Récupérer la date d'expiration d'un domaine en RDAP

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 4 juillet 2021. Dernière mise à jour le 7 octobre 2022

https://www.bortzmeyer.org/expiration-rdap.html

C'est dimanche, il ne fait pas beau, pas question de pique-nique, on va donc faire un peu de JSON. Un des problèmes les plus courants avec les noms de domaine est l'expiration accidentelle du nom. « Nous avons oublié de le renouveler » et paf plus rien ne marche. Pourtant, la date à laquelle le domaine expire est publiquement annoncée, via des protocoles comme whois ou RDAP. Comment utiliser RDAP pour récupérer cette date dans un programme, par exemple un programme d'alerte qui préviendra si l'expiration approche ?

Un peu de contexte d'abord ; un certain nombre (mais pas tous) de registres de noms de domaine exigent un renouvellement périodique du nom <https://www.afnic.fr/noms-de-domaine/tout-savoir/gerer-son-nom-de-domaine/>, avec paiement et parfois acceptation de nouvelles conditions. L'oubli de ce renouvellement est un gag très fréquent. On ne paie pas, et, à la date d'expiration, le domaine disparait ou bien est mis en attente, plus publié dans le DNS, ce qui fait que tous les services associés cessent de fonctionner. Dans le premier cas, le pire, le domaine a été supprimé, et un autre peut l'enregistrer rapidement. Il est donc crucial de ne pas laisser le domaine expirer. Un des outils indispensables quand on gère un domaine important est donc une supervision automatique de l'approche de l'expiration. L'information est diffusée par le registre, via plusieurs protocoles. Le traditionnel whois, normalisé dans le RFC 3912 [1] a plusieurs inconvénients notamment le fait que le format de sortie n'est pas normalisé. Il est donc difficile d'écrire un programme qui analyse l'information retournée (il existe quand même des solutions comme, en Perl, Net : :DRI <https://metacpan.org/pod/Net::DRI>). Au contraire, le plus moderne RDAP a un format de sortie normalisé (dans le RFC 9083). Utilisons-le, en prenant comme exemple le domaine de ce blog, bortzmeyer.org.

RDAP utilise HTTPS et produit du JSON (RFC 8259). D'autre part, pour interroger le bon serveur RDAP (celui du registre), il faut connaitre son nom, ce qui se trouve dans un registre IANA <https://data.iana.org/rdap/dns.json> qui associe le TLD à son serveur RDAP. On apprend ainsi que les informations concernant .org sont à demander à https://rdap.publicinterestregistry.org/rdap/. Utilisant le RFC 9082 pour former une requête RDAP correcte, on va utiliser curl pour poser la question sur bortzmeyer.org :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3912.txt

```
% curl -s https://rdap.publicinterestregistry.org/rdap/domain/bortzmeyer.org
```

On récupère un gros JSON compliqué, que je ne vous montre pas ici. On veut juste la date d'expiration du domaine. On va se servir pour cela de l'excellent outil jq <https://www.bortzmeyer.org/jq.html> :

```
% curl -s https://rdap.publicinterestregistry.org/rdap/domain/bortzmeyer.org | \
       jq '.events | map(select(.eventAction == "expiration"))[0] | .eventDate'
"2021-08-29T09:32:04.304Z"
```

OK, j'ai triché, ça m'a pris plus de quelques secondes pour écrire le programme jq qui extrait cette information, et j'ai dû vérifier dans le RFC 9083. Expliquons donc.

Partons du haut de l'objet JSON retourné. Il contient un membre nommé `events` (RFC 9083, section 4.5) qui indique les évènements passés ou futurs pertinents, comme la création du domaine ou comme sa future expiration. On doit donc commencer par demander au programme jq de filtrer sur ce membre, d'où le `.events` au début. Ensuite, on veut ne garder, parmi les évènements, que l'expiration. On applique donc (`map`) un test (`.eventAction == "expiration"`) à chaque élement du tableau `events` et on ne garde (`select`) que celui qui nous intéresse. Oui, « celui » et pas « ceux » car il n'y a normalement qu'un évènement d'expiration, donc on ne garde que le premier élement (`[0]`) du tableau produit. Ce premier élément, comme tous ceux qui composent le tableau `events` est un évènement (RFC 9083, section 4.5), un objet qui a ici deux membres, `eventAction`, son type (celui sur lequel on filtre, avec `.eventAction == "expiration"` et `eventDate`, l'information qu'on recherche. On termine donc par un filtre du seul `eventDate`, date qui est au format du RFC 3339. Et voilà. On peut ensuite traiter cette date comme on veut. Par exemple, le programme GNU date peut vous traduire cette date en secondes écoulées depuis l'"*epoch*" <https://www.bortzmeyer.org/epoch-50.html> :

```
% date --date=$(curl -s https://rdap.publicinterestregistry.org/rdap/domain/bortzmeyer.org | \
                jq -r '.events | map(select(.eventAction== "expiration"))[0] | .eventDate') \
    +%s
1630229524
```

Abandonnons maintenant curl et jq pour un programme écrit en Python. On utilise la bibliothèque Requests <http://python-requests.org/> pour faire du HTTPS et les bibliothèques standard incluses dans Python pour faire du JSON, du calcul sur les dates, etc. Récupérer le JSON est simple :

```
response = requests.get("%s/%s" % (SERVER, domain))
if response.status_code != 200:
    raise Exception("Invalid RDAP return code: %s" % response.status_code)
```

Le transformer en un dictionnaire Python également :

```
rdap = json.loads(response.content)
```

On va ensuite y chercher la date d'expiration, qu'on transforme en une belle date-et-heure Python sur laquelle il sera facile de faire des calculs :

```
for event in rdap["events"]:
    if event["eventAction"] == "expiration":
        expiration = datetime.datetime.strptime(event["eventDate"], RFC3339)
        now = datetime.datetime.utcnow()
        rest = expiration-now
```

Et on peut ensuite mettre les traitements qu'on veut, ici prévenir si l'expiration approche :

```
if rest < CRITICAL:
            print("CRITICAL: domain %s expires in %s, HURRY UP!!!" % (domain, rest))
            sys.exit(2)
elif rest < WARNING:
            print("WARNING: domain %s expires in %s, renew now" % (domain, rest))
            sys.exit(1)
else:
            print("OK: domain %s expires in %s" % (domain, rest))
            sys.exit(0)
```

Le code complet est disponible (en ligne sur `https://www.bortzmeyer.org/files/expiration-rdap.py`) (il faut aussi ce module (en ligne sur `https://www.bortzmeyer.org/files/ianardap.py`)). Notez qu'il n'est pas tellement robuste, il faudrait prévoir davantage de traitement d'erreur. Python fait toutefois en sorte que la plupart des erreurs soient détectées automatiquement (par exemple s'il n'existe pas de membre `events` dans le résultat) donc, au moins, vous n'aurez pas de résultats erronés. Et puis, par rapport au programme curl+jq plus haut, un grand avantage de ce programme Python est qu'il utilise le registre IANA des serveurs (décrit dans le RFC 9224) et qu'il trouve donc tout seul le serveur RDAP :

```
% ./expiration-rdap.py bortzmeyer.org
OK: domain bortzmeyer.org expires in 55 days, 18:54:01.281920

% ./expiration-rdap.py quimper.bzh
WARNING: domain quimper.bzh expires in 6 days, 18:17:25.919080, renew now
```

Une version plus élaborée de ce programme Python, utilisable depuis des logiciels de supervision automatique comme Nagios, Icinga ou Zabbix est disponible <`https://forge.chapril.org/bortzmeyer/check_expire`>.

Notez qu'un tel outil n'est qu'un élément parmi tout ce qu'il faut faire pour bien gérer l'éventuelle expiration de votre domaine. Il faut aussi mettre les dates de renouvellement dans votre agenda, il est recommandé d'activer l'auto-renouvellement ou, mieux, chez les registres qui le permettent, de louer pour plusieurs années <`https://www.afnic.fr/observatoire-ressources/actualites/le-fr-une-histoire-d`>. Je vous recommande également la lecture du « Guide du Titulaire Afnic <`https://www.afnic.fr/wp-media/uploads/2020/11/Guidepratique_Titulaire_VF.pdf`> » ou des « Bonnes pratiques pour l'acquisition et l'exploitation de noms de domaine de l'ANSSI <`https://www.ssi.gouv.fr/uploads/2014/05/guide_dns_anssi_1.2.pdf`> ».