RFC 4122 : A Universally Unique IDentifier (UUID) URN Namespace

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 avril 2008

Date de publication du RFC : Juillet 2005

https://www.bortzmeyer.org/4122.html

Ce RFC décrit un espace de noms URN pour les UUID, une famille d'identificateurs uniques, obtenus sans registre central.

Les UUID, également connus sous le nom de GUID, sont issus à l'origine du système des Apollo, adopté ensuite dans la plate-forme DCE. Ils ont une taille fixe, 128 bits, et sont obtenus localement, à partir d'un autre identificateur unique comme l'adresse MAC ou bien en tirant au hasard, la grande taille de leur espace de nommage faisant que les collisions sont très improbables (section 2 du RFC). Ce RFC reprend la spécification de DCE de l'Open Group (ex-OSF) et ajoute la définition d'un espace de noms pour des URN (section 3). (Il existe aussi une norme ITU sur les UUID et un registre des UUID <http://www.itu.int/ITU-T/asn1/uuid.html>, pour ceux qui y tiennent.)

Les UUID peuvent donc convenir pour identifier une entité sur le réseau, par exemple une machine mais aussi, vu leur nombre, comme identificateur unique pour des transactions (ce qui était un de leurs usages dans DCE). En revanche, ils ne sont pas résolvables.

Sur Unix, on peut fabriquer un UUID avec la commande `uuidgen`, qui affiche la représentation texte standard que normalise notre RFC :

```
% uuidgen
317e8ed3-1428-4ef1-9dce-505ffbcba11a
% uuidgen
ec8638fd-c93d-4c6f-9826-f3c71436443a
```

Sur une machine FreeBSD, un UUID de la machine est automatiquement généré (par le script `/etc/rc.d/hostid` et stocké dans le fichier `/etc/hostid`.

On trouve les UUID à de nombreux endroits en informatique, par exemple dans les logiciels Mozilla (cf. <http://developer.mozilla.org/en/docs/Generating_GUIDs>).

Pour l'affichage sous forme d'URN (RFC 8141 [1]), on ajoute juste l'espace `uuid` par exemple `urn:uuid:ec8638fd-`

La section 4 du RFC détaille le format interne de l'UUID (en dépit des apparences, l'UUID n'est pas plat, il a une structure). Notamment, la section 4.1.3 précise le champ Version (qui devrait plutôt s'appeler Type), puisqu'un UUID peut être généré à partir d'une estampille temporelle (version 1, option `-t` de `uuidgen`), d'une valeur aléatoire (version 4, option `-r`) ou d'un résumé cryptographique (version 3).

L'UUID utilise également l'adresse MAC de la machine, si elle est présente (section 4.1.6).

La section 4.2, elle, décrit le processus de génération d'un UUID à base temporelle. Idéalement, il faut utiliser une graine enregistrée sur le disque (pour éviter de générer des UID identiques) ainsi que l'instant de la génération. Mais lire sur le disque prend du temps (alors qu'on peut vouloir générer des UUID rapidement, par exemple pour identifier des transactions) et l'horloge de la machine n'a pas toujours une résolution suffisante pour éviter de lire deux fois de suite le même instant. Cette section contient donc également des avis sur la génération fiable d'UUID, par exemple (section 4.2.1.2) en gardant en mêmoire le nombre d'UUID générés, pour les ajouter à l'heure. Si on préfère des UUID créés par résumé cryptographique, la section 4.3 est là pour cela.

La section 6, consacrée à la sécurité, rappelle qu'un UUID ne doit pas être utilisé comme capacité (car il est trop facile à deviner) et qu'il ne faut pas demander à un humain de comparer deux UUID (ils se ressemblent trop pour un œil humain).

L'annexe A est une excellent mise en œuvre, en langage C, des principes décrits dans la section 4. Si on préfère utiliser Python, il existe, depuis la version 2.5, un module UUID <http://docs.python.org/lib/module-uuid.html> qui offre des fonctions de génération d'UUID de différentes versions :

```
import uuid

myuuid = uuid.uuid1() # Version 1, Time-based UUID
otheruuid = uuid.uuid4() # Version 4, Random-based UUID
yetanotheruuid = uuid.uuid5(uuid.NAMESPACE_DNS,
                            "www.example.org")
                       # Version 5, a name hashed by SHA1
if (myuuid == otheruuid or \
    myuuid == yetanotheruuid or \
    otheruuid == yetanotheruuid):
    print "They are equal, PANIC!"

print myuuid
print otheruuid
print yetanotheruuid
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc8141.txt`

À noter que le SGBD PostgreSQL, à partir de la version 8.3, inclus un type UUID <http://www.postgresql.org/docs/current/interactive/datatype-uuid.html>.

```
essais=> CREATE TABLE Transactions (id uuid, value INT);
CREATE TABLE
essais=> INSERT INTO Transactions VALUES
           ('74738ff5-5367-5958-9aee-98fffdcd1876', 42);
INSERT 0 1
essais=> INSERT INTO Transactions VALUES
            ('88e6441b-5f5c-436b-8066-80dca8222abf', 6);
INSERT 0 1
essais=> INSERT INTO Transactions VALUES ('Pas correct', 3);
ERROR:  invalid input syntax for uuid: "Pas correct"
essais=> SELECT * FROM Transactions;
                  id                  | value
--------------------------------------+-------
 74738ff5-5367-5958-9aee-98fffdcd1876 |    42
 88e6441b-5f5c-436b-8066-80dca8222abf |     6
(2 rows)
```

Il existe plusieurs exemples d'utilisation des UUID. Par exemple, Linux s'en sert pour identifier les disques attachés à la machine, de préférence à l'ancien système où l'ajout d'un nouveau disque pouvait changer l'ordre des numéros sur le bus et empêcher le système de trouver un disque. Un `:etc/fstab` typique sur Linux contient donc désormais des :

```
UUID=da8285a0-3a70-413d-baed-a1f48d7bf7b2       /home   ext3 defaults ...
```

plutôt que les anciens :

```
/dev/sda3       /home           ext3    defaults
```

car `sda3` n'est pas un identificateur stable. L'UUID, lui, est dans le système de fichiers et ne changera pas avec les changements sur le bus. On peut l'afficher avec `dumpefs` :

```
# dumpe2fs -h /dev/sda3
...
Filesystem UUID:          da8285a0-3a70-413d-baed-a1f48d7bf7b2
...
```