jim Schaad and John Mattsson provided helpful comments.

Author's Address

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

