Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 10, 2020

C. Bormann
Universitaet Bremen TZI

March 09, 2020

A feature freezer for the Concise Data Definition Language (CDDL)
draft-bormann-cbor-cddl-freezer-03

Abstract

In defining the Concise Data Definition Language (CDDL), some features have turned up that would be nice to have. In the interest of completing this specification in a timely manner, the present document was started to collect nice-to-have features that did not make it into the first RFC for CDDL, RFC 8610.

It is now time to discuss thawing some of the concepts discussed here. A number of additional proposals have been added.

Status of This Memo

This Internet-Draft will expire on September 10, 2020.

Copyright Notice

Table of Contents

1. Introduction

(TO DO: Insert an extended form of the abstract first here, expanding the reference to [RFC8610].)

There is always a danger for a document like this to become a shopping list; the intention is to develop this document further based on real-world experience with the first CDDL standard.

2. Base language features

2.1. Cuts

Section 3.5.4 of [RFC8610] alludes to a new language feature, _cuts_, and defines it in a fashion that is rather focused on a single application in the context of maps and generating better diagnostic information about them.

The present document is expected to grow a more complete definition of cuts, with the expectation that it will be upwards-compatible to the existing one in [RFC8610], before this possibly becomes a mainline language feature in a future version of CDDL.

3. Literal syntax

3.1. Computed Literals

CDDL cannot compute.

This is a liability in at least two situations:

- integers often need to be defined relative to an offset. It might be desirable to be able to write something like:

```
base = 16777216
a = base + 1
b = base + 2
```

- some string literals (in particular, complex regular expressions) would best be composed from components. This could be done by adding a concatenation operator (maybe even "+" as used for addition above), or by adding string interpolation to the string literal syntax.

3.2. Tag-oriented Literals

Some CBOR tags often would be most natural to use in a CDDL spec with a literal syntax that is tailored to their semantics instead of their serialization in CBOR. There is currently no way to add such syntaxes, no defined extension point either.

The text form of CoRAL [I-D.hartke-t2trg-coral] defines literals of the form

```
dt'2019-07-21T19:53Z'
```

for datetime items. (Similar advances should then probably be made in diagnostic notation.)

3.3. Regular Expression Literals

Regular expressions currently are notated as strings in CDDL, with all the string escaping rules applied once. It might be convenient to have a more conventional literal format for regular expressions, possibly also providing a place to add modifiers such as "/i". This might also imply "text .regexp ...", which with the proposal in

Section 5.1 then raises the question of how to indicate the regular expression flavor.

4. Embedded ABNF

It would often be desirable to define a text string type by employing ABNF [RFC5234] [RFC7405] embedded into the CDDL specification. Currently, that ABNF would usually need to be translated into a regular expression (if that is even possible).

Note that some applications of computed literals for strings could be covered by such a feature (or partially vice versa).

4.1. A feeble attempt at adding ABNF to CDDL

ABNF can easily be added in the same way that regular expressions were added, e.g., by defining a ".abnf" control operator.

There are several small issues, with solutions proposed here:

- ABNF can be used to define byte sequences as well as UTF-8 text strings interpreted as Unicode scalar sequences . This means there are two controls needed, e.g. ".abnfb" for byte sequences and ".abnf" for UTF-8 text strings.

- ABNF defines a list of rules, not a single expression (called "elements" in [RFC5234]). This could be resolved by requiring the control string to be one "element", followed by zero or more "rule".

- For the same reason, ABNF requires newlines; specifying newlines in CDDL text strings is tedious (and leads to essentially unreadable ABNF). One workaround might be to allow the specification of byte strings as control strings, interpreted as UTF-8; the syntax for text in byte strings is more flexible in CDDL than for text strings.

- One set of rules provided in an ABNF specification is often used in multiple positions, in particular staples such as DIGIT and ALPHA. This calls for some form of composition, e.g. by providing a .cat control operator (see also Section 3.1 above); we are glossing over some minor data typing issues here; these are again needed in byte sequence and text sequence forms. (A ".join" as in Python might be more versatile.)

These points, combined into an example:

```
; for draft-jones-cbor-date-tag-00
Tag1004 = #6.1004(text .abnf full-date)
; for RFC 7049
Tag0 = #6.0(text .abnf date-time)

full-date = "full-date" .cat rfc3339
date-time = "date-time" .cat rfc3339

; Note the trick of idiomatically starting with a newline, separating
;   off the element in the .cat from the rule-list
rfc3339 = '
   date-fullyear   = 4DIGIT
   date-month      = 2DIGIT  ; 01-12
   date-mday       = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31 based on
                             ; month/year
   time-hour       = 2DIGIT  ; 00-23
   time-minute     = 2DIGIT  ; 00-59
   time-second     = 2DIGIT  ; 00-58, 00-59, 00-60 based on leap sec
                             ; rules
   time-secfrac    = "." 1*DIGIT
   time-numoffset  = ("+" / "-") time-hour ":" time-minute
   time-offset     = "Z" / time-numoffset

   partial-time    = time-hour ":" time-minute ":" time-second
                     [time-secfrac]
   full-date       = date-fullyear "-" date-month "-" date-mday
   full-time       = partial-time time-offset

   date-time       = full-date "T" full-time
' .cat rfc5234-core

rfc5234-core = '
         DIGIT          =  %x30-39 ; 0-9
; abbreviated here
'
```

5. Controls

Controls are the main extension point of the CDDL language. It is relatively painless to add controls to CDDL. Several candidates have been identified that aren't quite ready for adoption, of which one shall be listed here.

5.1. Control operator .pcre

There are many variants of regular expression languages. Section 3.8.3 of [RFC8610] defines the .regexp control, which is based on XSD [XSD2] regular expressions. As discussed in that section, the most desirable form of regular expressions in many cases is the family called "Perl-Compatible Regular Expressions" ([PCRE]); however, no formally stable definition of PCRE is available at this time for normatively referencing it from an RFC.

The present document defines the control operator .pcre, which is similar to .regexp, but uses PCRE2 regular expressions. More specifically, a ".pcre" control indicates that the text string given as a target needs to match the PCRE regular expression given as a value in the control type, where that regular expression is anchored on both sides. (If anchoring is not desired for a side, ".*" needs to be inserted there.)

Similarly, ".es2018re" could be defined for ECMAscript 2018 regular expressions with anchors added.

5.2. Endianness in .bits

How useful would it be to have another variant of .bits that counts bits like in RFC box notation? (Or at least per-byte? 32-bit words don't always perfectly mesh with byte strings.)

5.3. .bitfield control

Provide a way to specify bitfields in byte strings and uints to a higher level of detail than is possible with .bits. Strawman:

```
Field = uint .bitfield Fieldbits

Fieldbits = [
  flag1: [1, bool],
  val: [4, Vals],
  flag2: [1, bool],
]

Vals = &(A: 0, B: 1, C: 2, D: 3)
```

Note that the group within the controlling array can have choices, enabling the whole power of a context-free grammar (but not much more).

6. Co-occurrence Constraints

While there are no co-occurrence constraints in CDDL, many actual use cases can be addressed by using the fact that a group is a grammar:

```
postal = {
  ( street: text,
    housenumber: text) //
  ( pobox: text .regexp "[0-9]+" )
}
```

However, constraints that are not just structural/tree-based but are predicates combining parts of the structure cannot be expressed:

```
session = {
  timeout: uint,
}

other-session = {
  timeout: uint  .lt [somehow refer to session.timeout],
}
```

As a minimum, this requires the ability to reach over to other parts of the tree in a control. Compare JSON Pointer [RFC6901] and JSON Relative Pointer [I-D.handrews-relative-json-pointer]. Stefan Goessner's jsonpath is a JSON variant of XPath that has not been formally standardized [jsonpath].

More generally, something akin to what Schematron is to Relax-NG may be needed.

7. Module superstructure

CDDL rules could be packaged as modules and referenced from other modules. There could be some control of namespace pollution, as well as unambiguous referencing ("versioning").

This is probably best achieved by a pragma-like syntax which could be carried in CDDL comments, leaving each module to be valid CDDL (if missing some rule definitions to be imported).

7.1. Namespacing

A convention for mapping CDDL-internal names to external ones could be developed, possibly steered by some pragma-like constructs. External names would likely be URI-based, with some conventions as they are used in RDF or Curies. Internal names might look similar to XML QNames. Note that the identifier character set for CDDL

deliberately includes $ and @, which could be used in such a convention.

8. Alternative Representations

For CDDL, alternative representations e.g. in JSON (and thus in YAML) could be defined, similar to the way YANG defines an XML-based serialization called YIN in Section 11 of [RFC6020]. One proposal for such a syntax is provided by the "cddlc" tool [cddlc]; this could be written up and agreed upon.

```
cddlj = ["cddl", +rule]
rule = ["=" / "/=" / "//=", namep, type]
namep = ["name", id] / ["gen", id, +id]
id = text .regexp "[A-Za-z@_$](([-.])*[A-Za-z0-9@_$])*"
op = ".." / "..." /
  text .regexp "\\.[A-Za-z@_$](([-.])*[A-Za-z0-9@_$])*"
namea = ["name", id] / ["gen", id, +type]
type = value / namea / ["op", op, type, type] /
  ["map", group] / ["ary", group] / ["tcho", 2*type] /
  ["unwrap", namea] / ["enum", group / namea] /
  ["prim", ?(0..7, ?uint)]
group = ["mem", null/type, type] /
  ["rep", uint, uint/false, group] /
  ["seq", 2*group] / ["gcho", 2*group]
value = ["number"/"text"/"bytes", text]
```

9. IANA Considerations

This document makes no requests of IANA.

10. Security considerations

The security considerations of [RFC8610] apply.

11. References

11.1. Normative References

[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>.

11.2. Informative References

[cddlc] "CDDL conversion utilities", n.d., <https://github.com/cabo/cddlc>.

[I-D.handrews-relative-json-pointer] Luff, G. and H. Andrews, "Relative JSON Pointers", draft-handrews-relative-json-pointer-02 (work in progress), September 2019.

[I-D.hartke-t2trg-coral] Hartke, K., "The Constrained RESTful Application Language (CoRAL)", draft-hartke-t2trg-coral-09 (work in progress), July 2019.

[jsonpath] "jsonpath online evaluator", n.d., <https://jsonpath.com>.

[PCRE] "Perl-compatible Regular Expressions (revised API: PCRE2)", n.d., <http://pcre.org/current/doc/html/>.

[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/info/rfc5234>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <https://www.rfc-editor.org/info/rfc6901>.

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

[XSD2] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

Acknowledgements

Many people have asked for CDDL to be completed, soon. These are usually also the people who have brought up observations that led to the proposals discussed here. Sean Leonard has campaigned for a regexp literal syntax.

Author's Address

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

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