Concise Binary Object Representation (CBOR) Encoding Rules for ASN.1

Objective Systems, Inc.

May 21, 2026

Copyright Notice
Copyright ©2026 Objective Systems, Inc. All rights reserved.
This document may be distributed in any form, electronic or otherwise, provided that it is distributed in its entirety and that the copyright and this notice are included.

Author's Contact Information
Comments, suggestions, and inquiries regarding this document may be submitted via electronic mail to info@obj-sys.com.

Table of Contents

©2026 Objective Systems, Inc.

Summary

This document specifies rules for encoding values of ASN.1 types using the Concise Binary Object Representation (CBOR). These encoding rules have been specified by Objective Systems, Inc.

CBOR is perhaps the most widely used JSON binary serialization format at this time. Unlike others, it has been standardized by a leading Internet standards body (the IETF). This paper provides a mapping of ASN.1 to CBOR to provide a way for users of ASN.1 standards to work with data in this format.

©2026 Objective Systems, Inc.

Normative References

The following references are considered normative for this specification.

- ITU-T Recommendation X.680 (2008) | ISO/IEC 8824-1:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.
- ITU-T Recommendation X.681 (2008) | ISO/IEC 8824-2:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.
- ITU-T Recommendation X.682 (2008) | ISO/IEC 8824-3:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (2008) | ISO/IEC 8824-4:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.
- ECMA-262 (2011), *ECMAScript Language Specification*
- IETF RFC 8949, *Concise Binary Object Representation (CBOR)* (https://www.rfc-editor.org/rfc/rfc8949.html)
- IETF RFC 9090, *Concise Binary Object Representation (CBOR) Tags for Object Identifiers* (https://www.rfc-editor.org/rfc/rfc9090)
- IEEE 754, *IEEE Standard for Floating-Point Arithmetic* (https://ieeexplore.ieee.org/document/8766229)
- IANA *Concise Binary Object Representation (CBOR) Tags* (https://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml)

©2026 Objective Systems, Inc.

Encoding by ASN.1 Type

All CBOR encoding examples below are from the cbor.me site.

BOOLEAN

A value of `true` is encoded as the CBOR `true` simple value, and a value of `false` is encoded as the CBOR `false` simple value (as described in RFC 8949).

So given this ASN.1:

```
bElement BOOLEAN
```

a value of FALSE would be encoded with this CBOR:

```
F4                    # primitive(20)
```

INTEGER

Most INTEGER values are encoded as described in RFC 8949; i.e., an INTEGER having the value zero or higher is encoded as a CBOR unsigned integer, and an INTEGER having a negative value is encoded as a CBOR negative integer.

An INTEGER having a positive value higher than 18,446,744,073,709,551,615 is encoded as a CBOR positive bignum. An INTEGER having a negative value smaller than -18,446,744,073,709,551,616 is encoded as a CBOR negative bignum.

So given this ASN.1:

```
iElement INTEGER
```

the following values would be encoded with the indicated CBOR:

23:

```
17                    # unsigned(23)
```

25:

```
18 19                 # unsigned(25)
```

281,474,976,710,656:

```
1B 0001000000000000 # unsigned(281474976710656)
```

18,446,744,073,709,551,616:

```
C2                    # tag(2)
   49                 # bytes(9)
      01000000000000000 # "\u0001\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000"
```

-1:

```
20                    # negative(0)
```

©2026 Objective Systems, Inc.

-500:

```
   39 01F3              # negative(499)
```

-18,446,744,073,709,551,616:

```
   3B FFFFFFFFFFFFFFFF # negative(18446744073709551615)
```

-18,446,744,073,709,551,617:

```
   C3                    # tag(3)
      49                 # bytes(9)
         01000000000000000 # "\u0001\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000"
```

ENUMERATED

Enumerated values are encoded as CBOR text strings[1].

So given this ASN.1:

```
   eElement ENUMERATED {zero, one, two}
```

a value of one would be encoded with this CBOR:

```
   63                   # text(3)
      6F6E65            # "one"
```

REAL

Base 2 floating-point numbers are encoded as half-precision (16 bits), single-precision (32 bits), or double-precision (64 bits) as specified in IEEE 754. The smallest format that preserves the value is used.

So given this ASN.1:

```
   rElement REAL (WITH COMPONENTS {..., base (2)})
```

the following values would be encoded with the indicated CBOR:

1.5:

```
   F9 3E00              # primitive(15872)
```

i.e., half-precision in 16 bits.

3.0 E 8 (the speed of light in meters per second):

```
   FA 4D8F0D18          # primitive(1301220632)
```

i.e., single-precision in 32 bits.

3.14 (pi):

```
   FB 40091EB851EB851F # primitive(4614253070214989087)
```

i.e.; double-precision in 64 bits.

1 Per RFC 8949, CBOR text strings are encoded using UTF-8.

©2026 Objective Systems, Inc.

Base 10 floating-point numbers are encoded as CBOR text strings.

So given this ASN.1:

```
rElement REAL (WITH COMPONENTS {..., base (10)})
```

the following values would be encoded with the indicated CBOR:

1.5:

```
63                  # text(3)
   312E35           # "1.5"
```

3.0 E 8:

```
65                  # text(5)
   332E304538       # "3.0E8"
```

3.14:

```
64                  # text(4)
   332E3134         # "3.14"
```

BIT STRING

BIT STRINGs of fixed size are encoded as CBOR byte strings. The number of bytes encoded is the minimum necessary to hold the indicated number of bits. Neither the number of used bits nor the number of unused bits is part of the encoding; it's expected that the consumer of the BIT STRING will know its size.

So given this ASN.1:

```
bsElement BIT STRING (SIZE(10))
```

the value 01010101 01 would be encoded with this CBOR:

```
42                  # bytes(2)
   5540             # "U@"
```

0x5540 is 0b01010101 01000000.

BIT STRINGs of variable size are encoded as an indefinite length CBOR map with two members. The key for the first member is the CBOR text string "length". The value of the first member is the number of bits encoded as a CBOR integer. The key for the second member is the CBOR text string "value". The value of the second member is a CBOR byte string. Thus the encoding makes use of the name/value pair (NVP) idea used by technologies like JSON and XML. The map is terminated by the CBOR break marker byte (0xFF).

So given this ASN.1:

```
bsElement BIT STRING
```

the value 01010101 01 would be encoded with this CBOR:

```
BF                  # map(*)
```

©2026 Objective Systems, Inc.

```
66                  # text(6)
   6C656E677468     # "length"
0A                  # unsigned(10)
65                  # text(5)
   76616C7565       # "value"
42                  # bytes(2)
   5540             # "U@"
FF                  # primitive(*)
```

As of this writing the CONTAINING clause is not supported for CBOR encoding of a BIT STRING and will result in an empty CBOR byte string being encoded.

OCTET STRING

Values of type OCTET string are encoded as CBOR byte strings.

So given this ASN.1:

```
osElement OCTET STRING
```

a value of 0xAC 0xDC would be encoded with this CBOR:

```
42                   # bytes(2)
  ACDC               # "\xAC\xDC"
```

As of this writing the CONTAINING clause is not supported for CBOR encoding of an OCTET STRING and will result in an empty CBOR byte string being encoded.

NULL

Values of type NULL are encoded as the CBOR `null` simple value (as described in RFC 8949).

So given this ASN.1:

```
nElement NULL
```

the value would be encoded with this CBOR:

```
F6                  # primitive(22)
```

SEQUENCE and SET

SEQUENCEs and SETs are encoded as indefinite length CBOR maps. Each element has an identifier (the element name) encoded as a CBOR text string. The identifier is followed by the encoding of the value per the rules expressed in this document. The map is terminated by a CBOR break marker byte (0xFF).

So given this ASN.1:

©2026 Objective Systems, Inc.

```
OSysTestRecord ::= SEQUENCE {
   iElement INTEGER,
   sElement UTF8String
}
```

a sequence with the value 42 for iElement and the value “abc” for sElement would be encoded with this CBOR:

```
BF                    # map(*)
   68                 # text(8)
      69456C656D656E74 # "iElement"
   18 2A              # unsigned(42)
   68                 # text(8)
      73456C656D656E74 # "sElement"
   63                 # text(3)
      616263          # "abc"
   FF                 # primitive(*)
```

SEQUENCE OF and SET OF

SEQUENCE OF and SET OF are encoded as indefinite length CBOR arrays. The array is terminated by a CBOR break marker byte (0xFF).

So given this ASN.1:

```
   soElement SEQUENCE OF INTEGER
```

the sequence 42 (the answer to life, the universe, and everything[2]), 339, and 248,655 (the farthest distance from Earth, in miles, realized by the Artemis II astronauts) would be encoded with this CBOR:

```
   9F                   # array(*)
      18 2A             # unsigned(42)
      19 0153           # unsigned(339)
      1A 0003CB4F       # unsigned(248655)
      FF                # primitive(*)
```

CHOICE

A CHOICE is encoded as an indefinite length CBOR map with one item representing the chosen alternative. The identifier is the name of the alternative encoded as a CBOR text string. The value is encoded per the rules expressed in this document.

2 The Hitchhiker’s Guide to the Galaxy, Douglas Adams

©2026 Objective Systems, Inc.

So given this ASN.1:

```
OsysTestRecord ::= CHOICE {
   bAlternative BOOLEAN,
   sAlternative UTF8String
}
```

if the boolean alternative is chosen with a value of true, the CBOR encoding would be as follows:

```
BF                              # map(*)
   6C                           # text(12)
      62416C7465726E6174697665  # "bAlternative"
   F5                           # primitive(21)
   FF                           # primitive(*)
```

Object Identifier

The encoding of an OBJECT IDENTIFIER specifies a tag number of 111 in the CBOR header bytes as indicated by the CBOR Tags Registry.

The OBJECT IDENTIFIER itself is encoded as a CBOR byte string using the mechanism documented in RFC 9090; i.e., the first two numbers of the value are collapsed into one octet, and the remaining numbers are encoded in groups of 7 bits, with the high-order bit set to 1 to indicate the encoding continues into the next octet or to 0 to indicate the encoding is finished.

So given this ASN.1:

```
   oiElement OBJECT IDENTIFIER
```

the value 1 2 100000 4 5 6 would be encoded with this CBOR:

```
   D8 6F                # tag(111)
      47                # bytes(7)
         2A868D20040506  # "*\x86\x8D \u0004\u0005\u0006"
```

Relative Object Identifier

The encoding of a RELATIVE-OID is the same as that for an OBJECT IDENTIFIER with the following exceptions:

- The tag number is 110 instead of 111, as indicated in the CBOR Tags Registry.
- The encoding does not attempt to collapse the first two values into one octet (RFC 9090).

So given this ASN.1:

```
   roElement RELATIVE-OID
```

the value 1 2 100000 4 5 6 would be encoded with this CBOR:

©2026 Objective Systems, Inc.

```
   D8 6E                  # tag(110)
      48                  # bytes(8)
         0102868D20040506 # "\u0001\u0002\x86\x8D \u0004\u0005\u0006"
```

Embedded-pdv

As of this writing the encoding of values of type EMBEDDED PDV is not supported for CBOR.

External

The encoding of values of type EXTERNAL is an encoding of this SEQUENCE[3]:

```
EXTERNAL ::= [UNIVERSAL 8] IMPLICIT SEQUENCE {
   direct-reference OBJECT IDENTIFIER OPTIONAL,
   indirect-reference INTEGER OPTIONAL,
   data-value-descriptor ObjectDescriptor OPTIONAL,
   encoding CHOICE {
      single-ASN1-type [0] ANY,
      octet-aligned [1] IMPLICIT OCTET STRING,
      arbitrary [2] IMPLICIT BIT STRING
   }
}
```

Time

Values of type TIME are encoded as CBOR text strings.

So given this ASN.1:

```
   tElement TIME((SETTINGS "Basic=Date Date=Y Year=Basic")|
               (SETTINGS "Basic=Date Date=Y Year=Proleptic"))
```

the value 1582 would be encoded with this CBOR:

```
   64                  # text(4)
      31353832         # "1582"
```

Restricted Character String

Restricted character strings are encoded as CBOR text strings.

So given this ASN.1:

3 ASN.1 Communication between Heterogeneous Systems, Olivier Dubuisson, page 299

©2026 Objective Systems, Inc.

```
RestrictedString ::= UTF8String ("xyz")

OsysTestRecord ::= SEQUENCE {
   sElement RestrictedString
}
```

the value “xyz” would be encoded with this CBOR:

```
   63                  # text(3)
      78797A           # "xyz"
```

Unrestricted Character String

Unrestricted character strings are encoded as CBOR text strings.

So given this ASN.1:

```
   sElement UTF8String
```

the value “abc” would be encoded with this CBOR:

```
   63                  # text(3)
      616263           # "abc"
```

Generalized time, Universal time, Object Descriptor

Values of type GeneralizedTime, UTCTime, and ObjectDescriptor are encoded as CBOR text strings.

So given this ASN.1:

```
   gtElement GeneralizedTime
```

the value for 5/5/2026 2:02:07.896 p.m. would be encoded with this CBOR:

```
   73                                     # text(19)
      3230323630353035313430323037 2E3839365A # "20260505140207.896Z"
```

And given this ASN.1:

```
   utElement UTCTime
```

the value for 5/5/2026 2:29:50 p.m. would be encoded with this CBOR:

```
   6D                             # text(13)
      32363035303531343239355 05A # "260505142950Z"
```

And given this ASN.1:

```
   odElement ObjectDescriptor
```

the value “awesome object” would be encoded with this CBOR:

```
   6E                                  # text(14)
      617765736F6D65206F626A656374 # "awesome object"
```

Open Type

If the type of an open type can be determined, the encoding will be of that type. If the type of an open type can't be determined, the encoding will be a CBOR byte string.

So given this ASN.1:

```
   otElement TYPE-IDENTIFIER.&Type
```

if the integer value 256 would be recognized as an INTEGER, it would be encoded as a CBOR integer:

```
   19 0100             # unsigned(256)
```

But with just the above ASN.1 and nothing else, the value would not be recognized as an INTEGER, so it's encoded as a CBOR byte string:

```
   42                   # bytes(2)
     0100               # "\u0001\u0000"
```

So if a value to be encoded in CBOR is a raw encoding in some other format, that raw encoding will be encoded to CBOR as a CBOR byte string.

Example

Pulling together the concepts discussed above, consider this ASN.1 specification:

```
OsysTest DEFINITIONS AUTOMATIC TAGS ::= BEGIN
EXPORTS;

ThisOrThat ::= CHOICE {
   this INTEGER,
   that UTF8String
}

Record ::= SEQUENCE {
   mainInfo SEQUENCE {
      bElement BOOLEAN,
      iElement INTEGER,
      eElement ENUMERATED {new, active, closed},
      r2Element REAL (WITH COMPONENTS {..., base (2)}),
      bsElement BIT STRING,
      osElement OCTET STRING,
      nElement NULL,
      cElement ThisOrThat,
      sElement UTF8String
   },
   r10Element REAL (WITH COMPONENTS {..., base (10)}),
```

©2026 Objective Systems, Inc.

```
   otElement TYPE-IDENTIFIER.&Type
}

Records ::= SEQUENCE OF record Record

END
```

Now consider this set of values for one instance of type Record to be encoded for the PDU type Records:

```
Records {
   {
      mainInfo {
         bElement = false
         iElement = 42
         eElement = new
         r2Element = 3.14
         bsElement = 01010101
         osElement = ACDC
         nElement =
         cElement {
            that = abc
         }
         sElement = xyz
      }
      r10Element = 123.45
      otElement = 0100
   }
}
```

This set of values would be encoded with the following CBOR:

```
9F                          # array(*)
   BF                       # map(*)
      68                    # text(8)
         6D61696E496E666F    # "mainInfo"
      BF                    # map(*)
```

©2026 Objective Systems, Inc.

```
68                  # text(8)
   62456C656D656E74  # "bElement"
F4                  # primitive(20)
68                  # text(8)
   69456C656D656E74  # "iElement"
18 2A               # unsigned(42)
68                  # text(8)
   65456C656D656E74  # "eElement"
63                  # text(3)
   6E6577            # "new"
69                  # text(9)
   7232456C656D656E74 # "r2Element"
FB 40091EB851EB851F  # primitive(4614253070214989087)
69                  # text(9)
   6273456C656D656E74 # "bsElement"
BF                  # map(*)
   66               # text(6)
      6C656E677468   # "length"
   08               # unsigned(8)
   65               # text(5)
      76616C7565    # "value"
   41               # bytes(1)
      55            # "U"
   FF               # primitive(*)
69                  # text(9)
   6F73456C656D656E74 # "osElement"
42                  # bytes(2)
   ACDC             # "\xAC\xDC"
68                  # text(8)
   6E456C656D656E74  # "nElement"
F6                  # primitive(22)
68                  # text(8)
```

©2026 Objective Systems, Inc.

```
      63456C656D656E74   # "cElement"
   BF                    # map(*)
      64                 # text(4)
         74686174        # "that"
      63                 # text(3)
         616263          # "abc"
      FF                 # primitive(*)
   68                    # text(8)
      73456C656D656E74   # "sElement"
   63                    # text(3)
      78797A             # "xyz"
   FF                    # primitive(*)
6A                       # text(10)
   723130456C656D656E74  # "r10Element"
66                       # text(6)
   3132332E3435          # "123.45"
69                       # text(9)
   6F74456C656D656E74    # "otElement"
42                       # bytes(2)
   0100                  # "\u0001\u0000"
FF                       # primitive(*)
FF                       # primitive(*)
```

©2026 Objective Systems, Inc.