Network Working Group
Internet-Draft
Intended status: Informational
Expires: 26 August 2026

C. Bormann
Universit辰t Bremen TZI

22 February 2026

CDDL models for some existing RFCs
draft-bormann-cbor-rfc-cddl-models-07

Abstract

A number of CBOR- and JSON-based protocols have been defined before CDDL was standardized or widely used.

This short draft records some CDDL definitions for such protocols, which could become part of a library of CDDL definitions available for use in CDDL2 processors. It focuses on CDDL in (almost) published IETF RFCs.

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-bormann-cbor-rfc-cddl-models/.

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

Source for this draft and an issue tracker can be found at https://github.com/cabo/common-cddl.

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 26 August 2026.

Copyright Notice

Copyright (c) 2026 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

(Please see abstract.) Add in [STD94] [STD90] [RFC8610] [RFC9682] [RFC9165] [RFC9741]

2. CDDL definitions for (almost) published RFCs

This section is intended to have one subsection for each CDDL data model presented for an existing RFC. As a start, it is fleshed out with three such data models.

2.1. RFC 7071 (Reputation Interchange)

Appendix H of [RFC8610] contains two CDDL definitions for [RFC7071], which are not copied here. Typically, the compact form would be used in applications using the RFC 7071 format; while the extended form might be useful to cherry-pick features of RFC 7071 into another protocol.

2.2. RFC 8366 (Voucher Artifact for Bootstrapping Protocols)

[RFC8366] defines a data model for a "Voucher Artifact", which can be represented in CDDL as:

```
voucher-artifact = {
  "ietf-voucher:voucher": {
    created-on: yang$date-and-time
    ? (
        expires-on: yang$date-and-time
        ? last-renewal-date: yang$date-and-time
        //
        nonce: json-binary<bytes .size (8..32)>
      )
    assertion: assertion
    serial-number: text
    ? idevid-issuer: json-binary<bytes>
    pinned-domain-cert: json-binary<bytes>
    ? domain-cert-revocation-checks: bool
  }
}

assertion = "verified" / "logged" / "proximity"

yang$date-and-time = text .regexp cat3<"[0-9]{4}-[0-9]{2}-[0-9]{2}T",
                                      "[0-9]{2}:[0-9]{2}:[0-9]{2}([.][0-9]+)?",
                                      "(Z|[+-][0-9]{2}:[0-9]{2})">

cat3<A,B,C> = (A .cat B) .cat C

json-binary<T> = text .b64c T
```

Figure 1: CDDL for RFC 8366

The two examples in the RFC can be validated with this little patchup script:

```
sed -e s/ue=/uQ=/ -e s/’"true"’/true/ | cddl rfc8366.cddl vp -
```

2.3. RFC 9457 (Problem Details for HTTP APIs)

[RFC9457] defines a simple data model that is reproduced in CDDL here:

```
problem-object = {
  ? type: preferably-absolute-uri
  ? title: text
  ? status: 100..599
  ? detail: text
  ? instance: preferably-absolute-uri
  * text .feature "problem-object-extension" => any
}

; RECOMMENDED: absolute URI or at least absolute path:
preferably-absolute-uri = ~uri
```

Figure 2: CDDL for RFC 9457

Note that Appendix B of [RFC9290] defines a related CBOR-specific data model that may be useful for tunneling [RFC7807] or [RFC9457] problem details in [RFC9290] Concise Problem Details.

2.4. RFC 9595 (YANG-SID)

[RFC9595] defines a data model for a "SID file" in YANG, to be transported as a YANG-JSON instance.

An equivalent CDDL data model is given here:

```
sid-file = {
  "ietf-sid-file:sid-file": {
    module-name: yang$yang-identifier
    ? module-revision: revision-identifier
    ? sid-file-version: sid-file-version-identifier
    ? sid-file-status: "unpublished" / "published"
    ? description: text
    ? dependency-revision: [* dependency-revision]
    ? assignment-range: [* assignment-range]
    ? item: [*item]
  }
}
```

```
rep<RE>=cat3<"(", RE, ")*">
opt<RE>=cat3<"(", RE, ")?">
cat3<A,B,C> = (A .cat B) .cat C

id-re = "[a-zA-Z_][a-zA-Z0-9\\-_.]*"
yang$yang-identifier = text .regexp id-re
revision-identifier = text .regexp "[0-9]{4}-[0-9]{2}-[0-9]{2}"
sid-file-version-identifier = uint .size 4
sid = text .decimal (0..0x7fffffffffffffff); uint63 as text string
plus-id<Prefix> = Prefix .cat id-re
schema-node-re = cat3<plus-id<"/">, plus-id<":">, ; qualified
                      rep<plus-id<"/"> .cat       ; optionally
                          opt<plus-id<":">> > >     ; qualified
schema-node-path = text .regexp schema-node-re

dependency-revision = {
  module-name: yang$yang-identifier
  module-revision: revision-identifier
}

assignment-range = {
  entry-point: sid
  size: sid
}

item = {
  ? status: "stable" / "unstable" / "obsolete"
  (
    namespace: "module" / "identity" / "feature"
    identifier: yang$yang-identifier
  //
    namespace: "data"
    identifier: schema-node-path
  )
  sid: sid
}
```

Figure 3: CDDL for RFC 9595

2.5. Your favorite RFC here...

3. CDDL definitions derived from IANA registries

Often, CDDL models need to use numbers that have been registered as values in IANA registries.

This section is intended to have one subsection for each CDDL data model presented that is derived from an existing IANA registry. As a start, it is fleshed out with two such data models.

The intention is that these reference modules are updated automatically (after each change of the registry or periodically, frequent enough.) Hence, this document can only present a snapshot for IANA-derived data models.

The model(s) presented here clearly are in proof-of-concept stage; suggestions for improvement are very welcome.

3.1. COSE Algorithms Registry

The IANA registry for COSE Algorithms is part of the IANA CBOR Object Signing and Encryption (COSE) registry group [IANA.cose].

The following automatically derived model defines some 80 CDDL rules that have the name for a COSE algorithm as its rule name and the actual algorithm number as its right hand side. The additional first rule is a type choice between all these constants; this could be used in places that just have to validate the presence of a COSE algorithm number that was registered at the time the model was derived.

This section does not explore potential filtering of the registry entries, e.g., by recommended status (such as leaving out deprecated entries) or by capabilities.

The names given in the COSE algorithms registry are somewhat irregular and do not consider their potential use in modeling or programming languages; the automatic derivation used here turns sequences of one or more spaces and other characters that cannot be in CDDL names ([/+] here) into underscores.

```
============= NOTE: '\' line wrapping per RFC 8792 ==============

algorithms = RS1 / A128CTR / A192CTR / A256CTR / A128CBC / \
  A192CBC / A256CBC / ESB512 / ESB384 / ESB320 / ESB256 / KT256 \
  / KT128 / TurboSHAKE256 / TurboSHAKE128 / WalnutDSA / RS512 / \
  RS384 / RS256 / Ed448 / ESP512 / ESP384 / ML-DSA-87 / ML-DSA-\
  65 / ML-DSA-44 / ES256K / HSS-LMS / SHAKE256 / SHA-512 / SHA-\
  384 / RSAES-OAEP_w_SHA-512 / RSAES-OAEP_w_SHA-256 / RSAES-\
  OAEP_w_RFC_8017_default_parameters / PS512 / PS384 / PS256 / \
  ES512 / ES384 / ECDH-SS_A256KW / ECDH-SS_A192KW / ECDH-\
  SS_A128KW / ECDH-ES_A256KW / ECDH-ES_A192KW / ECDH-ES_A128KW \
  / ECDH-SS_HKDF-512 / ECDH-SS_HKDF-256 / ECDH-ES_HKDF-512 / \
  ECDH-ES_HKDF-256 / Ed25519 / SHAKE128 / SHA-512_256 / SHA-256 \
  / SHA-256_64 / SHA-1 / direct_HKDF-AES-256 / direct_HKDF-AES-\
```

```
    128 / direct_HKDF-SHA-512 / direct_HKDF-SHA-256 / ESP256 / \
    EdDSA / ES256 / direct / A256KW / A192KW / A128KW / A128GCM / \
    A192GCM / A256GCM / HMAC_256_64 / HMAC_256_256 / HMAC_384_384 \
    / HMAC_512_512 / AES-CCM-16-64-128 / AES-CCM-16-64-256 / AES-\
    CCM-64-64-128 / AES-CCM-64-64-256 / AES-MAC_128_64 / AES-\
    MAC_256_64 / ChaCha20_Poly1305 / AES-MAC_128_128 / AES-\
    MAC_256_128 / AES-CCM-16-128-128 / AES-CCM-16-128-256 / AES-\
    CCM-64-128-128 / AES-CCM-64-128-256 / IV-GENERATION
  RS1 = -65535
  A128CTR = -65534
  A192CTR = -65533
  A256CTR = -65532
  A128CBC = -65531
  A192CBC = -65530
  A256CBC = -65529
  ESB512 = -268
  ESB384 = -267
  ESB320 = -266
  ESB256 = -265
  KT256 = -264
  KT128 = -263
  TurboSHAKE256 = -262
  TurboSHAKE128 = -261
  WalnutDSA = -260
  RS512 = -259
  RS384 = -258
  RS256 = -257
  Ed448 = -53
  ESP512 = -52
  ESP384 = -51
  ML-DSA-87 = -50
  ML-DSA-65 = -49
  ML-DSA-44 = -48
  ES256K = -47
  HSS-LMS = -46
  SHAKE256 = -45
  SHA-512 = -44
  SHA-384 = -43
  RSAES-OAEP_w_SHA-512 = -42
  RSAES-OAEP_w_SHA-256 = -41
  RSAES-OAEP_w_RFC_8017_default_parameters = -40
  PS512 = -39
  PS384 = -38
  PS256 = -37
  ES512 = -36
  ES384 = -35
  ECDH-SS_A256KW = -34
  ECDH-SS_A192KW = -33
```

```
ECDH-SS_A128KW = -32
ECDH-ES_A256KW = -31
ECDH-ES_A192KW = -30
ECDH-ES_A128KW = -29
ECDH-SS_HKDF-512 = -28
ECDH-SS_HKDF-256 = -27
ECDH-ES_HKDF-512 = -26
ECDH-ES_HKDF-256 = -25
Ed25519 = -19
SHAKE128 = -18
SHA-512_256 = -17
SHA-256 = -16
SHA-256_64 = -15
SHA-1 = -14
direct_HKDF-AES-256 = -13
direct_HKDF-AES-128 = -12
direct_HKDF-SHA-512 = -11
direct_HKDF-SHA-256 = -10
ESP256 = -9
EdDSA = -8
ES256 = -7
direct = -6
A256KW = -5
A192KW = -4
A128KW = -3
A128GCM = 1
A192GCM = 2
A256GCM = 3
HMAC_256_64 = 4
HMAC_256_256 = 5
HMAC_384_384 = 6
HMAC_512_512 = 7
AES-CCM-16-64-128 = 10
AES-CCM-16-64-256 = 11
AES-CCM-64-64-128 = 12
AES-CCM-64-64-256 = 13
AES-MAC_128_64 = 14
AES-MAC_256_64 = 15
ChaCha20_Poly1305 = 24
AES-MAC_128_128 = 25
AES-MAC_256_128 = 26
AES-CCM-16-128-128 = 30
AES-CCM-16-128-256 = 31
AES-CCM-64-128-128 = 32
AES-CCM-64-128-256 = 33
IV-GENERATION = 34
```

Figure 4: Derived CDDL for COSE Algorithms Registry

3.2. DNS Record Types

The IANA registry for DNS Record Types is part of the IANA Domain Name System (DNS) Parameters registry group [IANA.dns-parameters].

Using the library [IANA-REGISTRY] and a short script (Figure 5), a CDDL file for the Resource Record (RR) TYPEs registered in that registry group can be generated (Figure 6 in Appendix A):

```
require 'iana-registry'

DNS_RR = {}

REXML::XPath.each(IANA::Registry.load("dns-parameters").root,
    "//xmlns:registry[@id='dns-parameters-4']/xmlns:record",
    IANA::Registry::NS) do |x|
  typ = x.elements['type'].text
  value = x.elements['value'].text.to_i
  semantics = x.elements['description'].text
  if semantics && typ =~ /\A[-_A-Z0-9]+\z/
    DNS_RR[typ.gsub("-", "_")] = value
  end
end

puts DNS_RR.map { |t, v| "RR_#{t} = #{v}\n" }
```

Figure 5: Script for deriving CDDL for the Resource Record (RR) TYPEs Registry in DNS Parameters

4. IANA Considerations

This document makes no requests of IANA.

However, the use of IANA registries for deriving CDDL (e.g., as in Section 3) is an active subject of discussion.

5. Security considerations

The security considerations of [RFC8610], [RFC9682], [RFC9165], [RFC9741], [STD94] and [STD90] apply. This collection of CDDL models is not believed to create new security considerations. Errors in the models could -- if we knew of them, we'd fix those errors instead of explaining their security consequences in this section.

6. References

6.1. 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/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>.

[RFC9682] Bormann, C., "Updates to the Concise Data Definition Language (CDDL) Grammar", RFC 9682, DOI 10.17487/RFC9682, November 2024, <https://www.rfc-editor.org/rfc/rfc9682>.

[RFC9741] Bormann, C., "Concise Data Definition Language (CDDL): Additional Control Operators for the Conversion and Processing of Text", RFC 9741, DOI 10.17487/RFC9741, March 2025, <https://www.rfc-editor.org/rfc/rfc9741>.

[STD90] Internet Standard 90, <https://www.rfc-editor.org/info/std90>. At the time of writing, this STD comprises the following:

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>.

[STD94] Internet Standard 94, <https://www.rfc-editor.org/info/std94>. At the time of writing, this STD comprises the following:

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/info/rfc8949>.

6.2. Informative References

[IANA-REGISTRY] "iana-registry | Rubygems.org", <https://rubygems.org/gems/iana-registry>.

[IANA.cose] IANA, "CBOR Object Signing and Encryption (COSE)", <https://www.iana.org/assignments/cose>.

[IANA.dns-parameters]
IANA, "Domain Name System (DNS) Parameters",
<https://www.iana.org/assignments/dns-parameters>.

[RFC7071] Borenstein, N. and M. Kucherawy, "A Media Type for Reputation Interchange", RFC 7071, DOI 10.17487/RFC7071, November 2013, <https://www.rfc-editor.org/rfc/rfc7071>.

[RFC7807] Nottingham, M. and E. Wilde, "Problem Details for HTTP APIs", RFC 7807, DOI 10.17487/RFC7807, March 2016, <https://www.rfc-editor.org/rfc/rfc7807>.

[RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-editor.org/rfc/rfc8366>.

[RFC9290] Fossati, T. and C. Bormann, "Concise Problem Details for Constrained Application Protocol (CoAP) APIs", RFC 9290, DOI 10.17487/RFC9290, October 2022, <https://www.rfc-editor.org/rfc/rfc9290>.

[RFC9457] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457, DOI 10.17487/RFC9457, July 2023, <https://www.rfc-editor.org/rfc/rfc9457>.

[RFC9595] Veillette, M., Ed., Pelov, A., Ed., Petrov, I., Ed., Bormann, C., and M. Richardson, "YANG Schema Item iDentifier (YANG SID)", RFC 9595, DOI 10.17487/RFC9595, July 2024, <https://www.rfc-editor.org/rfc/rfc9595>.

Appendix A. Example CDDL generated from registries

This appendix collects examples that are too long for the main body of the text.

```
RR_A = 1
RR_NS = 2
RR_MD = 3
RR_MF = 4
RR_CNAME = 5
RR_SOA = 6
RR_MB = 7
RR_MG = 8
RR_MR = 9
RR_NULL = 10
RR_WKS = 11
RR_PTR = 12
```

```
   RR_HINFO = 13
   RR_MINFO = 14
   RR_MX = 15
   RR_TXT = 16
   RR_RP = 17
   RR_AFSDB = 18
   RR_X25 = 19
   RR_ISDN = 20
   RR_RT = 21
   RR_NSAP = 22
   RR_NSAP_PTR = 23
   RR_SIG = 24
   RR_KEY = 25
   RR_PX = 26
   RR_GPOS = 27
   RR_AAAA = 28
   RR_LOC = 29
   RR_NXT = 30
   RR_EID = 31
   RR_NIMLOC = 32
   RR_SRV = 33
   RR_ATMA = 34
   RR_NAPTR = 35
   RR_KX = 36
   RR_CERT = 37
   RR_A6 = 38
   RR_DNAME = 39
   RR_SINK = 40
   RR_OPT = 41
   RR_APL = 42
   RR_DS = 43
   RR_SSHFP = 44
   RR_IPSECKEY = 45
   RR_RRSIG = 46
   RR_NSEC = 47
   RR_DNSKEY = 48
   RR_DHCID = 49
   RR_NSEC3 = 50
   RR_NSEC3PARAM = 51
   RR_TLSA = 52
   RR_SMIMEA = 53
   RR_HIP = 55
   RR_NINFO = 56
   RR_RKEY = 57
   RR_TALINK = 58
   RR_CDS = 59
   RR_CDNSKEY = 60
   RR_OPENPGPKEY = 61
```

```
   RR_CSYNC = 62
   RR_ZONEMD = 63
   RR_SVCB = 64
   RR_HTTPS = 65
   RR_DSYNC = 66
   RR_HHIT = 67
   RR_BRID = 68
   RR_EUI48 = 108
   RR_EUI64 = 109
   RR_NXNAME = 128
   RR_TKEY = 249
   RR_TSIG = 250
   RR_IXFR = 251
   RR_AXFR = 252
   RR_MAILB = 253
   RR_MAILA = 254
   RR_URI = 256
   RR_CAA = 257
   RR_AVC = 258
   RR_DOA = 259
   RR_AMTRELAY = 260
   RR_RESINFO = 261
   RR_WALLET = 262
   RR_CLA = 263
   RR_IPN = 264
   RR_TA = 32768
   RR_DLV = 32769
```

Figure 6: Derived CDDL for the Resource Record (RR) TYPEs Registry in DNS Parameters

List of Figures

Acknowledgements

TBD

Author’s Address

Carsten Bormann
Universit辰t Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org