RFC 6350 : vCard Format Specification

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 septembre 2011

Date de publication du RFC : Août 2011

https://www.bortzmeyer.org/6350.html

---

vCard, désormais en version 4 avec ce RFC est le format standard de carnet d'adresses sur l'Internet. Tous les logiciels de gestion de carnet d'adresses peuvent (ou devraient pouvoir) lire et écrire du vCard. Ce RFC met à jour la norme vCard.

vCard est simplement un modèle de données (très léger) et une syntaxe pour représenter des données permettant de contacter des individus ou organisations. Par exemple, une entrée vCard pour mon employeur, l'AFNIC, pourrait être :

```
BEGIN:VCARD
VERSION:4.0
FN:Association Française pour le Nommage Internet en Coopération
KIND:org
GENDER:N
NICKNAME:AFNIC
LANG;PREF=1:fr
LANG;PREF=2:en
ADR:;;Immeuble International;Saint-Quentin-en-Yvelines;;78181;France
LOGO:http://www.afnic.fr/images/logo.png
TEL;VALUE=uri:tel:+33-1-39-30-83-00
EMAIL:afnic@afnic.fr
GEO:geo:2.0455,48.7873
BDAY:19980101
TZ:Paris/Europe
URL:http://www.afnic.fr/
NOTE;LANGUAGE=en:@AFNIC on Twitter\,
 AFNIC on Facebook.
END:VCARD
```

Ce texte peut ensuite être lu et interprété par les logiciels de gestion de carnet, être envoyé via l'Internet, etc. Un exemple courant est l'envoi d'une vCard dans le courrier, via MIME et le type `text/vcard` (section 10.1 pour l'enregistrement de ce type). Le cas échéant, il est relativement compréhensible par un humain (s'il comprend l'anglais et quelques sigles), ce qui justifie le type MIME `text` indiqué plus haut.

La version de vCard dans ce RFC est largement terminée depuis mi-2010. Mais de nombreux détails avaient été discutés jusqu'au dernier moment. Si la syntaxe ne pose guère de problèmes (on notera qu'il existe une variante XML normalisée, dans le RFC 6351 [1], et une autre en JSON, dans le RFC 7095), le modèle de données peut susciter des discussions sans fin.

Pour prendre des exemples de cette discussion, voici quelques sujets qui avaient été abordés lors de la réunion du groupe de travail à l'IETF de Pékin en 2010 :

— La nouvelle propriété `RELATED` qui peut valoir `spouse`, `friend`, etc. Une jolie liste des valeurs possibles est <`http://gmpg.org/xfn/11`>. On pourra coder toutes les relations Facebook en vCard.
— La longue discussion passionnée à propos du remplacement de l'ancien attribut `SEX` de vCard 3 par `GENDER`. La moitié des participants se sont exprimés, ce qui est rare à l'IETF. Le sexe est déterminé biologiquement et, en toute première approximation, peut être considéré comme binaire <`https://www.bortzmeyer.org/iso-5218.html`> (homme ou femme). Le genre est construit socialement et peut prendre bien plus de valeurs. `GENDER` pourra donc être du texte libre (une des propositions discutées à Pékin avait été de créer un registre IANA des différents genres possibles...) et les exemples du RFC (avec des valeurs comme "*grrrl*", "*intersex*" ou "*complicated*") illustrent la difficulté du problème. Le changement dans vCard permet donc de mieux prendre en compte des cas comme celui des transsexuels ou, tout simplement, des gens qui ont une autre identité que celle que leur impose la biologie. L'ironie de la longue et animée discussion est qu'il n'y avait que des hommes dans la salle. (`GENDER` est désormais décrit en détail dans la section 6.2.7 du RFC, avec notamment sa façon subtile de combiner deux champs, un énuméré et un en texte libre.)
— Plus triste, une discussion avait eu lieu sur la propriété `DEATH` permettant, si nécessaire, d'indiquer la date de la mort de l'entitée considérée. Elle a finalement été retirée. Mettre du vCard sur une pierre tombale aurait pourtant été curieux... Cela a finalement été possible quelque temps après, avec l'approbation du RFC 6474.
— Et les adresses de courrier électronique en Unicode du RFC 6532 ont évidemment suscité bien de la perplexité. Finalement, la propriété `EMAIL` permettra de telles adresses (voir la section 6.4.2 pour les détails).

J'espère que cette liste amusante vous aura donné une idée des problèmes que pose la définition d'un modèle de données standard. Retournons maintenant à la technique. D'abord, le niveau lexical (sections 3.1 et 3.2) : les fichiers vCard sont en UTF-8 (la possibilité d'autres encodages a été retirée, avec cette version de vCard), et sont composés de lignes terminées par CRLF (c'est-à-dire les deux caractères U+000D et U+000A). Comme pour les en-têtes du courrier électronique, on peut continuer sur plusieurs lignes : si une ligne commence par un espace ou une tabulation, c'est une continuation (regardez la `NOTE` dans l'exemple plus haut). Du fait de ces continuations, un outil mono-ligne comme grep n'est pas bien adapté pour chercher dans des vCards.

Ensuite, la syntaxe (section 3.3). Elle est très simple et indiquée en ABNF (RFC 5234). Une « carte » commence par la ligne `BEGIN:VCARD`, continue par une ligne avec le numéro de version, puis par une ou plusieurs lignes comportant les données, et se termine avec `END:VCARD`. Les données sont composées de **propriétés**, chaque ligne étant un nom de propriété (insensible à la casse, même si la convention

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc6351.txt`

recommandée est de les mettre en majuscules), suivi d'un :, puis de la valeur de la propriété (celle-ci pouvant elle-même être structurée, comme dans l'`ADR` ci-dessus).

Certains caractères sont spéciaux, comme la virgule et, si on veut les avoir comme valeur, doivent subir un échappement avec la barre inverse (section 3.4).

Les données peuvent avoir des **paramètres**. La section 5 décrit ce concept. Un paramètre permet de préciser des métadonnées sur une propriété. Par exemple, `LANGUAGE` permet d'indiquer la langue d'un texte. `PREF` permet de spécifier, lorsqu'il existe plusieurs exemplaires d'une propriété, laquelle est préférée (celle qui a la valeur la plus basse). Ainsi, dans l'exemple de l'AFNIC ci-dessus, c'est le français qui est la langue de contact préférée. `SORT-AS` permet de définir un tri entre les valeurs (un problème difficile pour les noms d'humains). `TYPE` permet d'indiquer, pour une `ADR`, si elle concerne la maison ou le travail. (vCard version 3 avait un mécanisme plus riche, qui aurait permis, dans l'exemple de l'AFNIC ci-dessus, d'indiquer la différence entre l'adresse postale - `TYPE=postal` et l'adresse physique - `TYPE=parcel`. Trop confus, il a été supprimé.) Et il existe plusieurs autres paramètres.

La sémantique est décrite dans la suite de cette section 3.3 et, pour chaque propriété, dans les sections 4, 5 et 6. Notons qu'une seule propriété est obligatoire (à part `VERSION`), `FN`, le nom de l'entité décrite dans la vCard. Pour chaque propriété, la section 4 indiquera si des valeurs multiples sont autorisées. Par exemple, on peut avoir plusieurs `NICKNAME` (surnom), mais un seul (facultatif) `GENDER` (genre).

Je ne vais évidemment pas lister **toutes** les propriétés que décrit la section 5 (voir le registre IANA `<https://www.iana.org/assignments/vcard-elements/vcard-elements.xml>` pour une liste à jour). Pour chacune, elle donne son nom, sa cardinalité (zéro, zéro-ou-une, une, zéro-ou-plus, une-ou-plus occurrence) et son type. Pour ce dernier, il y a de nombreuses possibilités comme, par exemple, du texte libre, un URI, une date ou un temps (en utilisant la norme ISO 8601 et pas le plus simple RFC 3339), un booléen, un nombre entier, une langue (représentée selon le RFC 5646), etc.

À certains égards, la section 6 est la plus importante du RFC. C'est la liste de toutes les propriétés possibles actuellement, avec leurs caractéristiques. Quelques exemples, non limitatifs :

— `N` indique le nom, sous forme d'une série de composants (dont la plupart sont facultatifs). Pour les humains, je rappelle surtout que le fait d'avoir Prénom + Nom est très loin d'être universel `<http://www.kalzumeus.com/2010/06/17/falsehoods-programmers-believe-about-names/>`.
— `KIND` indique quelle est la catégorie de l'entité décrite par cette « carte » (personne physique, organisation - comme dans l'exemple de l'AFNIC plus haut, etc).
— `PHOTO` est l'URI (éventuellement de plan `data:`, c'est-à-dire fournissant directement le contenu) d'une photo de la personne. `LOGO` joue un rôle analogue pour les organisations.
— `BDAY` est la date de naissance de l'entité.
— `IMPP` indique le moyen de contacter l'entité par messagerie instantanée.
— `LANG` permet d'indiquer, sous la forme d'une étiquette de langue comme `fr` ou `pt-BR`, la langue à utiliser pour contacter l'entité.
— `GEO` est la localisation physique habituelle de l'entité, en suivant de préférence la syntaxe du RFC 5870. (Notez le « de préférence ». Analyser la syntaxe d'une vCard est très simple, mais accéder à la sémantique est bien plus complexe, en raison du choix qui est laissé pour le codage d'informations comme le numéro de téléphone ou la position physique.)
— `KEY` permet de distribuer des clés cryptographiques (attention à comment on a récupéré le vCard avant de leur faire confiance...)

Notez que la liste n'est pas figée dans le temps (c'est une nouveauté de vCard version 4). Des propriétés ou paramètres nouveaux pourront être enregistrés, selon la procédure décrite dans la section 10.2. Il faudra d'abord les discuter sur la liste `vcarddav@ietf.org`, puis il y aura un examen par un expert. Si accepté, le nouvel élément sera enregistré à l'IANA <https://www.iana.org/assignments/vcard-elements/vcard-elements.xml>.

Le reste du RFC est consacré à des points d'utilisation des vCard. Par exemple, la section 7 est consacrée à la synchronisation des vCards entre deux engins (par exemple l'ordinateur du bureau et le "*smartphone*"). Que faire si une entrée a été modifiée sur un carnet et détruite dans un autre ? Deux mécanismes sont fournis pour aider à une fusion intelligente des deux carnets, une propriété `UID` qui identifie de manière unique et non ambigüe une entité (de préférence en utilisant le RFC 4122) et `PID` qui identifie une propriété. Les section 7.1.1 et 7.1.2 détaillent l'algorithme à utiliser lorsqu'on synchronise deux carnets. Par exemple, si on a identifié, grâce à `UID`, une entrée dans les deux carnets, et qu'une propriété est différente entre les carnets, alors, si la cardinalité d'une propriété est de 1, ou bien si le PID est le même, les deux propriétés doivent être fusionnées. Sinon, le synchroniseur est libre de fusionner, ou tout simplement de mettre les deux propriétés côte-à-côte. La section 7.2 donne des exemples détaillés de fusions. À noter qu'il existe un protocole pour la synchronisation des cartes avec un serveur, CardDAV, normalisé dans le RFC 6352.

Le message d'enregistrement du type MIME `text/vcard`, qui commençait la procédure, avait été fait en août 2010 <http://www.ietf.org/mail-archive/web/ietf-types/current/msg00954.html>.

Publier ainsi de l'information, parfois sensible, sur l'Internet, n'est pas sans conséquences. La section 9 couvre les risques de sécurité de vCard. Par exemple, la carte en elle-même n'offre aucune garantie d'authentification ou d'intégrité. Rien ne prouve qu'une carte indiquant `FN:Theo de Raadt` vient réellement de de Raadt. Et même si la carte était authentique au départ, rien n'indique qu'elle n'a pas été modifiée en cours de route, sauf si elle a été transportée de manière fiable (par exemple dans un message signé avec PGP). Bref, ne vous fiez pas aveuglément à n'importe quelle vCard trouvée sur l'Internet !

Autre risque, celui pour la vie privée. Ce n'est pas par hasard ou par oubli que vous ne trouverez pas `BDAY` ou `ADR` sur ma carte, un peu plus loin. Une carte doit donc ne comporter que des informations publiques, ou bien être uniquement transportée de manière sûre (par exemple via un canal chiffré) et vers des gens de confiance.

Les changements de vCard depuis la version 3 (qui était normalisée dans les RFC 2425 et RFC 2426) sont décrits dans l'annexe A. La liste est longue et pas facile à résumer (ce n'est qu'une accumulation de changements ponctuels). Le changement que je trouve le plus spectaculaire est la possibilité d'enregistrer de nouveaux éléments sans changer de RFC, juste par un processus d'enregistrement IANA. Sinon, voir la section 6.7.9 pour le rôle de la propriété `VERSION` pour gérer le fait que les cartes en circulation aient des versions différentes.

Si vous voulez en savoir plus sur vCard, le site Web du groupe de travail <http://www.vcarddav.org/> est très riche, quoique plutôt difficile d'accès (il est conçu pour le travail du groupe, pas pour la pédagogie).

Et du point de vue pratique, quels outils le programmeur a à sa disposition s'il veut lire et/ou écrire du vCard ? Il existe une bibliothèque pour les programmeurs C (le paquetage se nomme libvc-dev sur Debian), qui est utilisée dans des applications comme rolo <http://rolo.sourceforge.net/>. Elle n'est distribuée avec aucune documentation digne de ce nom et pas d'exemples. Pour les programmeurs Perl, il existe `Text::vCard`. Attention à son analyseur : très laxiste, il accepte des vCards bourrées d'erreurs de syntaxe.

Voici un exemple d'usage de cette bibliothèque Perl :

```
#!/usr/bin/perl

use Text::vCard::Addressbook;

for $file (@ARGV) {
    my $address_book = Text::vCard::Addressbook->new({
'source_file' => $file});
    $number = 0;
    foreach my $vcard ($address_book->vcards()) {
print "Got card for " . $vcard->fullname() . "\n";
$number++;
    }
    if (!$number) {
print "No correct vCard in $file\n";
    }
}
```

Ah, et si vous voulez me contacter, voici une carte pour moi :

```
BEGIN:VCARD
VERSION:4.0
FN:Stéphane Bortzmeyer
N:Bortzmeyer;Stéphane;;;
UID:urn:uuid:a06340f8-9aca-41b8-bf7a-094cbb33d57e
GENDER:M
KIND:individual
EMAIL:stephane+blog@bortzmeyer.org
TITLE:Indigène de l'Internet
PHOTO:http://www.bortzmeyer.org/images/moi.jpg
LANG;PREF=1:fr
LANG;PREF=2:en
IMPP;PREF=1:xmpp:bortzmeyer@dns-oarc.net
IMPP;PREF=2:xmpp:bortzmeyer@gmail.com
URL:http://www.bortzmeyer.org/
KEY:http://www.bortzmeyer.org/files/pgp-key.asc
END:VCARD
```

Merci à Simon Perreault pour sa relecture.