Workgroup: Network Working Group
Internet-Draft:
draft-bormann-cbor-cddl-more-control-01
Published: 9 March 2023
Intended Status: Standards Track
Expires: 10 September 2023
Authors: C. Bormann
Universität Bremen TZI

More Control Operators for CDDL

Abstract

The Concise Data Definition Language (CDDL), standardized in RFC 8610, provides "control operators" as its main language extension point. RFCs have added to this extension point both in an application-specific and a more general way.

The present document defines a number of additional generally application control operators for text conversion (Bytes, Integers, JSON), operations on text, and deterministic encoding.

Revision -01 of this draft reflects comments from initial discussion of the specification in the CBOR working group. It is intended to be ready for working group adoption.

About This Document

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

The latest revision of this draft can be found at https://cbor-wg.github.io/cddl-more-control/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-bormann-cbor-cddl-more-control/.

Discussion of this document takes place on the Concise Binary Object Representation (CBOR) Maintenance and Extensions 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/cbor-wg/cddl-more-control.

Status of This Memo

This Internet-Draft will expire on 10 September 2023.

Copyright Notice

Table of Contents

1. Introduction

The Concise Data Definition Language (CDDL), standardized in [RFC8610], provides "control operators" as its main language extension point (Section 3.8 of [RFC8610]). RFCs have added to this extension point both in an application-specific [RFC9090] and a more general [RFC9165] way.

The present document defines a number of additional generally applicable control operators:

Name	Purpose
.b64u, .b64c	Base64 representation of byte strings
.b64u-sloppy, .b64c-sloppy	(sloppy-tolerant variants of the above)
.hex, .hexlc, .hexuc	Base16 representation of byte strings
.b32, .h32	Base32 representation of byte strings
.b45	Base45 representation of byte strings
.decimal	Text representation of integer numbers
.json	Text representation of JSON values
.join	Building text from array of components
.cbordet, .cborseqdet	deterministically encoded CBOR data items, CBOR sequences

Table 1: New control operators in this document

1.1. Terminology

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.

This specification uses terminology from [RFC8610]. In particular, with respect to control operators, "target" refers to the left-hand side operand, and "controller" to the right-hand side operand. "Tool" refers to tools along the lines of that described in Appendix F of [RFC8610]. Note also that the data model underlying CDDL provides for text strings as well as byte strings as two separate types, which are then collectively referred to as "strings".

2. Text Conversion

2.1. Byte Strings: Base16 (Hex), Base32, Base64

A CDDL model often defines data that are byte strings in essence but need to be transported in various encoded forms, such as base64 or hex. This section defines a number of control operators to model these conversions.

The control operators generally are of a form that could be used like this:

```
signature-for-json = text .b64u signature
signature = bytes .cbor COSE_Sign1
```

The specification of these control operators is complicated by the large number of transformations in use. Inspired by Section 8 of [STD94], we use representations defined in [RFC4648] with the following names:

name	meaning	reference
.b64u	Base64URL, no padding	Section 5 of [RFC4648]
.b64u-sloppy	Base64URL, no padding, sloppy	Section 5 of [RFC4648]
.b64c	Base64 classic, padding	Section 4 of [RFC4648]
.b64c-sloppy	Base64 classic, padding, sloppy	Section 4 of [RFC4648]
.b32	Base32, no padding	Section 6 of [RFC4648]
.h32	Base32/hex alphabet, no padding	Section 7 of [RFC4648]
.hex	Base16 (hex), either case	Section 8 of [RFC4648]
.hexlc	Base16 (hex), lower case	Section 8 of [RFC4648]
.hexuc	Base16 (hex), upper case	Section 8 of [RFC4648]
.b45	Base45	[RFC9285]

Table 2: Control Operators for Text Conversion of byte strings

Note that this specification is somewhat opinionated here: It does not provide base64url, base32 or base32hex encoding with padding, or base64 classic without padding. Experience indicates that these combinations only ever occur in error, so the usability of CDDL is increased by not providing them in the first place. Also, adding "c" makes sure that any decision for classic base64 is actively taken.

The additional designation "sloppy" indicates that the text string is not validated for any additional bits being zero, in variance to what is specified in the paragraph behind table 1 in Section 4 of [RFC4648]. Note that the present specification is opinionated again in not specifying a sloppy variant of base32 or base32/hex, as no legacy use of sloppy base32(/hex) was known at the time of writing. Base45 is known to be suboptimal for use in environments with limited data transparency (such as URLs), but is included because of its close relationship to QR codes and its wide use in health informatics (note that base45 is at least strongly specified not to allow sloppy forms of encoding).

2.2. Numbers

name	meaning	reference
.decimal	Decimal Integer	---

Table 3: Control Operator for Text Conversion of Integers

This allows the modeling of text strings that carry numeric information, such as in the uint64/int64 formats of YANG-JSON [RFC7951].

```
yang-json-sid = text .decimal (0..9223372036854775807)
```

Again, the specification is opinionated by only providing numbers without leading zeros, i.e., the decimal numbers match the regular expression "0|-?[1-9][0-9]*" (of course, further restricted by the control type). Future specifications can provide octal, hexadecimal, or binary conversions.

2.3. JSON Values

Some applications store complete JSON texts into text strings, the JSON value for which can easily be defined in CDDL. This is supported by a control operator similar to .cbor in Section 3.8.4 of [RFC8610].

name	meaning	reference
.json	JSON	[STD90]

Table 4: Control Operator for Text Conversion of JSON values

```
embedded-claims = text .json claims
claims = {iss: issuer, exp: expiry}
```

Note that a .jsonseq is not provided, as no use case is known yet. There is no way to constrain the use of blank space in data items to be validated; variants (e.g, not providing for any blank space) could be defined.

3. Text Processing

3.1. Join

Often, text strings need to be constructed out of parts that can best be modeled as an array.

name	meaning	reference
.join	concatenate elements of an array	---

Table 5: Control Operator for Text Generation from Arrays

In general, this control operator is hard to validate as it would require full parser functionality. It is therefore recommended to only use it in simple cases, and leave full parsing to ABNF Section 3 of [RFC9165] or similar.

```
legacy-ip-address = text .join [digits<1>, ".", digits<2>,
                           ".", digits<3>, ".", digits<4>]
digits<N> = text .decimal byte<n>
```

4. Deterministic Encoding

[RFC8610] and [RFC8742] specify the control operators .cbor and .cborseq to indicate that the value of a byte string should be an encoded CBOR data item or a CBOR sequence.

This specification provides complementary control operators .cbordet and .cborseqdet that indicate that these data items/sequences need to be encoded in accordance to Sections 4.2.1 and 4.2.2 of [STD94].

name	meaning	reference
.cbordet	deterministically encoded CBOR data item	[RFC8610]
.cborseqdet	CBOR sequence made from deterministically encoded CBOR data items	[RFC8742]

Table 6: Control Operator for Deterministically Encoded Data Items and Sequences

Note that considerations of deterministic representation at the application level can often be expressed in the CDDL definition of the right-hand side and then do not need additional control operators.

5. IANA Considerations

This document requests IANA to register the contents of Table 7 into the registry "CDDL Control Operators" of [IANA.cddl]:

Name	Reference
.b64u	[RFCthis]
.b64u-sloppy	[RFCthis]
.b64c	[RFCthis]
.b64c-sloppy	[RFCthis]
.b45	[RFCthis]
.b32	[RFCthis]
.h32	[RFCthis]

Name	Reference
.hex	[RFCthis]
.hexlc	[RFCthis]
.hexuc	[RFCthis]
.decimal	[RFCthis]
.json	[RFCthis]
.join	[RFCthis]
.cbordet	[RFCthis]
.cborseqdet	[RFCthis]

Table 7: New control operators to be registered

6. Implementation Status

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

In the CDDL tool described in Appendix F of [RFC8610], the control operators defined in revision -00 of this specification are implemented as of version 0.10.2; implementation of the rest is ongoing.

7. Security considerations

The security considerations of [RFC8610] apply.

8. References

8.1. Normative References

[IANA.cddl] IANA, "Concise Data Definition Language (CDDL)", <https://www.iana.org/assignments/cddl>.

[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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[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>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/rfc/rfc8742>.

[RFC9165] Bormann, C., "Additional Control Operators for the Concise Data Definition Language (CDDL)", RFC 9165, DOI 10.17487/RFC9165, December 2021, <https://www.rfc-editor.org/rfc/rfc9165>.

[RFC9285] Fältström, P., Ljunggren, F., and D.W. van Gulik, "The Base45 Data Encoding", RFC 9285, DOI 10.17487/RFC9285, August 2022, <https://www.rfc-editor.org/rfc/rfc9285>.

[STD90] 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/rfc/rfc8259>.

[STD94] 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>.

8.2. Informative References

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/rfc/rfc7951>.

[RFC9090] Bormann, C., "Concise Binary Object Representation (CBOR) Tags for Object Identifiers", RFC 9090, DOI 10.17487/RFC9090, July 2021, <https://www.rfc-editor.org/rfc/rfc9090>.

Acknowledgements

Henk Birkholz suggested the need for many of the control operators defined here.

Author's Address

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

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