Internet Engineering Task Force (IETF)
Request for Comments: 5952
Updates: 4291
Category: Standards Track
ISSN: 2070-1721

S. Kawamura
NEC BIGLOBE, Ltd.
M. Kawashima
NEC AccessTechnica, Ltd.

August 2010

A Recommendation for IPv6 Address Text Representation

Abstract

As IPv6 deployment increases, there will be a dramatic increase in the need to use IPv6 addresses in text. While the IPv6 address architecture in Section 2.2 of RFC 4291 describes a flexible model for text representation of an IPv6 address, this flexibility has been causing problems for operators, system engineers, and users. This document defines a canonical textual representation format. It does not define a format for internal storage, such as within an application or database. It is expected that the canonical format will be followed by humans and systems when representing IPv6 addresses as text, but all implementations must accept and be able to handle any legitimate RFC 4291 format.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5952.

Table of Contents

1. Introduction 4
1.1. Requirements Language 4
2. Text Representation Flexibility of RFC 4291 4
2.1. Leading Zeros in a 16-Bit Field 4
2.2. Zero Compression 5
2.3. Uppercase or Lowercase 6
3. Problems Encountered with the Flexible Model 6
3.1. Searching 6
3.1.1. General Summary 6
3.1.2. Searching Spreadsheets and Text Files 6
3.1.3. Searching with Whois 6
3.1.4. Searching for an Address in a Network Diagram 7
3.2. Parsing and Modifying 7
3.2.1. General Summary 7
3.2.2. Logging 7
3.2.3. Auditing: Case 1 8
3.2.4. Auditing: Case 2 8
3.2.5. Verification 8
3.2.6. Unexpected Modifying 8
3.3. Operating 8
3.3.1. General Summary 8
3.3.2. Customer Calls 9
3.3.3. Abuse 9
3.4. Other Minor Problems 9
3.4.1. Changing Platforms 9
3.4.2. Preference in Documentation 9
3.4.3. Legibility 9
4. A Recommendation for IPv6 Text Representation 10
4.1. Handling Leading Zeros in a 16-Bit Field 10
4.2. "::" Usage 10
4.2.1. Shorten as Much as Possible 10
4.2.2. Handling One 16-Bit 0 Field 10
4.2.3. Choice in Placement of "::" 10
4.3. Lowercase 10
5. Text Representation of Special Addresses 11
6. Notes on Combining IPv6 Addresses with Port Numbers 11
7. Prefix Representation 12
8. Security Considerations 12
9. Acknowledgements 12
10. References 12
10.1. Normative References 12
10.2. Informative References 13
Appendix A. For Developers 14

1. Introduction

A single IPv6 address can be text represented in many ways. Examples are shown below.

2001:db8:0:0:1:0:0:1

2001:0db8:0:0:1:0:0:1

2001:db8::1:0:0:1

2001:db8::0:1:0:0:1

2001:0db8::1:0:0:1

2001:db8:0:0:1::1

2001:db8:0000:0:1::1

2001:DB8:0:0:1::1

All of the above examples represent the same IPv6 address. This flexibility has caused many problems for operators, systems engineers, and customers. The problems are noted in Section 3. A canonical representation format to avoid problems is introduced in Section 4.

1.1. Requirements Language

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].

2. Text Representation Flexibility of RFC 4291

Examples of flexibility in Section 2.2 of [RFC4291] are described below.

2.1. Leading Zeros in a 16-Bit Field

> 'It is not necessary to write the leading zeros in an individual field.'

Conversely, it is also not necessary to omit leading zeros. This means that it is possible to select from representations such as those in the following example. The final 16-bit field is different, but all of these addresses represent the same address.

2001:db8:aaaa:bbbb:cccc:dddd:eeee:0001

2001:db8:aaaa:bbbb:cccc:dddd:eeee:001

2001:db8:aaaa:bbbb:cccc:dddd:eeee:01

2001:db8:aaaa:bbbb:cccc:dddd:eeee:1

2.2. Zero Compression

'A special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros.'

It is possible to select whether or not to omit just one 16-bit 0 field.

2001:db8:aaaa:bbbb:cccc:dddd::1

2001:db8:aaaa:bbbb:cccc:dddd:0:1

In cases where there is more than one field of only zeros, there is a choice of how many fields can be shortened.

2001:db8:0:0:0::1

2001:db8:0:0::1

2001:db8:0::1

2001:db8::1

In addition, Section 2.2 of [RFC4291] notes,

'The "::" can only appear once in an address.'

This gives a choice on where in a single address to compress the zero.

2001:db8::aaaa:0:0:1

2001:db8:0:0:aaaa::1

2.3. Uppercase or Lowercase

[RFC4291] does not mention any preference of uppercase or lowercase.

2001:db8:aaaa:bbbb:cccc:dddd:eeee:aaaa

2001:db8:aaaa:bbbb:cccc:dddd:eeee:AAAA

2001:db8:aaaa:bbbb:cccc:dddd:eeee:AaAa

3. Problems Encountered with the Flexible Model

3.1. Searching

3.1.1. General Summary

A search of an IPv6 address if conducted through a UNIX system is usually case sensitive and extended options that allow for regular expression use will come in handy. However, there are many applications in the Internet today that do not provide this capability. When searching for an IPv6 address in such systems, the system engineer will have to try each and every possibility to search for an address. This has critical impacts, especially when trying to deploy IPv6 over an enterprise network.

3.1.2. Searching Spreadsheets and Text Files

Spreadsheet applications and text editors on GUI systems rarely have the ability to search for text using regular expression. Moreover, there are many non-engineers (who are not aware of case sensitivity and regular expression use) that use these applications to manage IP addresses. This has worked quite well with IPv4 since text representation in IPv4 has very little flexibility. There is no incentive to encourage these non-engineers to change their tool or learn regular expression when they decide to go dual-stack. If the entry in the spreadsheet reads, 2001:db8::1:0:0:1, but the search was conducted as 2001:db8:0:0:1::1, this will show a result of no match. One example where this will cause a problem is, when the search is being conducted to assign a new address from a pool, and a check is being done to see if it is not in use. This may cause problems for the end-hosts or end-users. This type of address management is very often seen in enterprise networks and ISPs.

3.1.3. Searching with Whois

The "whois" utility is used by a wide range of people today. When a record is set to a database, one will likely check the output to see if the entry is correct. If an entity was recorded as 2001:db8::/48,

but the whois output showed 2001:0db8:0000::/48, most non-engineers would think that their input was wrong and will likely retry several times or make a frustrated call to the database hostmaster. If there was a need to register the same prefix on different systems, and each system showed a different text representation, this would confuse people even more. Although this document focuses on addresses rather than prefixes, it is worth mentioning the prefix problems because the problems encountered with addresses and prefixes are mostly equal.

3.1.4. Searching for an Address in a Network Diagram

Network diagrams and blueprints often show what IP addresses are assigned to a system devices. In times of trouble shooting there may be a need to search through a diagram to find the point of failure (for example, if a traceroute stopped at 2001:db8::1, one would search the diagram for that address). This is a technique quite often in use in enterprise networks and managed services. Again, the different flavors of text representation will result in a time-consuming search leading to longer mean times to restoration (MTTR) in times of trouble.

3.2. Parsing and Modifying

3.2.1. General Summary

With all the possible methods of text representation, each application must include a module, object, link, etc. to a function that will parse IPv6 addresses in a manner such that no matter how it is represented, they will mean the same address. Many system engineers who integrate complex computer systems for corporate customers will have difficulties finding that their favorite tool will not have this function, or will encounter difficulties such as having to rewrite their macros or scripts for their customers.

3.2.2. Logging

If an application were to output a log summary that represented the address in full (such as 2001:0db8:0000:0000:1111:2222:3333:4444), the output would be highly unreadable compared to the IPv4 output. The address would have to be parsed and reformed to make it useful for human reading. Sometimes logging for critical systems is done by mirroring the same traffic to two different systems. Care must be taken so that no matter what the log output is, the logs should be parsed so they are equivalent.

3.2.3. Auditing: Case 1

When a router or any other network appliance machine configuration is audited, there are many methods to compare the configuration information of a node. Sometimes auditing will be done by just comparing the changes made each day. In this case, if configuration was done such that 2001:db8::1 was changed to 2001:0db8:0000:0000:0000:0000:0000:0001 just because the new engineer on the block felt it was better, a simple diff will show that a different address was configured. If this was done on a wide scale network, people will be focusing on 'why the extra zeros were put in' instead of doing any real auditing. Lots of tools are just plain diffs that do not take into account address representation rules.

3.2.4. Auditing: Case 2

Node configurations will be matched against an information system that manages IP addresses. If output notation is different, there will need to be a script that is implemented to cover for this. The result of an SNMP GET operation, converted to text and compared to a textual address written by a human is highly unlikely to match on the first try.

3.2.5. Verification

Some protocols require certain data fields to be verified. One example of this is X.509 certificates. If an IPv6 address field in a certificate was incorrectly verified by converting it to text and making a simple textual comparison to some other address, the certificate may be mistakenly shown as being invalid due to a difference in text representation methods.

3.2.6. Unexpected Modifying

Sometimes, a system will take an address and modify it as a convenience. For example, a system may take an input of 2001:0db8:0::1 and make the output 2001:db8::1. If the zeros were input for a reason, the outcome may be somewhat unexpected.

3.3. Operating

3.3.1. General Summary

When an operator sets an IPv6 address of a system as 2001:db8:0:0:1:0:0:1, the system may take the address and show the configuration result as 2001:DB8::1:0:0:1. Someone familiar with IPv6 address representation will know that the right address is set, but not everyone may understand this.

3.3.2. Customer Calls

When a customer calls to inquire about a suspected outage, IPv6 address representation should be handled with care. Not all customers are engineers, nor do they have a similar skill level in IPv6 technology. The network operations center will have to take extra steps to humanly parse the address to avoid having to explain to the customers that 2001:db8:0:1::1 is the same as 2001:db8::1:0:0:0:1. This is one thing that will never happen in IPv4 because IPv4 addresses cannot be abbreviated.

3.3.3. Abuse

Network abuse reports generally include the abusing IP address. This 'reporting' could take any shape or form of the flexible model. A team that handles network abuse must be able to tell the difference between a 2001:db8::1:0:1 and 2001:db8:1::0:1. Mistakes in the placement of the "::" will result in a critical situation. A system that handles these incidents should be able to handle any type of input and parse it in a correct manner. Also, incidents are reported over the phone. It is unnecessary to report if the letter is uppercase or lowercase. However, when a letter is spelled uppercase, people tend to specify that it is uppercase, which is unnecessary information.

3.4. Other Minor Problems

3.4.1. Changing Platforms

When an engineer decides to change the platform of a running service, the same code may not work as expected due to the difference in IPv6 address text representation. Usually, a change in a platform (e.g., Unix to Windows, Cisco to Juniper) will result in a major change of code anyway, but flexibility in address representation will increase the work load.

3.4.2. Preference in Documentation

A document that is edited by more than one author may become harder to read.

3.4.3. Legibility

Capital case D and 0 can be quite often misread. Capital B and 8 can also be misread.

4. A Recommendation for IPv6 Text Representation

A recommendation for a canonical text representation format of IPv6 addresses is presented in this section. The recommendation in this document is one that complies fully with [RFC4291], is implemented by various operating systems, and is human friendly. The recommendation in this section SHOULD be followed by systems when generating an address to be represented as text, but all implementations MUST accept and be able to handle any legitimate [RFC4291] format. It is advised that humans also follow these recommendations when spelling an address.

4.1. Handling Leading Zeros in a 16-Bit Field

Leading zeros MUST be suppressed. For example, 2001:0db8::0001 is not acceptable and must be represented as 2001:db8::1. A single 16-bit 0000 field MUST be represented as 0.

4.2. "::" Usage

4.2.1. Shorten as Much as Possible

The use of the symbol "::" MUST be used to its maximum capability. For example, 2001:db8:0:0:0:0:2:1 must be shortened to 2001:db8::2:1. Likewise, 2001:db8::0:1 is not acceptable, because the symbol "::" could have been used to produce a shorter representation 2001:db8::1.

4.2.2. Handling One 16-Bit 0 Field

The symbol "::" MUST NOT be used to shorten just one 16-bit 0 field. For example, the representation 2001:db8:0:1:1:1:1:1 is correct, but 2001:db8::1:1:1:1:1 is not correct.

4.2.3. Choice in Placement of "::"

When there is an alternative choice in the placement of a "::", the longest run of consecutive 16-bit 0 fields MUST be shortened (i.e., the sequence with three consecutive zero fields is shortened in 2001:0:0:1:0:0:0:1). When the length of the consecutive 16-bit 0 fields are equal (i.e., 2001:db8:0:0:1:0:0:1), the first sequence of zero bits MUST be shortened. For example, 2001:db8::1:0:0:1 is correct representation.

4.3. Lowercase

The characters "a", "b", "c", "d", "e", and "f" in an IPv6 address MUST be represented in lowercase.

5. Text Representation of Special Addresses

Addresses such as IPv4-Mapped IPv6 addresses, ISATAP [RFC5214], and IPv4-translatable addresses [ADDR-FORMAT] have IPv4 addresses embedded in the low-order 32 bits of the address. These addresses have a special representation that may mix hexadecimal and dot decimal notations. The decimal notation may be used only for the last 32 bits of the address. For these addresses, mixed notation is RECOMMENDED if the following condition is met: the address can be distinguished as having IPv4 addresses embedded in the lower 32 bits solely from the address field through the use of a well-known prefix. Such prefixes are defined in [RFC4291] and [RFC2765] at the time of this writing. If it is known by some external method that a given prefix is used to embed IPv4, it MAY be represented as mixed notation. Tools that provide options to specify prefixes that are (or are not) to be represented as mixed notation may be useful.

There is a trade-off here where a recommendation to achieve an exact match in a search (no dot decimals whatsoever) and a recommendation to help the readability of an address (dot decimal whenever possible) does not result in the same solution. The above recommendation is aimed at fixing the representation as much as possible while leaving the opportunity for future well-known prefixes to be represented in a human-friendly manner as tools adjust to newly assigned prefixes.

The text representation method noted in Section 4 should be applied for the leading hexadecimal part (i.e., ::ffff:192.0.2.1 instead of 0:0:0:0:0:ffff:192.0.2.1).

6. Notes on Combining IPv6 Addresses with Port Numbers

There are many different ways to combine IPv6 addresses and port numbers that are represented in text. Examples are shown below.

- [2001:db8::1]:80
- 2001:db8::1:80
- 2001:db8::1.80
- 2001:db8::1 port 80
- 2001:db8::1p80
- 2001:db8::1#80

The situation is not much different in IPv4, but the most ambiguous case with IPv6 is the second bullet. This is due to the "::"usage in

IPv6 addresses. This style is NOT RECOMMENDED because of its ambiguity. The [] style as expressed in [RFC3986] SHOULD be employed, and is the default unless otherwise specified. Other styles are acceptable when there is exactly one style for the given context and cross-platform portability does not become an issue. For URIs containing IPv6 address literals, [RFC3986] MUST be followed, as well as the rules defined in this document.

7. Prefix Representation

Problems with prefixes are the same as problems encountered with addresses. The text representation method of IPv6 prefixes should be no different from that of IPv6 addresses.

8. Security Considerations

This document notes some examples where IPv6 addresses are compared in text format. The example on Section 3.2.5 is one that may cause a security risk if used for access control. The common practice of comparing X.509 data is done in binary format.

9. Acknowledgements

The authors would like to thank Jan Zorz, Randy Bush, Yuichi Minami, and Toshimitsu Matsuura for their generous and helpful comments in kick starting this document. We also would like to thank Brian Carpenter, Akira Kato, Juergen Schoenwaelder, Antonio Querubin, Dave Thaler, Brian Haley, Suresh Krishnan, Jerry Huang, Roman Donchenko, Heikki Vatiainen, Dan Wing, and Doug Barton for their input. Also, a very special thanks to Ron Bonica, Fred Baker, Brian Haberman, Robert Hinden, Jari Arkko, and Kurt Lindqvist for their support in bringing this document to light in IETF working groups.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2765] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

10.2. Informative References

[ADDR-FORMAT] Bao, C., "IPv6 Addressing of IPv4/IPv6 Translators", Work in Progress, July 2010.

[RFC4038] Shin, M-K., Hong, Y-G., Hagino, J., Savola, P., and E. Castro, "Application Aspects of IPv6 Transition", RFC 4038, March 2005.

[RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, March 2008.

Appendix A. For Developers

We recommend that developers use display routines that conform to these rules. For example, the usage of getnameinfo() with flags argument NI_NUMERICHOST in FreeBSD 7.0 will give a conforming output, except for the special addresses notes in Section 5. The function inet_ntop() of FreeBSD7.0 is a good C code reference, but should not be called directly. See [RFC4038] for details.

Authors' Addresses

Seiichi Kawamura
NEC BIGLOBE, Ltd.
14-22, Shibaura 4-chome
Minatoku, Tokyo 108-8558
JAPAN

Phone: +81 3 3798 6085
EMail: kawamucho@mesh.ad.jp

Masanobu Kawashima
NEC AccessTechnica, Ltd.
800, Shimomata
Kakegawa-shi, Shizuoka 436-8501
JAPAN

Phone: +81 537 23 9655
EMail: kawashimam@necat.nec.co.jp