RFC 7484 : Finding the Authoritative Registration Data (RDAP) Service

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 avril 2015

Date de publication du RFC : Mars 2015

https://www.bortzmeyer.org/7484.html

Le nouveau protocole RDAP <https://www.bortzmeyer.org/weirds-rdap.html> 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. Comment le client RDAP qui veut des informations sur 2001:67c:288::2 sait-il à quel serveur demander ? Ce fut une des plus chaudes discussions au sein du groupe de travail IETF qui a mis au point RDAP. 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.

Le protocole RDAP <https://www.bortzmeyer.org/weirds-rdap.html> est décrit entre autres dans le RFC 7480 [1] 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. 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/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. (À noter que c'est le

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7480.txt

premier registre IANA au format JSON.) 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). Actuellement, ces services sont encore vides :

```
% curl http://data.iana.org/rdap/ipv4.json
{
  "description": "RDAP bootstrap file for IPv4 address allocations",
  "publication": "2015-03-20T20:39:57Z",
  "services": [],
  "version": "1.0"
}
```

Mais dans le futur, ce sera un tableau donnant, pour des objets donnés, les URL des serveurs RDAP à contacter pour ces objets indiqués, par exemple :

```
 {
       "version": "1.0",
       "publication": "2024-01-07T10:11:12Z",
       "description": "RDAP Bootstrap file for example registries.",
       "services": [
         [
           ["1.0.0.0/8", "192.0.0.0/8"],
           [
             "https://rir1.example.com/myrdap/"
           ]
         ],
...
```

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 7482. Ainsi, cherchant des informations sur `192.0.2.24`, on ajoutera `ip/192.0.2.24` à l'URL de base ce qui, avec le "*bootstrap service*" précédent, donnerait `https://rir1.example.com/myrdap/ip/19`

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 4291. Voici un exemple IPv6 :

```
       "services": [
         [
           ["2001:200::/23", "2001:db8::/32"],
           [
             "https://rir2.example.com/myrdap/"
           ]
         ],
         [
           ["2600::/16", "2100:ffff::/32"],
           [
             "http://example.org/"
           ]
         ],
         [
           ["2001:200:1000::/36"],
           [
             "https://example.net/rdaprir2/",
             "http://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`. (Si elle échoue, on peut passer au deuxième URL du tableau, celui sans HTTPS.)

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"],
           [
             "http://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 `http://example.org/`, si on en cherche sur `toto.org`, on ira en `https://registry.example.com/myrdap/`.

Pour les numéros d'AS (section 5.3), on se sert d'intervalles de numéros :

```
"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.

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, qui a pris du temps et n'a été effectif que le 25 mars 2015, en pleine réunion IETF à Dallas. 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) 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 faudra juste modifier les procédures de gestion de ces registres existants, pour permettre d'indiquer le serveur RDAP (une information qui n'est pas récoltée aujourd'hui, ce qui explique pourquoi le "*bootstrap service*" est vide). Ainsi, le formulaire de gestion d'un TLD par son responsable devra être modifié pour ajouter un champ "serveur RDAP" comme il y a aujourd'hui un champ "serveur Whois".

Aujourd'hui, les fichiers de ce service sont :

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

Voici, par exemple, celui d'IPv6 (encore vide) :

```
% curl -v http://data.iana.org/rdap/ipv6.json
...
> GET /rdap/ipv6.json HTTP/1.1
> User-Agent: curl/7.38.0
> Host: data.iana.org
> Accept: */*
>
< HTTP/1.1 200 OK
< Accept-Ranges: bytes
< Content-Type: text/plain; charset=UTF-8
< Date: Wed, 01 Apr 2015 13:47:28 GMT
< Etag: "63ea7-96-511be51c6e140"
< Last-Modified: Fri, 20 Mar 2015 20:39:57 GMT
< Server: ECAcc (ewr/14C3)
< X-Cache: HIT
< Content-Length: 150
<
{
  "description": "RDAP bootstrap file for IPv6 address allocations",
  "publication": "2015-03-20T20:39:57Z",
  "services": [],
  "version": "1.0"
}
```

Et celui des TLD, qui nous montre les deux pionniers, la république tchèque et l'Argentine :

```
% curl -v http://data.iana.org/rdap/dns.json
{
  "description": "RDAP bootstrap file for Domain Name System registrations",
  "publication": "2016-12-09T04:10:13Z",
  "services": [
    [
      [
        "ar"
      ],
      [
        "https://rdap.nic.ar"
      ]
    ],
    [
      [
        "cz"
      ],
      [
        "https://rdap.nic.cz"
      ]
    ]
  ],
  "version": "1.0"
}
```

Testons si cela marche vraiment :

```
% curl -v https://rdap.nic.ar/domain/edu.ar
...
   "nameservers": [
      {
        "objectClassName": "nameserver",
```

```
    "handle": "noc.uncu.edu.ar",
    "links": [
      {
        "href": "https://rdap.nic.ar/nameserver/noc.uncu.edu.ar",
        "type": "application/rdap+json",
        "rel": "self",
        "value": "https://rdap.nic.ar/nameserver/noc.uncu.edu.ar"
      }
    ],
    "ldhName": "noc.uncu.edu.ar"
  },
  {
    "objectClassName": "nameserver",
    "handle": "ns1.mrecic.gov.ar",
    "links": [
      {
        "href": "https://rdap.nic.ar/nameserver/ns1.mrecic.gov.ar",
        "type": "application/rdap+json",
        "rel": "self",
        "value": "https://rdap.nic.ar/nameserver/ns1.mrecic.gov.ar"
      }
    ],
    "ldhName": "ns1.mrecic.gov.ar"
  },
  ...
```


Parfait, tout marche.

Question lecture, John Levine, dans son article sur RDAP <http://jl.ly/ICANN/weirds14.writeback>, parlait de ce problème de "*bootstrap*" : « "*The biggest delay in getting RDAP done turned out to be the bootstrap, figuring out where the server is for each top level domain, IP range, or ASN range. A lot of proposals that seemed reasonable, like a fixed name (http ://rdap.domain) or a DNS approach like SRV records turned out either to have administrative issues, or to be hard to implement (you can't do SRV lookups from Javascript.) There were also issues that were arguably technical or political (depending which side you're on) about specifying the URL syntax of the queries.*" » Notez qu'on parle parfois de "*bootstrap service*" pour désigner, non pas un ensemble de fichiers dirigeant vers le bon serveur RDAP, comme décrit dans ce RFC, mais un serveur qui ne fait que des redirections, comme celui (expérimental) en `rdap.org` qui, ici, redirige à juste titre vers Afilias :

```
% curl -v http://rdap.org/domain/rmll.info
...
> GET /domain/rmll.info HTTP/1.1
> User-Agent: curl/7.35.0
> Host: rdap.org
> Accept: */*
>
< HTTP/1.0 301 Moved Permanently
< Date: Thu, 26 Mar 2015 13:57:20 GMT
< Server: Apache
< X-Powered-By: PHP/5.3.3
< Content-Language: en
< Status: 301
< Location: http://rdg.afilias.info/.well-known/rdap/domain/rmll.info
< Content-Length: 0
< Content-Type: application/rdap+json; charset=UTF-8
```

D'ailleurs, si vous cherchez une mise en œuvre d'un tel redirecteur, il y a rdapbootstrap <http://projects.arin.net/rdapbootstrap/> (en Java).