Workgroup: Internet Engineering Task Force
Internet-Draft:
draft-evan-amateur-radio-ipv6-04
Published: 16 February 2023
Intended Status: Experimental
Expires: 20 August 2023
A E. Pratten
u
t
h
o
r
s
:

# A Method for Deriving Stable IPv6 Interface Identifiers from Amateur Radio Callsigns

## Abstract

This document defines a method for generating stable IPv6 Interface Identifiers for amateur packet radio nodes. This method is meant to be an alternative to hardware address based Interface Identifier generation such that the benefits of stable addressing may be achieved even on nodes that have unstable, changing, or experimental networking hardware. Instead of a physically-derived address, this method utilizes an amateur radio node's government-assigned callsign as the basis for its Interface Identifier.

## 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 20 August 2023.

## Copyright Notice

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

This document specifies the steps an amateur packet radio node takes in order to generate a stable and unique IPv6 Interface Identifier (IID) [RFC2460]. The resulting Interface Identifier SHALL be used in conjunction with processes such as (but not limited to) Stateless Address Autoconfiguration (SLAAC) [RFC4862], DHCPv6 [RFC3315], or manual configuration to configure IPv6 connectivity on the node.

Callsign-derived Interface Identifier generation requires minimal manual configuration, and when paired with SLAAC may allow a mobile amateur packet radio node to automatically connect to, and communicate with any compliant amateur radio network provided that the node has been configured with a callsign, and is communicating on the correct radio frequency.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 3. Foreword on Node Identification

Amateur packet radio nodes generally identify themselves with a string of ASCII characters comprised of:

1. The station's government-assigned callsign

2. A dash (-)

3. A number ranging from 0 to 15, inclusive. This will be referred to for the remainder of this document as the node's "ID". Stations that do not use a node ID generally will use a "0" in this place.

For example, a node operated under the callsign "VA3ZZA" with the node ID of "5" would identify itself on-air as "VA3ZZA-5".

## 4. The Algorithm

To determine a 64 bit long [RFC4291] Interface Identifier for an amateur packet radio node in conformance with this specification, the following steps MUST be be taken:

1. Set the lest significant 4 bits of the Interface Identifier to the node's ID number.

2. If the callsign is less than or equal to 9 characters in length:

    1. Set the most significant bit of the Interface Identifier (the hash bit) to 0. This indicates that "Direct Encoding" is in use.

    2. Using the "Direct Encoding Charset" defined below, pack the UPPERCASE callsign into the middle 59 bits of the Interface Identifier. Callsigns shorter than 9 characters must be right-padded with spaces. This means that the callsign "VA3ZZA" would be encoded as "VA3ZZA " (3 trailing spaces).

3. If the callsign is greater than 9 characters in length:

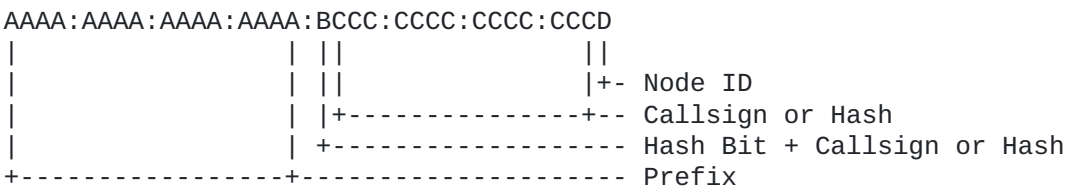
    1. Set the most significant bit of the Interface Identifier (the hash bit) to 1. This indicates that hashing is in use.

    2. SHA-256 hash the callsign

    3. Bitwise AND the hash with the 64 bit value 0x7FFFFFFFFFFFFFF0

    4. Bitwise OR the result to the Interface Identifier

Addresses generated using this method will look like the following:




### 4.1. Direct Encoding Charset

When directly packing a shorter callsign into the Interface Identifier, the following charset MUST be used:

```
<space> - 000000  S - 010011
      A - 000001  T - 010100
      B - 000010  U - 010101
      C - 000011  V - 010110
      D - 000100  W - 010111
      E - 000101  X - 011000
      F - 000110  Y - 011001
      G - 000111  Z - 011010
      H - 001000  0 - 011011
      I - 001001  1 - 011100
      J - 001010  2 - 011101
      K - 001011  3 - 011110
      L - 001100  4 - 011111
      M - 001101  5 - 100000
      N - 001110  6 - 100001
      O - 001111  7 - 100010
      P - 010000  8 - 100011
      Q - 010001  9 - 100100
      R - 010010  / - 100101
```

Each character translates to 6 bits. This allows for more efficient packing of longer callsigns.

### 4.2. An Example Implementation

```
import hashlib

CHARSET = " ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789/"

def encode(callsign: str, node_id: int = 0) -> int:

    # The output is a 64-bit integer
    output = 0x0000000000000000

    # The right-most 4 bits are the node ID
    output |= node_id & 0x0F

    # If the callsign is longer than 9 characters,
    # perform a hash operation
    if len(callsign) > 9:
        # Set the hash bit
        output |= 0x8000000000000000

        # Hash the callsign
        hashed = hashlib.sha256(callsign.encode("ascii")).digest()

        # Truncate the hash to 59 bits
        output |= int.from_bytes(hashed, "big") & 0x7FFFFFFFFFFFFFF0

    else:
        # Text-transform and right-pad the callsign
        # with spaces as needed
        callsign = callsign.upper().ljust(9, " ")

        # Fill in the remaining bits via the charset.
        # The leftmost callsign character will be the
        # leftmost 6 bits of the output
        for i in range(9):
            six_bits = CHARSET.index(callsign[i]) & 0x3F
            output |= six_bits << (58 - (i * 6))

    return output
```

### 4.3. Using the Interface Identifier

It is recommended to use the generated Interface Identifier with Stateless Address Autoconfiguration to automatically determine the node's IPv6 prefix and establish routes to other hosts in a radio network.

While SLAAC is the recommended method of configuration, it is not required. Amateur packet radio networks may also use alternate address configuration mechanisms such as DHCPv6 or manual configuration as the participants see fit.

### 4.4. Resolving SLAAC Duplicate Address Detection Conflicts

As a side effect of basing the Interface Identifier on an existing globally unique identifier, DAD [RFC4862] conflicts should be non-existent for permanent callsigns.

If a conflict is encountered when using a permanent callsign, the node operator SHOULD contact the operator of the offending station, and/or the appropriate regulatory authority about possible unauthorized use of a callsign.

Due to the need for hashing of longer temporary callsigns, a hash collision may occur. If this happens, the node operator SHOULD append a suffix to the node's callsign. Callsign suffixes are arbitrary strings that start with "/" and contain characters A-Z and 0-9. For example, adding a suffix of "IETF" to "VA3ZZA" could result in a callsign like "VA3ZZA/IETF".

### 4.5. Benefits of this method

This method of Interface Identifier generation has the following benefits:

* Callsigns are uniquely assigned to stations by existing governing bodies. Using them as the basis of address creation will ensure a unique seed or basis for the Interface Identifier.

* Hashing long callsigns instead of trying to directly encode them allows support for excessively long temporary callsigns.

* Encoding the station ID in the final nibble of the address allows for up to 16 nodes under the same callsign to be assigned addresses within the same /124. This allows address-based access control logic to operate on a whole callsign (first 60 bits of the interface ID) at once, an ability not possible if the ID was also hashed.

* Stations operating with permanent callsigns will have the benefit of their callsign being automatically transmitted with every packet (encoded in the source address), in a format that satisfies many governments' station identification requirements.

### 4.6. Drawbacks of this method

* Stations operating with excessively long temporary callsigns will need to separately identify their transmissions as their callsign will not be transmitted by default with each packet.

* Stations operating with excessively long temporary callsigns may encounter hash collisions with other stations (although extremely unlikely). In the event of this happening, the node operator will be required to perform additional reconfiguration of their node.

## 5. Privacy Considerations

The International Telecommunication Union requires all stations operating in the amateur service to self-identify when transmitting.

Various countries also impose further requirements such as the interval and method by which stations must identify themselves.

The legal requirement to identify all transmissions nullifies any privacy benefits gained from other privacy-aware addressing methods.

## 6. IANA Considerations

This memo includes no request to IANA.

## 7. Security Considerations

This document should not affect the security of the Internet.

## 8. References

### 8.1. Normative References


**[RFC4291]** Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/rfc/rfc4291>.

**[RFC4862]** Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

**[RFC2460]** Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

**[RFC3315]** Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <https://www.rfc-editor.org/info/rfc3315>.

**[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>.


## Author's Address


Evan Pratten

Email: evan@ewpratten.com
URI: https://ewpratten.com