RFC 7808 : Time Zone Data Distribution Service

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 avril 2016

Date de publication du RFC : Mars 2016

https://www.bortzmeyer.org/7808.html

Ah, les fuseaux horaires... Quelle source infinie de complications pour le programmeur, pour l'administrateur système de machines situées partout dans le monde, ou tout simplement pour le voyageur ou l'organisateur de télé-réunions internationales... Comme tout serait plus simple si, comme dans le roman d'Asimov « Les cavernes d'acier », les hommes vivaient en permanence sous terre, utilisant un temps uniforme. Mais, en attendant ce futur béni, il faut bien gérer les fuseaux horaires ("*time zones*"). Il existe une base de données officielle <https://www.iana.org/time-zones> de ces fuseaux, il ne reste plus qu'à la distribuer efficacement, pour que toute machine connectée à l'Internet ait la même information. C'est le but du protocole normalisé dans ce RFC.

Que contient une base de données des fuseaux horaires (section 1 du RFC) ? Typiquement une liste de ces fuseaux avec, pour chacun d'entre eux, son décalage par rapport à UTC et ses règles concernant l'heure d'été. Les fuseaux ne suivent pas strictement le méridien, ils collent en général à des frontières inter ou intra-étatiques. Les fuseaux eux-même, et leurs décalages avec UTC, sont très stables. Mais les règles concernant l'heure d'été changent, elles, souvent. Il y a aussi le problème des secondes intercalaires, qui sont ajoutées (ou, en théorie, retirées) par l'IERS de manière imprévisible (car la rotation de la Terre n'est pas prévisible). La base actuelle représente un travail d'érudition formidable <http://blog.jonudell.net/2009/10/23/a-literary-appreciation-of-the-olsonzoneinfotz-database/>.

Au passage, un point de terminologie. Le RFC parle de « "*time zone*" » là où on dit en général en français « fuseau horaire ». Le terme français n'est pas idéal car il fait penser à un fuseau allant d'un pôle à l'autre (et englobant donc des pays différents, ayant des heures d'été différentes) alors qu'une "*time zone*" est plus restreinte. Il faut donc bien se rappeler qu'on parle ici de zones limitées à un pays, et ayant des règles (décalage avec UTC, heure d'été) uniformes. Il faut donc peut-être mieux dire « zone » ou « zone horaire » en français.

Cette information est essentielle pour les protocoles et les formats qui gèrent le temps, comme iCalendar (RFC 5545 [1]), ainsi que pour les systèmes d'exploitation. Il faut donc maintenir à jour la base de données et, jusqu'à ce RFC, il n'existait pas de mécanisme standard pour cela. Par exemple, pour un système d'exploitation comme Debian, la mise à jour se fait typiquement via le mainteneur d'un paquetage « base des fuseaux horaires » (chez Debian, le paquetage se nomme `tzdata`), qui introduit les changements dans son paquetage, paquetage qui est ensuite installé sur chaque machine lorsque l'administrateur système décide des mises à jour. Si le système est vieux et plus maintenu, ce sera à l'administratreur système local de « patcher » ses fichiers. Parfois, des paquetages viennent avec leur propre base des fuseaux horaires, distincte de celle du système, pour compliquer un peu les choses.

Au passage, signalons que ce RFC normalise le protocole mais qu'il faut également disposer d'une source de données de référence. Il en existe une à l'IANA, décrite dans le RFC 6557.

La section 2 décrit à grands traits l'architecture du système. Tout en amont, des contributeurs envoient des données sur les fuseaux horaires, elles sont agrégées par le fournisseur de référence ("*root provider*", par exemple l'IANA citée plus haut). Ce fournisseur l'envoie ensuite aux clients (par exemple par le protocole de ce RFC). Le schéma du RFC est un peu plus compliqué, supposant que divers intermédiaires pourraient apparaitre. Notez que le protocole décrit ici est entre fournisseur et clients. En amont (entre contributeurs et fournisseur), il n'y a pas de mécanisme spécifié.

La section 3 de notre RFC définit la terminologie. Une zone ("*time zone*") se caractérise donc par un ensemble de règles uniformes. La zone a un nom (utilisé par exemple dans la propriété `TZID` du RFC 5545) mais il n'existe pas de norme pour cela. Chaque fournisseur pourrait utiliser son schéma de nommage. Ainsi, la base IANA `<https://www.iana.org/time-zones>` utilise un nom de région, une barre oblique et un nom de pays ou de ville comme `Europe/Paris` ou `Indian/Cocos`. En tout cas, il faut se méfier des abréviations ambigues, n'ayant souvent qu'une signification locale, comme PST (qui peut désigner trois zones différentes).

L'information sur les zones change (autrement, on n'aurait pas besoin d'un protocole pour télécharger les mises à jour) et il est donc important d'avoir un numéro de version permettant de savoir si on a bien la dernière version. Il peut être global à la base, ou bien spécifique à chaque zone.

Maintenant, le protocole lui-même (section 4 du RFC). Il repose, comme souvent de nos jours, sur le couple « HTTP (RFC 7230) et JSON (RFC 8259) », ce dernier servant à représenter les méta-données sur les zones. Les ressources auxquelles on accède en HTTP utilisent les gabarits d'URI du RFC 6570. Et les données elle-mêmes, les informations sur une zone horaire, en quoi sont-elles codées ? Le format par défaut est celui d'iCalendar (RFC 5545). Mais on peut aussi utiliser la représentation en XML du RFC 6321 ou celle en JSON du RFC 7265. La classique négociation de contenu de HTTP (RFC 7231, section 5.3.2) sert au client à choisir son format parmi ceux que le serveur veut bien fournir.

La base peut être assez grosse et les clients vouloir la toute dernière version. La charge sur le réseau des fournisseurs risquerait alors de devenir insupportable. C'est pourquoi il est possible de faire des requêtes conditionnelles, de façon à ne récupérer des données que si elles sont nouvelles. Là encore, c'est du classique HTTP avec les "*Etags*" (RFC 7232). Un client peut alors interroger le serveur du fournisseur une fois par jour (valeur recommandée par le RFC) sans risque de saturer le réseau.

Si le client reçoit uniquement le décalage avec UTC et les règles pour l'heure d'été, calculer une heure dans le futur peut être délicat car ces règles sont complexes. Le protocole prévoit donc la possibilité pour le client de demander au serveur de faire ces calculs et de lui envoyer les résultats.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5545.txt`

Et si ça ne marche pas, que le serveur, pour une raison ou pour une autre, ne peut pas envoyer ce qui lui est demandé ? Il doit alors utiliser les erreurs structurées du RFC 7807 pour signaler ce qui ne va pas. (Les types d'erreurs possibles figurent dans un registre IANA <https://www.iana.org/assignments/tzdist-identifiers/tzdist-identifiers.xml#identifiers>.)

Comment est-ce qu'un client de ce protocole trouve le serveur ? Il doit de toute façon connaître le nom de domaine de son fournisseur, supposons ici `example.net`. Ensuite, il y a au moins deux méthodes :

— Utiliser les enregistrements SRV (RFC 2782), en demandant `_timezone._tcp.example.net`. Il obtiendra ainsi le nom du serveur et le port à utiliser. Un enregistrement supplémentaire, sous le même nom que l'enregistrement SRV, de type TXT, indique un éventuel chemin à ajouter dans l'URI.
— Ou bien utiliser les URI `.well-known` du RFC 8615, en faisant une requête HTTP pour la ressource `timezone`.

La section 5 du RFC décrit ensuite les actions que le serveur peut effectuer à la demande. Elles sont spécifiées en utilisant les gabarits du RFC 6570. Donc, par exemple, un gabarit `{/service-prefix}/capabilities` indique que le client doit ajouter `/capabilities` au préfixe de chemin découvert comme indiqué précédemment. Prenons un exemple complet : le client sait que son fournisseur de zones horaires est `example.net`. Il fait deux requêtes DNS et obtient deux enregistrements :

```
_timezone._tcp.example.net. SRV 0 1 8081 tz.example.com.
_timezone._tcp.example.net. TXT "path=/timezones"
```

S'il cherche à déterminer les capacités du serveur, il va alors faire une requête à l'URL `http://tz.example.com:8081`

C'est quoi, ces "*capabilities*" que je viens d'utiliser dans l'exemple ? C'est la première des actions possibles sur un serveur. Comme son nom l'indique, elle renverra un objet JSON contenant un membre `actions` qui sera la liste des actions possibles sur ce serveur (avec, pour chacune, le gabarit d'URI à utiliser). Une liste des actions standards possibles figure dans un registre IANA <https://www.iana.org/assignments/tzdist-actions/tzdist-actions.xml#actions>.

L'action `list`, elle, donne la liste des zones horaires connues du serveur. Et l'action `get` renvoie les données pour une zone précise. Son gabarit est `{/service-prefix,data-prefix}/zones{/tzid}{?start,end}` (`tzid` étant l'identificateur de la zone, par exemple `Pacific/Noumea`) et un exemple d'utilisation est :

```
[Requête]

GET /timezones/zones/America%2FNew_York HTTP/1.1
Host: tz.example.com:8081
Accept:text/calendar

[Réponse]

HTTP/1.1 200 OK
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: text/calendar; charset="utf-8"
Content-Length: xxxx
ETag: "123456789-000-111"

BEGIN:VCALENDAR
...
BEGIN:VTIMEZONE
TZID:America/New_York
...
END:VTIMEZONE
END:VCALENDAR
```

Les codes d'erreur habituels de HTTP sont utilisés donc, par exemple, si on demande un `tzid` inconnu, on récupérera un beau 404, mais avec le corps de la réponse en JSON, suivant le RFC 7807 :

```
[Requête]

GET /timezones/zones/Atlantid%2FPlutopolis HTTP/1.1
Host: tz.example.com:8081
Accept:text/calendar

[Réponse]

HTTP/1.1 404 Not Found
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: application/problem+json; charset="utf-8"
Content-Language: en
Content-Length: xxxx

{
     "type": "urn:ietf:params:tzdist:error:tzid-not-found",
     "title": "Time zone identifier was not found on this server",
     "status": 404
}
```

Il existe plusieurs autres actions, comme `expand` qui dit au serveur de faire les calculs d'heure d'été lui-même, ou `find`, qui permet de chercher une zone par une partie de son nom.

Notez bien qu'il n'y a pas d'URI fixe et pré-déterminé pour les actions : il faut utiliser les gabarits pour les générer.

Les détails des objets JSON qui peuvent être renvoyés en réponse à ces actions figurent en section 6 de notre RFC.

Et la sécurité ? Elle est cruciale car, si on peut changer la connaissance qu'une machine a de l'heure, plein d'attaques deviennent possibles (fausser les estampilles temporelles dans les journaux, activer ou désactiver un certificat, etc). Il faut donc prendre soin d'utiliser un fournisseur fiable, et de récupérer la base de manière sécurisée. (HTTPS, forcément, et avec vérification sérieuse de l'identité du serveur, en suivant le RFC 6125 ou bien le RFC 6698).

Il n'y a apparemment pour le moment qu'une mise en œuvre, dans le système de calendrier Bedework <`https://www.apereo.org/projects/bedework`>. Je ne connais pas encore de service disponible qui serve la base de données suivant ce protocole (c'est pour cela que je ne montre pas d'exemple réel). Notamment, l'IANA ne le fait pas (ce n'était pas demandé dans le RFC). Il existe des services qui distribue la base, mais avec un autre protocole, comme <`https://timezonedb.com/api`>.