CBOR (Concise Binary Object Representation) Maint. and Ext.
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

C. Bormann
Universit辰t Bremen TZI
M. Matejka
CZ.NIC

2 March 2026

Stand-in Tags for YANG-CBOR
draft-bormann-cbor-yang-standin-03

Abstract

YANG (RFC 7950) is a data modeling language used to model configuration data, state data, parameters and results of Remote Procedure Call (RPC) operations or actions, and notifications.

YANG-CBOR (RFC 9254) defines encoding rules for YANG in the Concise Binary Object Representation (CBOR) (RFC 8949). While the overall structure of YANG-CBOR is encoded in an efficient, binary format, YANG itself has its roots in XML and therefore traditionally encodes some information such as date/times and IP addresses/prefixes in a verbose text form.

This document defines how to use existing CBOR tags for this kind of information in YANG-CBOR as a "stand-in" for the text-based information that would be found in the original form of YANG-CBOR.

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-yang-standin/.

Discussion of this document takes place on the CBOR (Concise Binary Object Representation) 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/cabo/yang-standin.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 3 September 2026.

Copyright Notice

Table of Contents

1. Introduction

(see abstract)

2. Conventions and Definitions

The terminology of [RFC9254] applies.

The present document focuses on a "basic" processing model where the original source of YANG-modeled data serves as the encoder, and where the decoder is part of the ultimate destination of those data.
// TODO: More elaborate processing models are possible and desirable; several of the terms defined below are defined here only so they can be used to discuss processing models of this kind.

Legacy representation: The (often text-based) representation for a YANG data item as used in YANG-XML, YANG-JSON, and (unchanged) YANG-CBOR.

Stand-in tag: A CBOR tag that can supply the information that is equivalent to a legacy representation in a more efficient format (e.g., using binary data).

Encoder: The party which generates (sends) CBOR data described by YANG.

Intermediate Encoder: An encoder which isn't the original author of the data, converting it from legacy representation.

Aggressive Intermediate Encoder: An intermediate encoder that might choose to discard some information of a legacy representation in order to be able to use a stand-in tag. Such a choice may be based on knowledge of the Decoder's handling of such information (e.g, to accommodate intolerant decoders), or it may be a general characteristic of the service provided by the intermediate encoder (e.g., in order to serve as a legacy-eschewing encoder).

Legacy-Eschewing Encoder: An encoder that does not generate legacy representations in places where a stand-in tag might instead be used. An intermediate encoder may need to be aggressive to achieve this.

Decoder: The party which receives and parses CBOR data described by YANG.

Intolerant Decoder: A decoder that does not accept legacy representations in places where a stand-in tag might instead be used. Such a decoder is designed to interoperate only with a legacy-eschewing encoder.

Intermediate Decoder: A decoder which isn't the final recipient of the data, converting it to legacy representation.

Data Transfer: A series of actions, generally beginning by data origination, encoding, continuing by optional intermediate transcoding, sending and receiving, and finally decoding and consuming.

Round Trip: Part of a data transfer between an encoder generating CBOR data with stand-in tags and a decoder parsing the data.

Legacy Round Trip: A Round Trip where the encoder is an intermediate encoder or the decoder is an intermediate decoder and any of these converts from or to the legacy representation.

Unambiguous Round Trip: A Legacy Round Trip that provides exactly the same legacy representation (not just semantically equivalent). The stand-in tag is also said to "unambiguously stand in" for the legacy representation.

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 [BCP14] (RFC2119) (RFC8174) when, and only when, they appear in all capitals, as shown here.

3. Stand-In Tags

As a complement to the representations defined in [RFC9254], the present document defines efficient representations of certain types of YANG-modeled data, based on both existing and newly registered "stand-in" tags that are designed to represent the underlying data. Whenever a stand-in tag is defined for a specific type, the representation as specified in this document may be used. Otherwise, the encoding is as specified in [RFC9254].

This section defines several standard stand-in tags for general use, in many cases based on the type definitions of [RFC9911].

(
// TODO: The detailed conditions under which certain transformations can be applied to YANG-CBOR data representations after these have been generated are TBD; several processing models are conceivable for this. These processing models are out of scope of the present document; work has started in [I-D.marenamat-netmod-core-yang-transcoding]. One such condition might be: Where information starts out in a legacy representation, these tags are only used when an Unambiguous Round Trip can be achieved.)

3.1. ietf-yang-types: Tag 1 (Date/Time) and Tag 100 (Date)

Section 3 of [RFC9911] defines the following types in ietf-yang-types:

YANG type	base type	specification	stand-in
date-and-time	string	[RFC6021]	tag 1
date	string	[RFC9911]	(none)
date-no-zone	string	[RFC9911]	tag 100

Table 1: Legacy date and date/time representation in ietf-yang-types

Tag 1 (Section 3.4.2 of RFC 8949 [STD94]) can unambiguously stand in for all date-and-time values that:

* do not specify a time zone (note that [RFC9911] recommends using "Z", but allows using the legacy "-00:00" format, for time-zone-free date-times)
* are not an inserted leap second (23:59:60 or 23:59:61)
* do not have trailing zeroes in the fractional part of the seconds.
* do not have fractional parts of the seconds with a precision that cannot be represented in floating-point tag content in a tag 1.

All other date-and-time values stay in legacy representation. (// TODO: explore the use of tag 1001 for some of these cases.)

Tag 1 uses an integer tag content for all date-and-time values without fractional seconds and a floating-point tag content for values that have fractional seconds given.

Tag 100 [RFC8943] can unambiguously stand in for all date-no-zone values.

3.2. ietf-yang-types: Tags 37 (UUID), CPA113 (hex-string), and CPA114 (dotted-quad)

Section 3 of [RFC9911] defines the following types in ietf-yang-types:

YANG type	base type	specification	stand-in
uuid	string	[RFC9911]	tag 37
hex-string	string	[RFC9911]	tag CPA113
mac-address	string	[RFC6021]	tag CPA113
phys-address	string	[RFC6021]	tag CPA113

Table 2: Legacy UUID and colon-separated hexadecimal representations in ietf- yang-types

These types are hexadecimal representations of byte strings, adorned in various ways.

uuid stands for a 16-byte byte string (Section 4 of [RFC9562]), represented in hexadecimal with ASCII minus/hyphen characters added in specific positions. Tag 37 (see also Section 7 of [I-D.bormann-cbor-notable-tags]) can be used as a binary stand-in for this adorned hexadecimal representation. According to the description of uuid in Section 3 of [RFC9911], "the canonical representation uses lowercase characters". For consistency with this specification, an intermediate decoder of a tag 37 stand-in MUST use lowercase characters in the uuid hex string generated (// TODO: align with processing model).

hex-string, and the similar, but more specific types mac-address and phys-address, stand for byte strings in various lengths (exactly 6 bytes for mac-address, variable-length for the others), represented in hexadecimal with ASCII colon characters added between the representations of each of the bytes. This specification defines tag number CPA113 Section 7.1 to be an additional "Expected Later

Encoding" tag (similar to tag 23, see Section 3.4.5.2 of RFC 8949 [STD94]), except that the expected encoding of CPA113 includes colons and uses lowercase hex digits.

The following example implementation of the transformation in a decoder shows the use of lowercase hex characters (%02x as opposed to %02X) and the insertion of colon characters between the hex-represented bytes:

```
def tag_cpa113_to_legacy(s)
  s.bytes.map{|x| "%02x" % x}.join(":")
end
```

Note: Section 2.4 of [RFC9542] defines tag number 48 for MAC addresses. This could be used in place of tag CPA113, but only for MAC addresses, not for other byte strings of a similar form. This specification therefore requests IANA to assign a new CBOR tag that can be used as a stand-in for all instances of colon-separated text strings of hexadecimally represented bytes, as shown in Table 2.

Note: Related stand-in semantics have not been defined so far for tag 21 or 22 that were defined alongside tag 23: YANG has a base type "binary" that is encoded in base64 classic in YANG-XML and YANG-JSON, but already encoded in a binary byte string in YANG-CBOR. Use cases that might actually use base type "string" for base64-encoded data in YANG have not been considered. However, tag 21 or 22 could be used as stand-in tags if that is useful for some specific YANG model not considered here.

```
// RFC-Editor: This document uses the CPA (code point allocation)
// convention described in [I-D.bormann-cbor-draft-numbers]. For
// each usage of the term "CPA", please remove the prefix "CPA" from
// the indicated value and replace the residue with the value
// assigned by IANA; perform an analogous substitution for all other
// occurrences of the prefix "CPA" in the document. Finally, please
// remove this note.
```

YANG type	base type	specification	stand-in
dotted-quad	string	[RFC6991]	tag CPA114

Table 3: Legacy dotted-quad representation in ietf-yang-types

dotted-quad stands for an unsigned 32-bit integer (Section 3 of [RFC9911]), represented as a text string in decimal values for 4 byte values in network byte order, separated by ASCII dot characters. Tag CPA114 can be used as a binary stand-in for this adorned bytewise decimal representation.

IANA is requested to assign CPA114 as the CBOR tag to indicate a value that is represented by a dotted-quad in YANG. The tag content is an unsigned integer of a size not greater than 4 bytes (0..0xFFFFFFFF, uint32). It is expected that when the data is being displayed e.g. to human operator, the data will be shown as a string of 4 decimal numbers giving the number as four bytes in network byte order, separated by ASCII dot characters.

// RFC-Editor: This document uses the CPA (code point allocation) convention described in [I-D.bormann-cbor-draft-numbers]. For each usage of the term "CPA", please remove the prefix "CPA" from the indicated value and replace the residue with the value assigned by IANA; perform an analogous substitution for all other occurrences of the prefix "CPA" in the document. Finally, please remove this note.

// TODO: Better formulation of the dotted-quad expected later encoding.
PROPOSED: RFC 9911 is clear that dotted-quad stands for an unsigned integer that fits into 32 bits; we should not replace that with a byte string.
RESOLVED: Should we fix the 4-byte length or make it more generic with arbitrary length byte strings? (For usage as IPv4 similar identifier -- see ietf-ospf.yang -- the 4-byte length is sufficient and brings less complexity)

3.3. ietf-inet-types: Tags 54 and 52 (IP addresses and prefixes)

Section 4 of [RFC9911] defines in ietf-inet-types:

YANG type	base type	specification	stand-in
ip-address	union	[RFC6021]	(see union)
ipv6-address	string	[RFC6021]	tag 54
ipv4-address	string	[RFC6021]	tag 52
ip-address-no-zone	union	RFC 6991	(see union)
ipv6-address-no-zone	ipv6-address	RFC 6991	tag 54
ipv4-address-no-zone	ipv4-address	RFC 6991	tag 52
ip-address-link-local	union	[RFC9911]	(see union)
ipv6-address-link-local	ipv6-address	[RFC9911]	tag 54
ipv4-address-link-local	ipv4-address	[RFC9911]	tag 52
ip-prefix	union	[RFC6021]	(see union)
ipv6-prefix	string	[RFC6021]	tag 54
ipv4-prefix	string	[RFC6021]	tag 52
ip-address-and-prefix	union	[RFC9911]	(see union)
ipv6-address-and-prefix	string	[RFC9911]	tag 54
ipv4-address-and-prefix	string	[RFC9911]	tag 52

Table 4: Legacy representations in ietf-yang-types

(
// TODO: align this and next paragraph with processing model.) An intermediate encoder MAY normalize IPv6 addresses and prefixes that do not comply with [RFC5952] but can be converted into the stand-in representation. For example, IPv6 address written as 2001:db8:: is the same as 2001:0db8::0:0 and both would be converted to 54(h'20010db80000000000000000000000'); only the first one complies with [RFC5952]. The encoder MAY refuse to convert the latter one.

If the schema specifies ip-prefix, an intermediate encoder MAY normalize prefixes with non-zero bits after the prefix end. For example, if the legacy representation of ipv6-prefix is 2001:db8:1::/40, the encoder may either refuse it as malformed or convert it to 2001:db8::/40 and represent as 54([40, h'20010db8']).

The encoder implementation should be clear about which normalizations are employed and how.

Adapted examples from [RFC9164]:

Stand-in representation of IPv6 address 2001:db8:1234:deed:beef:cafe:face:feed is 54(h'20010db81234deedbeefcafefacefeed').

CBOR encoding of stand-in (19 bytes):

```
D8 36                                   # tag(54)
   50                                   # bytes(16)
      20010DB81234DEEDBEEFCAFEFACEFEED
```

CBOR encoding of legacy representation (40 bytes):

```
78 26                                    # text(38)
   323030313A6462383A313233343A646565643A
   626565663A636166653A666163653A66656564
   # "2001:db8:1234:deed:beef:cafe:face:feed"
```

Stand-in representation of IPv6 prefix 2001:db8:1234::/48 is 54([48, h'20010db81234']).

CBOR encoding of stand-in (12 bytes):

```
D8 36                # tag(54)
   82                # array(2)
      18 30          # unsigned(48)
      46             # bytes(6)
         20010DB81234 # " \u0001\r\xB8\u00124"
```

CBOR encoding of legacy representation (19 bytes):

```
72                                      # text(18)
   323030313A6462383A313233343A3A2F3438 # "2001:db8:1234::/48"
```

Stand-in representation of IPv6 link-local address fe80::0202:02ff:ffff:fe03:0303/64%eth0 is 54([h'fe8000000000020202ffffffe030303', 64, 'eth0']).

CBOR encoding of stand-in (27 bytes):

```
D8 36                                   # tag(54)
   83                                   # array(3)
      50                                # bytes(16)
         FE8000000000020202FFFFFFE030303
      18 40                             # unsigned(64)
      44                                # bytes(4)
         65746830                       # "eth0"
```

CBOR encoding of legacy representation (40 bytes):

```
78 26                                   # text(38)
   666538303A3A303230323A303266663A666666
   663A666530333A303330332F36342565746830
   # "fe80::0202:02ff:ffff:fe03:0303/64%eth0"
```

Stand-in representation of IPv4 address 192.0.2.1 is 52(h'c0000201').

CBOR encoding of stand-in (7 bytes):

```
D8 34          # tag(52)
   68          # bytes(4)
      C0000201
```

CBOR encoding of legacy representation (10 bytes):

```
69                    # text(9)
   3139322e302e322e31 # "192.0.2.1"
```

Stand-in representation of IPv4 prefix 192.0.2.0/24 is 52([24, h'c0000200']).

CBOR encoding of stand-in (10 bytes):

```
D8 34             # tag(52)
   82             # array(2)
      18 18       # unsigned(24)
      48          # bytes(4)
         C0000200
```

CBOR encoding of legacy representation (13 bytes):

```
6C                          # text(12)
   3139322e302e322e302f3234 # "192.0.2.0/24"
```

Stand-in representation of IPv4 address combined with prefix 192.0.2.1/24 is 52([h'c0000201', 24]).

CBOR encoding of stand-in (10 bytes):

```
D8 34             # tag(52)
   82             # array(2)
      48          # bytes(4)
         C0000201
      18 18       # unsigned(24)
```

CBOR encoding of legacy representation (13 bytes):

```
6C                          # text(12)
   3139322e302e322e312f3234 # "192.0.2.1/24"
```

TO DO: Check how the unions in [RFC9911] interact with this. E.g., the union ip-address needs to be parsed to decide between tag 54 and tag 52.

3.4. Union handling

When the schema specifies a union data type for a node, there are additional requirements on the encoder and decoder. The encoder MUST be fully aware of data semantics and use the appropriate data type and encoding. The decoder MUST use the data type information for further processing, in a similar way as specified in Section 6.10 of [RFC7951].

(
// TODO: Align the rest of the section with processing models.) An encoder which is fully aware of data semantics MUST use the appropriate data type, even though it isn't formally specified by the schema.

If an intermediate encoder doesn't fully understand the data semantics, it needs to find out which type the data actually is to choose the right stand-in. If more types are possible, it MAY choose any of these which allow for an Unambiguous Round Trip, otherwise it SHOULD keep the legacy representation.

If a decoder receives data for a union-typed node, it MUST accept any data type of the union, even though it may violate additional constraints outside the schema.

4. Using Stand-In Tags

4.1. The Standin File

Encoder and decoder need some agreement over whether, where and how stand-in tags are used. This agreement is embodied in a "standin file", which has approximately the data content of Table 5 (representation TBD):

YANG typename	Tag number(s)	Description (*)
ietf-yang-types:date-and-time	1	Section 3.1 of RFCthis
ietf-yang-types:date-no-zone	100	Section 3.1 of RFCthis
ietf-yang-types:uuid	37	Section 3.2 of RFCthis
ietf-yang-types:hex-string	CPA113	Section 3.2 of RFCthis
ietf-yang-types:mac-address	CPA113	Section 3.2 of RFCthis
ietf-yang-types:phys-address	CPA113	Section 3.2 of RFCthis
ietf-yang-types:dotted-quad	CPA114	Section 3.2 of RFCthis
ietf-inet-types:ip-address	54, 52 (see union)	Section 3.3 of RFCthis

ietf-inet-types:ipv6-address	54	Section 3.3 of RFCthis
ietf-inet-types:ipv4-address	52	Section 3.3 of RFCthis
ietf-inet-types:ip-address-no-zone	54, 52 (see union)	Section 3.3 of RFCthis
ietf-inet-types:ipv6-address-no-zone	54	Section 3.3 of RFCthis
ietf-inet-types:ipv4-address-no-zone	52	Section 3.3 of RFCthis
ietf-inet-types:ip-address-link-local	54, 52 (see union)	Section 3.3 of RFCthis
ietf-inet-types:ipv6-address-link-local	54	Section 3.3 of RFCthis
ietf-inet-types:ipv4-address-link-local	52	Section 3.3 of RFCthis
ietf-inet-types:ip-prefix	54, 52 (see union)	Section 3.3 of RFCthis
ietf-inet-types:ipv6-prefix	54	Section 3.3 of RFCthis
ietf-inet-types:ipv4-prefix	52	Section 3.3 of RFCthis
ietf-inet-types:ip-address-and-prefix	54, 52 (see union)	Section 3.3 of RFCthis
ietf-inet-types:ipv6-address-and-prefix	54	Section 3.3 of RFCthis
ietf-inet-types:ipv4-address-and-prefix	52	Section 3.3 of RFCthis

Table 5: Information Content of Standin File

(*) insert identityrefs from YANG module here

The YANG typename is a name composed of a module name and the name of a type defined there, separated by a colon; names of built-in types do not contain a module name or a colon. The tag number is the number of the CBOR Tag used for representing certain instances of this type. The description is a YANG identity (identified by a SID for an identityref (Section 6.10.1 of [RFC9254]), or, for experimental use, a name for an identityref (Section 6.10.2 of [RFC9254])) defined in a YANG model; the description of the identity either points to a section of the defining document that defines its usage or contains this information outright.

(Note that there is currently no way to assign a SID to a typename; this could be added to the definition of a ".sid" file in [RFC9595].)

Standin Files SHOULD generally be defined in a document such as the present one; they SHOULD NOT be liberally made up for specific applications (unless there are overriding concerns motivating choosing ultimate efficiency over interoperability).

4.2. Defining Stand-In Usage in Schema

Requiring modifications to a YANG model in order to use it with stand-in tags would pose significant deployment hurdles to using stand-in tags.

A YANG model may want to restrict the information content of the YANG-modeled data it describes in such a way that stand-in tags can always be used, e.g., by using date-no-zone in place of date where that is applicable, or by excluding features of a YANG data type that cannot be represented in a stand-in-tag.

ISSUE: Should this document define such restricted types, e.g.:

```
typedef efficient-date-and-time {
  type date-and-time {
    pattern '.*Z'
  }
  description
    "The efficient-date-and-time type is a profile of the
     date-and-time type that is intended to always enable using a
     stand-in tag as per ((this document)), e.g., by not expressing
     a time-zone-offset.
     Not all restrictions that make this possible are expressed in
     the above YANG string pattern.";
}
```

(This particular example is additionally problematic since the usual way to indicate the absence of time zone information in ISO 8601 date-times is using Z as the time zone indicated as specified in Section 3 of [RFC9911], not -00:00 as was previously required by Section 3 of [RFC6991] but not allowed by ISO 8601; see [RFC9557] for references to versions of ISO 8601 and additional discussion of this.)
// Note that this paragraph does not reference ISO 8601 because that
// is complicated and best done by consulting [RFC9557].

4.3. Original stand-ins

The simplest situation is when no intermediate encoders and decoders are involved in the data transfer, therefore the round trip is not legacy. In this case, no conversions are involved and data is validated using the schema extension from the previous section.

4.4. Legacy Round Trip

Producing a stand-in MUST be triggered by schema usage. Intermediate encoders MUST NOT encode stand-ins when no schema is available.

It’s generally not recommended to do a legacy round trip where both the encoder and decoder are converting from and to the legacy representation.

5. Negotiation

(
// TODO: Align with processing models.)

Introducing stand-in tags in YANG-CBOR requires some form of consent between the producer and the consumer of YANG-CBOR information:

* A producer that creates YANG-CBOR containing stand-in tags needs to know whether the consumer supports stand-in tags, and, possibly, which specific stand-in tags it supports. We speak about the _capability_ of a consumer to consume stand-in tags. A producer MUST NOT employ stand-in tags unless it knows about the capabilities of the consumer. A consumer SHOULD indicate its capabilities for consuming stand-in tags.

* A consumer may not want to implement certain legacy text-based representations where more efficient (and easy to implement) stand-in tags are available, i.e., it may use an intolerant decoder. This places a _requirement_ on the producer to use a legacy-eschewing encoder (which therefore needs to have the _capability_ to produce YANG-CBOR where those stand-in tags are

used, in place of legacy representations). Where the consumer employs an intolerant decoder, stand-in tags are _required_ by the consumer: for interoperating with a producer's encoder, this MUST be legacy-eschewing, i.e. it MUST NOT employ legacy representations. A consumer that has requirements for only receiving stand-in tags in place of legacy representations, MUST indicate this to the producer.

ISSUE: Where do we put those two aspects of negotiation?

* NETCONF negotiation
* yang-library
* media-type parameters
* ?

6. Security Considerations

TODO Security

7. IANA Considerations

7.1. New CBOR Tags

In the registry "CBOR Tags" [IANA.cbor-tags], IANA is requested to assign the tag in Table 6.

Tag	Data Item	Semantics	Reference
CPA113	byte string	Expected Later Encoding: colon-separated hexadecimal representation of a byte string	draft-bormann-yang-standin, Section 3.2
CPA114	unsigned integer	Expected Later Encoding: dot-separated decimal representation of network-byte-order ordered bytes of the integer.	
draft-bormann-yang-standin, Section 3.2			

Table 6: New CBOR Tag Defined by this Specification

7.2. stand-in tags?

ISSUE: Do we want to have a separate registry for stand-in files?

7.3. media-type parameters

ISSUE: Should the use of stand-in tags be mentioned in the various YANG-CBOR-based media types (as a media type parameter)?

Compare how application/yang-data+cbor can use parameters id=name/ id=sid to indicate another encoding decision.

8. References

8.1. Normative References

[BCP14] Best Current Practice 14, <https://www.rfc-editor.org/info/bcp14>. At the time of writing, this BCP comprises the following:

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/info/rfc2119>.

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/info/rfc8174>.

[IANA.cbor-tags] IANA, "Concise Binary Object Representation (CBOR) Tags", <https://www.iana.org/assignments/cbor-tags>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/rfc/rfc5952>.

[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>.

[RFC8943] Jones, M., Nadalin, A., and J. Richter, "Concise Binary Object Representation (CBOR) Tags for Date", RFC 8943, DOI 10.17487/RFC8943, November 2020, <https://www.rfc-editor.org/rfc/rfc8943>.

[RFC9164] Richardson, M. and C. Bormann, "Concise Binary Object Representation (CBOR) Tags for IPv4 and IPv6 Addresses and Prefixes", RFC 9164, DOI 10.17487/RFC9164, December 2021, <https://www.rfc-editor.org/rfc/rfc9164>.

[RFC9254] Veillette, M., Ed., Petrov, I., Ed., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", RFC 9254, DOI 10.17487/RFC9254, July 2022, <https://www.rfc-editor.org/rfc/rfc9254>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/rfc/rfc9562>.

[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>.

[RFC9911] Sch旦nw辰lder, J., Ed., "Common YANG Data Types", RFC 9911, DOI 10.17487/RFC9911, December 2025, <https://www.rfc-editor.org/rfc/rfc9911>.

[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>.

8.2. Informative References

[I-D.bormann-cbor-notable-tags] Bormann, C., "Notable CBOR Tags", Work in Progress, Internet-Draft, draft-bormann-cbor-notable-tags-16, 25 February 2026, <https://datatracker.ietf.org/doc/html/draft-bormann-cbor-notable-tags-16>.

[I-D.marenamat-netmod-core-yang-transcoding] Mat卜嬲ka, M., "Transcoding Data Modeled with YANG", Work in Progress, Internet-Draft, draft-marenamat-netmod-core-yang-transcoding-00, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-marenamat-netmod-core-yang-transcoding-00>.

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, DOI 10.17487/RFC6021, October 2010, <https://www.rfc-editor.org/rfc/rfc6021>. This specification is obsoleted by [RFC6991], which is obsoleted by [RFC9911].

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/rfc/rfc6991>. This specification is obsoleted by [RFC9911].

[RFC9542] Eastlake 3rd, D., Ably, J., and Y. Li, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 9542, DOI 10.17487/RFC9542, April 2024, <https://www.rfc-editor.org/rfc/rfc9542>.

[RFC9557] Sharma, U. and C. Bormann, "Date and Time on the Internet: Timestamps with Additional Information", RFC 9557, DOI 10.17487/RFC9557, April 2024, <https://www.rfc-editor.org/rfc/rfc9557>.

List of Tables

Acknowledgments

TODO acknowledge.

Authors' Addresses

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

Maria Matejka
CZ.NIC
Milesovska 1136/5
13000 Praha
Czechia
Email: maria.matejka@nic.cz, mq@jmq.cz