Workgroup: CBOR Working Group
Internet-Draft:
draft-ietf-cbor-network-addresses-05
Published: 12 July 2021
Intended Status: Standards Track
Expires: 13 January 2022
Authors: M. Richardson — Sandelman Software Works
C. Bormann — Universität Bremen TZI

CBOR tags for IPv4 and IPv6 addresses and prefixes

Abstract

This document describes two CBOR Tags to be used with IPv4 and IPv6 addresses and prefixes.

RFC-EDITOR-please remove: This work is tracked at https://github.com/cbor-wg/cbor-network-address

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[RFC8949] defines a number of CBOR Tags for common items.

Tag 260 and tag 261 was later defined through IANA. These tags cover addresses (260), and prefixes (261). Tag 260 distinguishes between IPv4, IPv6 and Ethernet through the length of the byte string only. Tag 261 was not documented well enough to be used.

The present specification achieves an explicit indication of IPv4 or IPv6, and the possibility to omit trailing zeroes.

This document provides a format for IPv6 and IPv4 addresses, prefixes, and addresses with prefixes. Prefixes MUST omit trailing zeroes in the address. Due to the complexity of testing the value of omitting trailing zeros for addresses was considered non-essential and support for that was removed in this specification.

This document does not deal with 6 or 8-byte Ethernet addressees.

2. Terminology

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.

3. Protocol

These tags can applied to byte strings to represent a single address.

When applied to an array that starts with a number, they represent a CIDR-style prefix of that length. When a byte string (without prefix) appears in a context where a prefix is expected, then it is to be assumed that all bits are relevant. That is, for IPv4, a /32 is implied, and for IPv6, a /128 is implied.

When applied to an array that starts with a byte string, that stands for an IP address, followed by the bit length of a prefix built out of the first length bits of the address.

3.1. IPv6

IANA has allocated tag 54 for IPv6 uses. (Note that this is the ASCII code for '6'.)

An IPv6 address is to be encoded as a sixteen-byte byte string (Section 3.1 of [RFC8949], major type 2), enclosed in Tag number 54.

An IPv6 prefix, such as 2001:db8:1234::/48 is to be encoded as a two element array, with the length of the prefix first. Trailing zero bytes MUST be omitted.

For example:

```
54([ 48, h'20010db81234'])
```

An IPv6 address combined with a prefix length, such as being used for configuring an interface, is to be encoded as a two element array, with the (full-length) IPv6 address first and the length of the associated network the prefix next.

For example:

```
54([h'20010db81234DEEDBEEFCAFEFACEFEED', 56])
```

Note that the address-with-prefix form can be reliably distinguished from the prefix form only in the sequence of the array elements.

3.2. IPv4

IANA has allocated tag 52 for IPv4 uses. (Note that this is the ASCII code for '4'.)

An IPv4 address is to be encoded as a four-byte byte string (Section 3.1 of [RFC8949], major type 2), enclosed in Tag number 52.

An IPv4 prefix, such as 192.0.2.0/24 is to be encoded as a two element array, with the length of the prefix first. Trailing zero bytes MUST be omitted.

For example:

```
52([ 24, h'C00002'])
```

An IPv4 address combined with a prefix length, such as being used for configuring an interface, is to be encoded as a two element array, with the (full-length) IPv4 address first and the length of the associated network the prefix next.

For example, 192.0.2.1/24 is to be encoded as a two element array, with the length of the prefix (implied 192.0.2.0/24) last.

```
52([ h'C0000201', 24])
```

Note that the address-with-prefix form can be reliably distinguished from the prefix form only in the sequence of the array elements.

4. Encoder Consideration for prefixes

An encoder may omit as many right-hand (trailing) bytes which are all zero as it wishes.

There is no relationship between the number of bytes omitted and the prefix length. For instance, the prefix 2001:db8::/64 is optimally encoded as:

```
54([64, h'20010db8'])
```

An encoder MUST take care to set all trailing bits to zero. While decoders are expected to ignore them, such garbage entities could be used as a covert channel, or may reveal the state of what would otherewise be private memory contents. So for example, 2001:db8:1230::/44 MUST be encoded as:

```
52([44, h'20010db81230'])
```

even though variations like:

```
54([44, h'20010db81233'])  WRONG
54([45, h'20010db8123f'])  WRONG
```

would be parsed in the exact same way.

The same considerations apply to IPv4 prefixes.

5. Decoder Considerations for prefixes

A decoder MUST consider all bits to the right of the prefix length to be zero.

A decoder MUST handle the case where a prefix length specifies that more bits are relevant than are actually present in the byte-string. As a pathological case, ::/128 can be encoded as

```
54([128, h''])
```

A recommendation for implementation is to first create an array of 16 (or 4) bytes in size, set it all to zero.

Then looking at the length of the included byte-string, and of the prefix-length, rounded up to the next multiple of 8, and taking whichever is smaller, copy that many bytes from the byte-string into the array.

Finally, looking at the last three bits of the prefix-length (that is, the prefix-length modulo 8), use a static array of 8 values to force the lower bits, non-relevant bits to zero.

A particularly paranoid decoder could examine the lower non-relevant bits to determine if they are non-zero, and reject the prefix. This would detect non-compliant encoders, or a possible covert channel.

6. CDDL

For use with CDDL [RFC8610], the typenames defined in Figure 1 are recommended:

```
ip-address-or-prefix = ipv6-address-or-prefix /
                       ipv4-address-or-prefix

ipv6-address-or-prefix = #6.54(ipv6-address /
                               ipv6-address-with-prefix /
                               ipv6-prefix)
ipv4-address-or-prefix = #6.52(ipv4-address /
                               ipv4-address-with-prefix /
                               ipv4-prefix)

ipv6-address = bytes .size 16
ipv4-address = bytes .size 4

ipv6-address-with-prefix = [ipv6-address, ipv6-prefix-length]
ipv4-address-with-prefix = [ipv4-address, ipv4-prefix-length]

ipv6-prefix-length = 0..128
ipv4-prefix-length = 0..32

ipv6-prefix = [ipv6-prefix-length, ipv6-prefix-bytes]
ipv4-prefix = [ipv4-prefix-length, ipv4-prefix-bytes]

ipv6-prefix-bytes = bytes .size (uint .le 16)
ipv4-prefix-bytes = bytes .size (uint .le 4)
```

Figure 1

7. Security Considerations

Identifying which byte sequences in a protocol are addresses may allow an attacker or eavesdropper to better understand what parts of a packet to attack.

Reading the relevant RFC may provide more information, so it would seem that any additional security that was provided by not being able to identify what are IP addresses falls into the security by obscurity category.

The right-hand bits of the prefix, after the prefix-length, are ignored by this protocol. A malicious party could use them to transmit covert data in a way that would not affect the primary use of this encoding. Such abuse would be detected by examination of the raw protocol bytes. Users of this encoding should be aware of this possibility.

8. IANA Considerations

IANA has allocated two tags from the Specification Required area of the Concise Binary Object Representation (CBOR) Tags:

8.1. Tag 54 - IPv6

Data Item: byte string or array
Semantics: IPv6, [prefixlen,IPv6], [IPv6,prefixpart]

8.2. Tag 52 - IPv4

Data Item: byte string or array
Semantics: IPv4, [prefixlen,IPv4], [IPv4,prefixpart]

9. 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/info/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/info/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/info/rfc8610>.

[RFC8949] 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>.

Appendix A. Changelog

This section is to be removed before publishing as an RFC.

*03

*02

*01 added security considerations about covert channel

Acknowledgements

none yet

Authors' Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Carsten Bormann
Universität Bremen TZI
Germany

Email: cabo@tzi.org