Workgroup: CBOR
Internet-Draft:
draft-bormann-cbor-update-8610-grammar-00
Updates: 8610 (if approved)
Published: 9 March 2023
Intended Status: Standards Track
Expires: 10 September 2023
Authors: C. Bormann
Universität Bremen TZI

Updates to the CDDL grammar of RFC 8610

Abstract

At the time of writing, the Concise Data Definition Language (CDDL) is defined by RFC 8610 and RFC 9165. The latter has used the extension point provided in RFC 8610, the *control operator*.

As CDDL is being used in larger projects, the need for corrections and additional features has become known that cannot be easily mapped into this single extension point. Hence, there is a need for evolution of the base CDDL specification itself.

The present document updates errata and makes other small fixes for the ABNF grammar defined for CDDL in RFC 8610.

Previous versions of the changes in this document were part of draft-bormann-cbor-cddl-2-draft and previously draft-bormann-cbor-cddl-freezer. This submission extracts out those grammar changes that are ready for WG adoption and publication.

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/update-8610-grammar/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-bormann-cbor-update-8610-grammar/.

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/cbor-wg/update-8610-grammar.

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 10 September 2023.

Copyright Notice

Copyright (c) 2023 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

At the time of writing, the Concise Data Definition Language (CDDL) is defined by RFC 8610 and RFC 9165. The latter has used the extension point provided in RFC 8610, the *control operator*.

As CDDL is being used in larger projects, the need for corrections and additional features has become known that cannot be easily mapped into this single extension point. Hence, there is a need for evolution of the base CDDL specification itself.

The present document updates errata and makes other small fixes for the ABNF grammar defined for CDDL in RFC 8610.

Previous versions of the changes in this document were part of draft-bormann-cbor-cddl-2-draft and previously draft-bormann-cbor-cddl-freezer. This submission extracts out those grammar changes that are ready for WG adoption and publication.

Proposals for grammar and other changes that need more work can be found in [I-D.bormann-cbor-cddl-2-draft]. Proposals for other additions to the CDDL specification base are in [I-D.draft-bormann-cbor-cddl-freezer].

Note that the existing extension point "control operator" (Section 3.8 of [RFC8610]) can be exercised for new features in parallel to the work described here; one set of such proposals is in [I-D.bormann-cbor-cddl-more-control].

The present document, in conjunction with [RFC8610] as well as [RFC9165] and [I-D.bormann-cbor-cddl-more-control], is intended to be the specification base of what has colloquially been called CDDL 1.1. Additional documents describe further work on CDDL.

1.1. Conventions and Definitions

The Terminology from [RFC8610] applies. The grammar in [RFC8610] is based on ABNF, which is defined in [RFC5234] and [RFC7405].

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.

2. Clarifications and Changes based on Errata Reports

Compatibility:

errata fix (targets 1.0 and 2.0)

A number of errata reports have been made around some details of text string and byte string literal syntax: [Err6527] and [Err6543]. These are being addressed in this section, updating details of the ABNF for these literal syntaxes. Also, [Err6526] needs to be applied (backslashes have been lost during RFC processing in some text explaining backslash escaping).

2.1. Err6527 (text string literals)

The ABNF used in [RFC8610] for the content of text string literals is rather permissive:

```
; RFC 8610 ABNF:
text = %x22 *SCHAR %x22
SCHAR = %x20-21 / %x23-5B / %x5D-7E / %x80-10FFFD / SESC
SESC = "\" (%x20-7E / %x80-10FFFD)
```

This allows almost any non-C0 character to be escaped by a backslash, but critically misses out on the \uXXXX and \uHHHH\uLLLL forms that JSON allows to specify characters in hex (which should be applying here according to Bullet 6 of Section 3.1 of [RFC8610]). Both can be solved by updating the SESC production to:

```
; new rules collectively defining SESC:
SESC = "\" ( %x22 / "/" / "\" /                 ; \" \/ \\
             %x62 / %x66 / %x6E / %x72 / %x74 / ; \b \f \n \r \t
             (%x75 hexchar) )                   ; \uXXXX
hexchar = non-surrogate / (high-surrogate "\" %x75 low-surrogate)
non-surrogate = ((DIGIT / "A"/"B"/"C" / "E"/"F") 3HEXDIG) /
                ("D" %x30-37 2HEXDIG )
high-surrogate = "D" ("8"/"9"/"A"/"B") 2HEXDIG
low-surrogate = "D" ("C"/"D"/"E"/"F") 2HEXDIG
```

Figure 1: Updated string parsing to allow hex escapes

(Notes: In ABNF, strings such as "A", "B" etc. are case-insensitive, as is intended here. We could have written %x62 as %s"b", but didn't, in order to maximize ABNF tool compatibility.)

Now that SESC is more restrictively formulated, this also requires an update to the BCHAR production used in the ABNF syntax for byte string literals:

```
; RFC 8610 ABNF:
bytes = [bsqual] %x27 *BCHAR %x27
BCHAR = %x20-26 / %x28-5B / %x5D-10FFFD / SESC / CRLF
bsqual = "h" / "b64"
```

In BCHAR, the updated version explicitly allows \', which is no longer allowed in the updated SESC:

```
; new rule for BCHAR:
BCHAR = %x20-26 / %x28-5B / %x5D-10FFFD / SESC / "\'" / CRLF
```

Figure 2: Updated rule for BCHAR

2.2. Err6543 (byte string literals)

The ABNF used in [RFC8610] for the content of byte string literals lumps together byte strings notated as text with byte strings notated in base16 (hex) or base64 (but see also updated BCHAR production above):

```
; RFC 8610 ABNF:
bytes = [bsqual] %x27 *BCHAR %x27
BCHAR = %x20-26 / %x28-5B / %x5D-10FFFD / SESC / CRLF
```

Change proposed by Errata Report 6543

Errata report 6543 proposes to handle the two cases in separate productions (where, with an updated SESC, BCHAR obviously needs to be updated as above):

```
; Err6543 proposal:
bytes = %x27 *BCHAR %x27
      / bsqual %x27 *QCHAR %x27
BCHAR = %x20-26 / %x28-5B / %x5D-10FFFD / SESC / CRLF
QCHAR = DIGIT / ALPHA / "+" / "/" / "-" / "_" / "=" / WS
```

Figure 3: Errata Report 8653 Proposal to Split the Byte String Rules

This potentially causes a subtle change, which is hidden in the WS production:

```
; RFC 8610 ABNF:
WS = SP / NL
SP = %x20
NL = COMMENT / CRLF
COMMENT = ";" *PCHAR CRLF
PCHAR = %x20-7E / %x80-10FFFD
CRLF = %x0A / %x0D.0A
```

Figure 4: Definition of WS from RFC 8610

This allows any non-C0 character in a comment, so this fragment becomes possible:

```
foo = h'
   43424F52 ; 'CBOR'
   0A       ; LF, but don't use CR!
'
```

The current text is not unambiguously saying whether the three apostrophes need to be escaped with a \ or not, as in:

```
foo = h'
   43424F52 ; \'CBOR\'
   0A       ; LF, but don\'t use CR!
'
```

... which would be supported by the existing ABNF in [RFC8610].

No change needed after *[Section 2.1]*

This document takes the simpler approach of leaving the processing of the content of the byte string literal to a semantic step after processing the syntax of the bytes/BCHAR rules as updated by Figure 1 and Figure 2.

The rules in Figure 4 are therefore applied to the result of this processing where bsqual is given as h or b64.

Note that this approach also works well with the use of byte strings in Section 3 of [RFC9165]. It does require some care when copy-pasting into CDDL models from ABNF that contains single quotes (which may also hide as apostrophes in comments); these need to be escaped or possibly replaced by %x27.

Finally, our approach may leave the door open wider to extending bsqual as proposed in Appendix A.1 of [I-D.bormann-cbor-cddl-2-draft].

3. Small Enabling Grammar Changes

The two subsections in this section specify two small changes to the grammar that are intended to enable certain kinds of specifications.

3.1. Empty data models

Compatibility: backward (not forward)

[RFC8610] requires a CDDL file to have at least one rule.

```
; RFC 8610 ABNF:
cddl = S 1*(rule S)
```

This makes sense when the file has to stand alone, as a CDDL data model needs to have at least one rule to provide an entry point (start rule).

With CDDL 2.0, CDDL files can also include directives, and these might be the source of all the rules that ultimately make up the module created by the file. Any other rule content in the file has to be available for directive processing, making the requirement for at least one rule cumbersome.

Therefore, we extend the grammar as follows:

```
; new top-level rule:
cddl = S *(rule S)
```

and make the existence of at least one rule a semantic constraint, to be fulfilled after processing of all directives.

3.2. Non-literal Tag Numbers

Compatibility: backward (not forward)

The CDDL 1.0 syntax for expressing tags in CDDL is (ABNF as in [RFC5234]):

```
; extracted from RFC 8610 ABNF:
type2 /= "#" "6" ["." uint] "(" S type S ")"
```

This means tag numbers can only be given as literal numbers (uints). Some specifications operate on ranges of tag numbers, e.g., [RFC9277] has a range of tag numbers 1668546817 (0x63740101) to 1668612095 (0x6374FFFF) to tag specific content formats. This can currently not be expressed in CDDL.

CDDL 2.0 extends this to:

```
; new rules collectively defining the tagged case:
type2 /= "#" "6" ["." tag-number] "(" S type S ")"
tag-number = uint / ("<" type ">")
```

So the above range can be expressed in a CDDL fragment such as:

```
ct-tag<content> = #6.<ct-tag-number>(content)
ct-tag-number = 1668546817..1668612095
; or use 0x63740101..0x6374FFFF
```

Note that this syntax reuses the angle bracket syntax for generics; this reuse is innocuous as a generic parameter/argument only ever occurs after a rule name (id), while it occurs after . here. (Whether there is potential for human confusion can be debated; the above example deliberately uses generics as well.)

4. Security Considerations

The grammar fixes and updates in this document are not believed to create additional security considerations. The security considerations in Section 5 of [RFC8610] do apply, and specifically the potential for confusion is increased in an environment that uses a combination of CDDL tools some of which have been updated and some of which have not been, in particular based on Section 2.

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. 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>.

[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>.

6.2. Informative References

[Err6526] "Errata Report 6526", RFC 8610, <https://www.rfc-editor.org/errata/eid6526>.

[Err6527] "Errata Report 6527", RFC 8610, <https://www.rfc-editor.org/errata/eid6527>.

[Err6543] "Errata Report 6543", RFC 8610, <https://www.rfc-editor.org/errata/eid6543>.

[I-D.bormann-cbor-cddl-2-draft]
Bormann, C., "CDDL 2.0 -- a draft plan", Work in Progress, Internet-Draft, draft-bormann-cbor-cddl-2-draft-01, 5 March 2023, <https://datatracker.ietf.org/doc/html/draft-bormann-cbor-cddl-2-draft-01>.

[I-D.bormann-cbor-cddl-more-control]
Bormann, C., "More Control Operators for CDDL", Work in Progress, Internet-Draft, draft-bormann-cbor-cddl-more-control-00, 25 February 2023, <https://datatracker.ietf.org/doc/html/draft-bormann-cbor-cddl-more-control-00>.

[I-D.draft-bormann-cbor-cddl-freezer]
Bormann, C., "A feature freezer for the Concise Data Definition Language (CDDL)", Work in Progress, Internet-Draft, draft-bormann-cbor-cddl-freezer-10, 24 October 2022, <https://datatracker.ietf.org/doc/html/draft-bormann-cbor-cddl-freezer-10>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <https://www.rfc-editor.org/rfc/rfc7405>.

[RFC9277] Richardson, M. and C. Bormann, "On Stable Storage for Items in Concise Binary Object Representation (CBOR)", RFC 9277, DOI 10.17487/RFC9277, August 2022, <https://www.rfc-editor.org/rfc/rfc9277>.

Appendix A. Updated Collected ABNF for CDDL

This appendix provides the full ABNF from [RFC8610] with the updates applied in the present document.

```
cddl = S *(rule S)
rule = typename [genericparm] S assignt S type
     / groupname [genericparm] S assigng S grpent

typename = id
groupname = id

assignt = "=" / "/="
assigng = "=" / "//="

genericparm = "<" S id S *("," S id S ) ">"
genericarg = "<" S type1 S *("," S type1 S ) ">"

type = type1 *(S "/" S type1)

type1 = type2 [S (rangeop / ctlop) S type2]
; space may be needed before the operator if type2 ends in a name

type2 = value
      / typename [genericarg]
      / "(" S type S ")"
      / "{" S group S "}"
      / "[" S group S "]"
      / "~" S typename [genericarg]
      / "&" S "(" S group S ")"
      / "&" S groupname [genericarg]
      / "#" "6" ["." tag-number] "(" S type S ")"
      / "#" DIGIT ["." uint]                ; major/ai
      / "#"                                 ; any
tag-number = uint / ("<" type ">")


rangeop = "..." / ".."

ctlop = "." id

group = grpchoice *(S "//" S grpchoice)

grpchoice = *(grpent optcom)

grpent = [occur S] [memberkey S] type
       / [occur S] groupname [genericarg]  ; preempted by above
       / [occur S] "(" S group S ")"

memberkey = type1 S ["^" S] "=>"
          / bareword S ":"
          / value S ":"

bareword = id
```

```
optcom = S ["," S]

occur = [uint] "*" [uint]
      / "+"
      / "?"

uint = DIGIT1 *DIGIT
     / "0x" 1*HEXDIG
     / "0b" 1*BINDIG
     / "0"

value = number
      / text
      / bytes

int = ["-"] uint

; This is a float if it has fraction or exponent; int otherwise
number = hexfloat / (int ["." fraction] ["e" exponent ])
hexfloat = ["-"] "0x" 1*HEXDIG ["." 1*HEXDIG] "p" exponent
fraction = 1*DIGIT
exponent = ["+"/"-"] 1*DIGIT

text = %x22 *SCHAR %x22
SCHAR = %x20-21 / %x23-5B / %x5D-7E / %x80-10FFFD / SESC

SESC = "\" ( %x22 / "/" / "\" /                 ; \" \/ \\
             %x62 / %x66 / %x6E / %x72 / %x74 / ; \b \f \n \r \t
             (%x75 hexchar) )                   ; \uXXXX
hexchar = non-surrogate / (high-surrogate "\" %x75 low-surrogate)
non-surrogate = ((DIGIT / "A"/"B"/"C" / "E"/"F") 3HEXDIG) /
                ("D" %x30-37 2HEXDIG )
high-surrogate = "D" ("8"/"9"/"A"/"B") 2HEXDIG
low-surrogate = "D" ("C"/"D"/"E"/"F") 2HEXDIG

bytes = [bsqual] %x27 *BCHAR %x27
BCHAR = %x20-26 / %x28-5B / %x5D-10FFFD / SESC / "\'" / CRLF
bsqual = "h" / "b64"

id = EALPHA *(*("-" / ".") (EALPHA / DIGIT))
ALPHA = %x41-5A / %x61-7A
EALPHA = ALPHA / "@" / "_" / "$"
DIGIT = %x30-39
DIGIT1 = %x31-39
HEXDIG = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
BINDIG = %x30-31

S = *WS
WS = SP / NL
SP = %x20
```

```
NL = COMMENT / CRLF
COMMENT = ";" *PCHAR CRLF
PCHAR = %x20-7E / %x80-10FFFD
CRLF = %x0A / %x0D.0A
```

Figure 5: ABNF for CDDL as updated

Acknowledgments

TODO acknowledge.

Author's Address

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

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