RFC 9224 : Finding the Authoritative Registration Data Access Protocol (RDAP) Service

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 25 avril 2022

Date de publication du RFC : Mars 2022

https://www.bortzmeyer.org/9224.html

Le protocole RDAP d'accès à l'information sur les objets (noms de domaines, adresses IP, etc) stockés dans un registre fonctionne en interrogeant le serveur en HTTP. Encore faut-il trouver le serveur à interroger. Comment le client RDAP qui veut des informations sur 2001:67c:288::2 sait-il à quel serveur demander ? Ce RFC décrit le mécanisme choisi. En gros, l'IANA gère des « méta-services » qui donnent la liste de serveurs RDAP, mais il peut aussi y avoir redirection d'un serveur RDAP vers un autre. Ce RFC remplace le premier RFC sur le sujet, le RFC 7484 [1], avec très peu de changements.

Le protocole RDAP est décrit entre autres dans le RFC 7480 qui précise comment utiliser HTTP pour transporter les requêtes et réponses RDAP. Comme indiqué plus haut, il ne précise pas comment trouver le serveur RDAP responsable d'un objet donné. Le prédécesseur de RDAP, whois, avait exactement le même problème, résolu par exemple en mettant en dur dans le logiciel client tous les serveurs whois connus. En pratique, beaucoup de gens font simplement une requête Google et accordent une confiance aveugle auu premier résultat venu, quel que soit sa source. Au contraire, ce RFC vise à fournir un mécanisme pour trouver la source faisant autorité.

Avant d'exposer la solution utilisée, la section 1 de notre RFC rappelle comment ces objets (noms de domaine, adresses IP, numéros d'AS, etc) sont alloués et délégués. L'IANA délégue les TLD aux registres de noms de domaines, des grands préfixes IP et les blocs de numéros d'AS aux RIR. Il est donc logique que RDAP suive le même chemin : pour trouver le serveur responsable, on commence par demander à l'IANA, qui a déjà les processus pour cela, et maintient les registres correspondants (adresses IPv4 <https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml>, adresses IPv6 <https://www.iana.org/assignments/ipv6-unicast-address-assignments/

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7484.txt

ipv6-unicast-address-assignments.xml>, numéros d'AS <https://www.iana.org/assignments/as-numbers/as-numbers.xml> et domaines <https://www.iana.org/domains/root/db>).

Pour permettre à RDAP de fonctionner, ce RFC demande donc à l'IANA quelques fichiers supplémentaires, au format JSON (RFC 8259), dont le contenu dérive des registres cités plus haut. Et l'IANA doit aussi prévoir des serveurs adaptés à servir ces fichiers, nommés "*RDAP Bootstrap Service*", à de nombreux clients RDAP.

Le format de ces fichiers JSON est décrit dans la section 3 de notre RFC. Chaque "*bootstrap service*" contient des métadonnées comme un numéro de version et une date de publication, et il contient un membre nommé `services` qui indique les URL où aller chercher l'information (la syntaxe formelle figure en section 10). Voici un extrait d'un des fichiers actuels :

```
% curl https://data.iana.org/rdap/ipv4.json
  "description": "RDAP bootstrap file for IPv4 address allocations",
  "publication": "2019-06-07T19:00:02Z",
  "services": [
    [
      [
        "41.0.0.0/8",
        "102.0.0.0/8",
        "105.0.0.0/8",
        "154.0.0.0/8",
        "196.0.0.0/8",
        "197.0.0.0/8"
      ],
      [
        "https://rdap.afrinic.net/rdap/",
        "http://rdap.afrinic.net/rdap/"
      ]
    ],
    ...
```

On voit que le contenu est un tableau donnant, pour des objets donnés, les URL des serveurs RDAP à contacter pour ces objets indiqués, par exemple pour l'adresse IP `154.3.2.1`, on ira demander au serveur RDAP d'AfriNIC en `https://rdap.afrinic.net/rdap/`.

Le membre `publication` indique la date de parution au format du RFC 3339. Les URL indiqués se terminent par une barre oblique, le client RDAP a juste à leur ajouter une requête formulée selon la syntaxe du RFC 9082. Ainsi, cherchant des informations sur `192.0.2.24`, on ajoutera `ip/192.0.2.24` à l'URL de base.

Pour les adresses IP (section 5), les entrées sont des préfixes, comme dans le premier exemple montré plus haut et la correspondance se fait avec le préfixe le plus spécifique (comme pour le routage IP). Les préfixes IPv4 suivent la syntaxe du RFC 4632 et les IPv6 celle du RFC 5952. Voici un exemple IPv6 (légèrement modifié d'un cas réel) :

```
  [
     [
       "2001:200::/23",
       "2001:4400::/23",
       "2001:8000::/19",
       "2001:a000::/20",
       "2001:b000::/20",
       "2001:c00::/23",
       "2001:e00::/23",
```

```
        "2400::/12"
      ],
      [
        "https://rdap.apnic.net/"
      ]
    ],
    [
      [
        "2001:200:1000::/36"
      ],
      [
        "https://example.net/rdaprir2/"
      ]
```

Si on cherche de l'information sur le préfixe `2001:200:1000::/48`, il correspond à deux entrées dans le tableau des services (`2001:200::/23` et `2001:200:1000::/36`) mais la règle du préfixe le plus long (le plus spécifique) fait qu'on va utiliser `2001:200:1000::/36`, et la requête finale sera donc `https://example.net/rdaprir2/ip/2001:200:1000::/48`.

Pour les domaines (section 4), les entrées des services sont des noms de domaine :

```
...
       "services": [
         [
           ["net", "com", "org"],
           [
             "https://registry.example.com/myrdap/"
           ]
         ],
         [
           ["foobar.org", "mytld"],
           [
             "https://example.org/"
           ]
         ],
...
```

L'entrée sélectionnée est la plus longue (en nombre de composants du nom, pas en nombre de caractères) qui correspond. Dans l'exemple ci-dessus, si on cherche des informations sur `foobar.org`, on ira voir le serveur RDAP en `https://example.org/`, si on en cherche sur `toto.org`, on ira en `https://registry.example.com/myrdap/`. (En pratique, aujourd'hui, le tableau des serveurs RDAP ne contient que des TLD.)

Pour les numéros d'AS (section 5.3), on se sert d'intervalles de numéros et de la syntaxe du RFC 5396 :

```
"services": [
         [
           ["2045-2045"],
           [
             "https://rir3.example.com/myrdap/"
           ]
         ],
         [
           ["10000-12000", "300000-400000"],
           [
             "http://example.org/"
           ]
         ],
...
```

Les entités (section 6 de notre RFC) posent un problème particulier, elles ne se situent pas dans un espace arborescent, contrairement aux autres objets utilisable avec RDAP. (Rappel : les entités sont les contacts, les titulaires, les BE…) Il n'existe donc pas de "*bootstrap service*" pour les entités (ni, d'ailleurs, pour les serveurs de noms, cf. section 9). En pratique, si une requête RDAP renvoie une réponse incluant un "*handle*" pour une entité, il n'est pas idiot d'envoyer les requêtes d'information sur cette entité au même serveur RDAP mais il n'est pas garanti que cela marche.

Notez (section 7) que le tableau `services` ne sera pas forcément complet et que certains objets peuvent ne pas s'y trouver. Par exemple, dans un tableau pour les TLD, les registres n'ayant pas encore de serveur RDAP ne seront logiquement pas cités. On peut toujours tenter un autre serveur, en espérant qu'il utilisera les redirections HTTP. Par exemple, ici, on demande au serveur RDAP de l'APNIC pour une adresse RIPE. On est bien redirigé avec un code 301 (RFC 7231, section 6.4.2) :

```
%  curl -v http://rdap.apnic.net/ip/2001:67c:288::2
...
> GET /ip/2001:67c:288::2 HTTP/1.1
> User-Agent: curl/7.38.0
> Host: rdap.apnic.net
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Date: Wed, 01 Apr 2015 13:07:00 GMT
< Location: http://rdap.db.ripe.net/ip/2001:67c:288::2
< Content-Type: application/rdap+json
...
```

La section 8 couvre quelques détails liés au déploiement de ce service. C'est que le "*Bootstrap Service*" est différent des autres registres IANA. Ces autres registres ne sont consultés que rarement (par exemple, lors de l'écriture d'un logiciel) et jamais en temps réel. Si le serveur HTTP de l'IANA plante, ce n'est donc pas trop grave. Au contraire, le "*Bootstrap Service*" doit marcher en permanence, car un client RDAP en a besoin. Pour limiter la pression sur les serveurs de l'IANA, notre RFC recommande que les clients RDAP ne consultent pas ce service à chaque requête RDAP, mais qu'au contraire ils mémorisent le JSON récupéré, en utilisant le champ `Expires:` de HTTP (RFC 7234, section 5.3) pour déterminer combien de temps doit durer cette mémorisation. Néanmoins, l'IANA a dû ajuster ses serveurs HTTP et louer les services d'un CDN pour assurer ce rôle relativement nouveau.

Le client RDAP doit d'autre part être conscient que le registre n'est pas mis à jour instantanément. Par exemple, si un nouveau TLD est ajouté par le gouvernement états-unien, via Verisign, TLD dont le registre a un serveur RDAP, cette information ne sera pas disponible immédiatement pour tous les clients RDAP.

Comme son ancêtre whois, RDAP soulève plein de questions de sécurité intéressantes, détaillées plus précisement dans le RFC 7481.

La section 12 de notre RFC détaille les processus IANA à l'œuvre. En effet, et c'est une autre différence avec les registres IANA habituels, il n'y a pas de mise à jour explicite des registres du "*bootstrap service*", ils sont mis à jour implicitement comme effet de bord des allocations et modifications d'allocation dans les registres d'adresses IPv4 <`https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml`>, adresses IPv6 <`https://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xml`>, numéros d'AS <`https://www.iana.org/assignments/as-numbers/as-numbers.xml`> et domaines <`https://www.iana.org/domains/root/db`>. Il a juste fallu modifier les procédures de gestion de ces registres existants, pour permettre d'indiquer le

serveur RDAP. Ainsi, le formulaire de gestion d'un TLD par son responsable a été modifié pour ajouter un champ "serveur RDAP" comme il y avait un champ "serveur Whois".

Aujourd'hui, les fichiers de ce service sont :

— `https://data.iana.org/rdap/dns.json`
— `https://data.iana.org/rdap/ipv4.json`
— `https://data.iana.org/rdap/ipv6.json`
— `https://data.iana.org/rdap/asn.json`

Voici, par exemple, celui d'IPv6 (notez le champ `Expires:` dans la réponse) :

```
%  curl -v https://data.iana.org/rdap/ipv6.json
...
* Connected to data.iana.org (2606:2800:11f:bb5:f27:227f:1bbf:a0e) port 80 (#0)
> GET /rdap/ipv6.json HTTP/1.1
> Host: data.iana.org
> User-Agent: curl/7.68.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Expires: Tue, 26 Apr 2022 12:30:52 GMT
< Last-Modified: Wed, 06 Nov 2019 19:00:04 GMT
...
{
  "description": "RDAP bootstrap file for IPv6 address allocations",
  "publication": "2019-11-06T19:00:04Z",
  "services": [
    [
      [
        "2001:4200::/23",
        "2c00::/12"
      ],
      [
        "https://rdap.afrinic.net/rdap/",
        "http://rdap.afrinic.net/rdap/"
...
```

Et celui des TLD :

```
% curl -v http://data.iana.org/rdap/dns.json
...
  "description": "RDAP bootstrap file for Domain Name System registrations",
  "publication": "2022-04-22T16:00:01Z",
  "services": [
    [
      [
        "nowruz",
        "pars",
        "shia",
        "tci",
        "xn--mgbt3dhd"
      ],
      [
        "https://api.rdap.agitsys.net/"
      ]
    ],
```

Testons si cela marche vraiment :

```
% curl -v https://api.rdap.agitsys.net/domain/www.nowruz
...
 "nameservers": [
    {
      "objectClassName": "nameserver",
      "ldhName": "ns1.aftermarket.pl.",
      "unicodeName": "ns1.aftermarket.pl."
    },
    {
      "objectClassName": "nameserver",
      "ldhName": "ns2.aftermarket.pl.",
      "unicodeName": "ns2.aftermarket.pl."
    }
  ]
```

Parfait, tout marche.

Il y a très peu de changements depuis le RFC précédent, le RFC 7484, quelques nettoyages seulement, et un changement de statut sur le chemin des normes.

Si vous aimez lire des programmes, j'ai fait deux mises en œuvre de ce RFC, la première en Python est incluse dans le code utilisé pour mon article « Récupérer la date d'expiration d'un domaine en RDAP <https://www.bortzmeyer.org/expiration-rdap.html> ». Le code principal est mais vous noterez que, contrairement à ce que demande le RFC dans sa section 8, la mémorisation du "*bootstreap registry*" n'utilise pas le champ HTTP `Expires:`. La deuxième, en Elixir, est plus correcte et est disponible dans . Lorsque la constante `@verbose` est vraie, le programme affiche les étapes. S'il n'a pas mémorisé de données :

```
% mix run find_rdap.exs quimper.bzh
Starting
No expiration known
Retrieving from https://data.iana.org/rdap/dns.json
Updating the cache
RDAP bootstrap file for Domain Name System registrations published on 2022-04-22T16:00:01Z
RDAP server for quimper.bzh is https://rdap.nic.bzh/
Retrieving RDAP info at https://rdap.nic.bzh/domain/quimper.bzh
{"rdapConformance":["rdap_level_0","...
```

S'il en a mémorisé :

```
% mix run find_rdap.exs quimper.bzh
Starting
Expiration is 2022-04-26 12:44:48Z
Using the cache ./iana-rdap-bootstrap.json
RDAP bootstrap file for Domain Name System registrations published on 2022-04-22T16:00:01Z
RDAP server for quimper.bzh is https://rdap.nic.bzh/
Retrieving RDAP info at https://rdap.nic.bzh/domain/quimper.bzh
{"rdapConformance":["rdap_level_0", ...
```

Mais il existe évidemment une mise en œuvre de ce RFC dans tous les clients RDAP comme :

— L'application de Viagenie <https://viagenie.ca/rdapbrowser/>,
— Celle de l'ICANN <https://lookup.icann.org> (application Web),
— L'ARIN a un service de redirection, `https://rdap-bootstrap.arin.net/bootstrap`, qui lit la base IANA et envoie les redirections appropriées, quand un client RDAP l'interroge.