RFC 5952 : A Recommendation for IPv6 Address Text Representation

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 août 2010

Date de publication du RFC : Août 2010

https://www.bortzmeyer.org/5952.html

Comme pour IPv4, mais de manière bien plus commune, les adresses IPv6 ont plusieurs représentations <https://www.bortzmeyer.org/representation-texte.html> possibles. Ainsi (section 1 du RFC), l'adresse 2001:db8:0:0:1:0:0:1 peut aussi s'écrire 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 ou 2001:DB8:0:0:1::1. Cette variété peut dans certains cas être cause de confusion, notre RFC propose donc **une** forme recommandée (ici, 2001:db8::1:0:0:1).

La syntaxe des adresses IPv6 est fixée le RFC 4291 [1], section 2.2. Cette syntaxe est très souple, et venait sans format recommandé, « canonique ». La section 2 liste les points où le RFC 4291 laissait le choix :

— Possibilité d'indiquer (ou pas) les zéros initiaux dans chaque champ. 2001:db8:0:0:1:0:0:1 et 2001:0db8:0:0:1:0:0:1 sont ainsi équivalentes (deuxième champ).
— Possibilité de compression des champs nuls consécutifs en les remplaçant par ::. 2001:db8:0:0:0:0:0:1 et 2001:db8::1 sont ainsi la même adresse. Si le :: peut apparaitre à deux endroits, le RFC 4291 impose, pour éviter toute ambiguité, de ne le mettre qu'une fois mais sans préciser où. Ainsi, 2001:db8::aaaa:0:0:1 et 2001:db8:0:0:aaaa::1 sont la même adresse.
— Pour les chiffres hexadécimaux qui sont des lettres de A à F, on peut utiliser n'importe quelle casse.

Est-ce que cela pose vraiment des problèmes ? Parfois, dit notre RFC, dont la section 3 liste les problèmes **possibles** (sans les hiérarchiser, ce que je regrette car certains cas semblent quand même assez rares en pratique). Premier problème, la recherche d'une adresse dans un fichier texte ou bien avec un tableur. Si on utilise aveuglément grep sur ses fichiers, on risque de ne pas trouver l'adresse IP (avec grep, il faudrait utiliser une expression rationnelle mais les tableurs, par exemple, n'en disposent pas forcément et leurs utilisateurs peuvent ne pas y penser). Notez qu'avec un SGBD qui dispose d'un type « adresse IP <http://www.postgresql.org/docs/current/interactive/datatype-net-types.html> » comme PostgreSQL, ce problème n'existe pas, le SGBD ne traite pas l'adresse comme du texte :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4291.txt

```
essais=> CREATE TABLE Machines (name TEXT, address INET);
CREATE TABLE
essais=> INSERT INTO Machines VALUES ('gandalf', '2001:db8::cafe:0:1');
INSERT 0 1
essais=> INSERT INTO Machines VALUES ('saroumane', '2001:db8::bad:0:1');
INSERT 0 1
essais=> SELECT * FROM Machines WHERE address = '2001:DB8:0:0:0:CAFE:0:1';
  name   |      address
---------+--------------------
 gandalf | 2001:db8::cafe:0:1
(1 row)
```

On voit que, malgré une représentation toute différente de l'adresse, la machine `gandalf` a bien été trouvée. Si tout le monde utilisait des logiciels de gestion d'adresses IP bâtis sur ce principe, il n'y aurait pas de problème. Mais, comme le note le RFC, les méthodes « du pauvre » à base de grep ou d'Excel sont courantes. (Voir aussi les sections 3.1.3 et 3.1.4, moins convaincantes à mon avis.)

Des problèmes analogues surviennent lorsqu'on veut écrire un programme capable d'analyser des adresses IPv6 sous toutes leurs formes légales (section 3.2.1, avec laquelle je ne suis guère d'accord : il existe des bibliothèques toutes faites pour cela, dans tous les langages, comme `inet_pton()` pour C, et celui qui réinvente la roue en écrivant un analyseur tout neuf en PHP ou Visual Basic mérite les ennuis qu'il aura).

Le RFC cite d'autres problèmes possibles, comme le fait qu'un module de journalisation qui afficherait les adresses IP sous leur forme longue (comme `2001:0db8:0:0:1:0:0:1`) produirait des historiques peu lisibles, ou comme le fait qu'un mécanisme d'audit (par exemple avec un outil comme diff) ou de gestion de versions qui analyserait des changements dans la configuration d'un routeur pourrait croire à tort qu'il y a un changement lorsqu'une adresse IPv6 passe d'une forme à une autre (section 3.2.3). Bien d'autres points analogues sont pointés du doigt par le RFC.

Enfin, le jeu de caractères étendu de l'hexadécimal entraine un risque de confusion entre D et 0, de même qu'entre B et 8 (section 3.4.3).

Quelle solution propose donc notre RFC ? La section 4 est la partie normative du document : elle définit une **forme canonique** qui devrait être suivie systématiquement lorsqu'une adresse IPv6 est affichée. Rien de nouveau dans cette forme, qui est déjà celle choisie par la plupart des logiciels, à part sa désignation comme forme canonique officielle. Attention, cette obligation ne porte que sur la **sortie** d'adresses IPv6, en **entrée**, un logiciel conforme à la norme IPv6 doit toujours accepter les différentes syntaxes.

Une conséquence de cette existence d'une forme canonique est que le logiciel n'affichera donc pas toujours ce qu'on lui a indiqué. Pour reprendre l'exemple PostgreSQL :

```
essais=> INSERT INTO Machines VALUES ('galadriel',
                    '2001:0DB8:0:DBA8:0:0:0:1');
INSERT 0 1
essais=> SELECT * FROM Machines WHERE name = 'galadriel';
   name    |      address
-----------+--------------------
 galadriel | 2001:db8:0:dba8::1
```

L'adresse sera affichée sous une forme différente de celle sous laquelle elle a été entrée. La section 3.3.1 du RFC expliquait pourtant que c'était une mauvaise idée que d'afficher sous une forme différente, petite contradiction de ce RFC.

Donc, concrètement, comment doit être affichée une adresse ?

— Les zéros initiaux dans un champ doivent être supprimés (section 4.1). `2001:0db8::0001` doit être écrit `2001:db8::1`.
— L'indication d'une suite de champs nuls, `::` doit être utilisée au maximum, doit s'appliquer à la suite la plus longue (s'il y en a plusieurs) et, en cas d'égalité, à la première (section 4.2). Ainsi, `2001:db8:0:0:0:0:0:1` doit s'écrire `2001:db8::1`, `2001:db8:0:42:0:0:0:1` doit être mis sous la forme `2001:db8:0:42::1` et `2001:db8:0:0:137:0:0:1` doit être affiché `2001:db8::137:0:0:1`.
— Les chiffres hexadécimaux doivent être en minuscule (section 4.3), donc `2001:DB8::BAD:DCAF` doit être `2001:db8::bad:dcaf`.

Le RFC prévoit aussi le cas des adresses spéciales comme les adresses IPv4 représentées en IPv6 (section 5).

Si l'adresse IP indiquée comprend également un port, il y avait traditionnellement plusieurs formes. La section 6 rend obligatoire la syntaxe avec crochets `[2001:db8::deb:1]:80`, issue du RFC 3986 (section 3.2.2) et qui n'était obligatoire que pour les URL.

L'annexe A donne des conseils pour les programmeurs, qui vont devoir écrire des programmes affichant des formes correctes. Ainsi, sur FreeBSD 7.0, l'utilisation de `getnameinfo()` avec l'option `NI_NUMERICHOST` produit déjà le résultat correct, sauf pour les adresses dites spéciales.

De même, PostgreSQL produit déjà des adresses au bon format. Et avec `inet_pton()` ? Le programme (en ligne sur `https://www.bortzmeyer.org/files/canonicalize-v6.c`) montre que son comportement est bien celui du RFC :

```
% ./canonicalize-v6 toto 2001:db8:Bad:0:0::0:1 127.0.0.1 35:0FF::1 2001:0:0:1:b:0:0:A 2001:db8:0:0:1:0:0:0
toto -> Illegal input IPv6 address
2001:db8:Bad:0:0::0:1 -> 2001:db8:bad::1
127.0.0.1 -> Illegal input IPv6 address
35:0FF::1 -> 35:ff::1
2001:0:0:1:b:0:0:A -> 2001::1:b:0:0:a
2001:db8:0:0:1:0:0:0 -> 2001:db8:0:0:1::
```

Voir aussi le RFC 4038 pour des détails sur les questions IPv6 pour les applications.