RFC 7483 : JSON Responses for the Registration Data Access Protocol (RDAP)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 mars 2015

Date de publication du RFC : Mars 2015

http://www.bortzmeyer.org/7483.html

Dans l'ensemble des normes sur le protocole RDAP <http://www.bortzmeyer.org/weirds-rdap.html>, ce RFC est destiné à décrire le format de **sortie**, celui des réponses envoyées par le serveur d'information RDAP. Ce format est basé sur JSON et, pour utiliser RDAP en remplacement de whois, il faudra donc se munir de quelques outils pour traiter le JSON.

JSON est normalisé dans le RFC 8259 [1] et représente aujourd'hui le format de choix pour les données structurées c'est-à-dire analysables par un programme. (L'annexe E de notre RFC explique en détail le pourquoi du choix de JSON.) Une des caractéristiques importantes de whois est en effet que son format de sortie n'a aucune structure, c'est du texte libre, ce qui est pénible pour les programmeurs qui essaient de le traiter (mais, d'un autre côté, cela ralentit certains usages déplorables, comme la récolte d'adresses de courrier à des fins de spam, jouant ainsi un rôle de "*semantic firewall*"). L'utilisation typique de RDAP est en HTTP (RFC 7480) avec les requêtes exprimées dans la syntaxe des URL du RFC 7482. Voici tout de suite un exemple réel, avec la réponse JSON :

```
% curl http://rdg.afilias.info/rdap/domain/kornog-computing.info
...
{
   "entities": [
...
         "objectClassName": "entity",
         "roles": [
            "technical",
            "billing",
            "administrative"
         ],
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8259.txt

```
         "vcardArray": [
            "vcard",
            [
               [
                  "version",
                  {},
                  "text",
                  "4.0"
               ],
               [
                  "fn",
                  {},
                  "text",
                  "Simon TOSSER"
               ],
               [
                  "fn",
                  {},
                  "text",
                  "KORNOG computing"
               ],
               [
                  "adr",
                  {},
                  "text",
                  [
                     "",
                     "",
                     "26b rue d'Aiguillon",
                     "MORLAIX",
                     "",
                     "29600",
                     "FR"
                  ]
               ],
...
   "events": [
      {
         "eventAction": "registration",
         "eventDate": "2007-08-14T17:02:33Z"
      },
      {
         "eventAction": "last changed",
         "eventDate": "2014-08-14T01:54:07Z"
      },
      {
         "eventAction": "expiration",
         "eventDate": "2015-08-14T17:02:33Z"
      }
   ],
   "handle": "D19378523-LRMS",
   "lang": "en",
   "ldhName": "kornog-computing.info",
...
   "nameservers": [
      {
         "ldhName": "ns1.kornog-computing.net",
         "objectClassName": "nameserver",
         "remarks": [
            {
               "description": [
                  "Summary data only. For complete data, send a specific query for the object."
               ],
               "title": "Incomplete Data",
               "type": "object truncated due to unexplainable reasons"
            }
         ]
      },
      {
```

```
         "ldhName": "ns2.kornog-computing.net",
         "objectClassName": "nameserver",
         "remarks": [
            {
               "description": [
                  "Summary data only. For complete data, send a specific query for the object."
               ],
               "title": "Incomplete Data",
               "type": "object truncated due to unexplainable reasons"
            }
         ]
      }
   ],
   "notices": [
      {
         "description": [
            "Access to AFILIAS WHOIS information is provided to assist persons in determining the contents of a
         ],
         "title": "TERMS OF USE"
      }
   ],
   "objectClassName": "domain",
...
   "rdapConformance": [
      "rdap_level_0"
   ],
   "secureDNS": {
      "zoneSigned": false
   },
   "status": [
      "clientDeleteProhibited -- http://www.icann.org/epp#clientDeleteProhibited",
      "clientTransferProhibited -- http://www.icann.org/epp#clientTransferProhibited"
   ]
}
```


La section 1.2 présente le modèle de données de RDAP. Dans les réponses, on trouve des types simples (comme des chaînes de caractères), des tableaux JSON, et des objets JSON qui décrivent les trucs sur lesquels on veut de l'information (noms de domaine, adresses IP, etc). Ces objets peuvent comporter des tableaux ou d'autres objets et, lorsqu'une réponse renvoie plusieurs objets (cas des recherches ouvertes, cf. RFC 7482, section 3.2), peuvent eux-même être regroupés en tableaux. Il existe plusieurs classes de ces objets. D'abord, les classes communes aux registres de noms de domaine et aux RIR :

— Domaines (oui, même pour les RIR, pensez à `in-addr.arpa` et `ip6.arpa`),
— Serveurs de noms,
— Entités diverses, comme les "*handles*" des contacts.

Deux classes sont spécifiques aux RIR :

— Réseaux IP, identifiés par leur préfixe,
— Systèmes autonomes.

On verra peut-être apparaître d'autres classes avec le temps, si l'usage de RDAP se répand.

La section 2 de notre RFC décrit comment JSON est utilisé. Le type MIME renvoyé est `application/rdap+json`. La section 2 commence par un rappel que le client RDAP **doit** ignorer les membres inconnus dans les objets JSON, afin de préserver la future extensibilité. (Cette règle ne s'applique pas au jCard - cf. RFC 7095 - embarqué.) Si un serveur veut ajouter des membres qui lui sont spécifiques, il est recommandé qu'il préfixe leur nom avec un identificateur court suivi d'un tiret bas. Ainsi, cet objet RDAP standard :


```
{
     "handle" : "ABC123",
     "remarks" :
     [
       {
         "description" :
```

```
        [
          "She sells sea shells down by the sea shore.",
          "Originally written by Terry Sullivan."
        ]
      }
    ]
  }
```

S'il est servi par le « registre de la Lune » (registre imaginaire qui fournit les exemples de ce RFC), celui-ci, lorsqu'il voudra ajouter des membres, les précédera de `lunarNIC_` :

```
{
     "handle" : "ABC123",
     "lunarNic_beforeOneSmallStep" : "TRUE THAT!",
     "remarks" :
     [
       {
         "description" :
         [
           "She sells sea shells down by the sea shore.",
           "Originally written by Terry Sullivan."
         ]
       }
     ],
     "lunarNic_harshMistressNotes" :
     [
       "In space,",
       "nobody can hear you scream."
     ]
  }
```

(Je vous laisse décoder les références geeks dans l'exemple.)

La section 3 s'occupe des types de données de base. On utilise du JSON normal, tel que spécifié dans le RFC 8259, avec ses chaînes de caractères, ses booléens, son `null`... Un "*handle*" (l'identifiant unique - par registre - d'un contact, parfois nommé "*NIC handle*" ou "*registry ID*") est une chaîne. Les adresses IP se représentent également par une chaîne de caractères (ne pas oublier le RFC 5952 pour IPv6). Les pays sont identifiés par le code à deux lettres de ISO 3166. Les noms de domaines peuvent être sous une forme ASCII ou bien en Unicode. Les dates et heures suivent le RFC 3339, et les URI le RFC 3986.

Les informations plus complexes (comme les détails sur un contact) utilisent jCard, normalisé dans le RFC 7095.

Que trouve-t-on dans les objets JSON renvoyés par un serveur RDAP ? Une indication de la version de la norme :

```
"rdapConformance" :
   [
     "rdap_level_0"
   ]
```

Et des liens vers des ressources situées ailleurs, suivant le cadre du RFC 8288 :

```
        "links": [
           {
              "href": "http://rdg.afilias.info/rdap/entity/ovh53ec16bekre5",
              "rel": "self",
              "type": "application/rdap+json",
              "value": "http://rdg.afilias.info/rdap/entity/ovh53ec16bekre5"
           }
```

On peut aussi avoir du texte libre, comme dans l'exemple plus haut avec le membre `remarks`. Ce membre sert aux textes décrivant la classe, alors que `notices` est utilisé pour le service RDAP. Un exemple :

```
   "notices": [
      {
         "description": [
            "Access to AFILIAS WHOIS information is provided to assist persons in determining the contents of a
         ],
         "title": "TERMS OF USE"
      }
   ],
```

Contrairement à whois, RDAP permet l'internationalisation sous tous ses aspects. Par exemple, on peut indiquer la langue des textes avec une étiquette de langue (RFC 5646) :

```
   "lang": "en",
```

Enfin, la classe (le type) de l'objet renvoyé par RDAP est indiqué par un membre `objectClassName` :

```
   "objectClassName": "domain",
```

Ces classes, justement. La section 5 les décrit. Il y a d'abord la classe `Entity` qui correspond aux requêtes `/entity` du RFC 7482. Elle sert à décrire les personnes et les organisations. Parmi les membres importants pour les objets de cette classe, `handle` qui est un identificateur de l'instance de la classe, et `roles` qui indique la relation de cette instance avec l'objet qui la contient (par exemple `"roles": ["technical"]` indiquera que cette `Entity` est le contact technique de l'objet). L'information de contact sur une `Entity` se fait avec le format jCard du RFC 7095, dans un membre `vcardArray`. Voici un exemple :

```
% curl http://rdg.afilias.info/rdap/entity/ovh53ec16bekre5
...
        "objectClassName": "entity",
        "roles": [
           "technical",
           "billing",
           "administrative"
        ],
        "vcardArray": [
           "vcard",
           [
              [
                 "version",
                 {},
                 "text",
                 "4.0"
```

```
            ],
            [
               "fn",
               {},
               "text",
               "Simon TOSSER"
            ],
            [
               "fn",
               {},
               "text",
               "KORNOG computing"
            ],
            [
               "adr",
               {},
               "text",
               [
                  "",
                  "",
                  "26b rue d'Aiguillon",
                  "MORLAIX",
                  "",
                  "29600",
                  "FR"
               ]
            ],
            [
               "email",
               {},
               "text",
               "simon.tosser@gmail.com"
            ],
            [
               "tel",
               {
                  "type": "work"
               },
               "uri",
               "tel:+33.661463099"
            ],
         ]
      ]
```

La classe `Nameserver` correspond aux requêtes `/nameserver` du RFC 7482. Notez qu'un registre peut gérer les serveurs de noms de deux façons : ils peuvent être vus comme des objets autonomes, enregistrés tels quel dans le registre (par exemple via le RFC 5732), ayant des attributs par exemple des contacts, et interrogeables directement par whois ou RDAP (c'est le modèle de `.com`, on dit alors que les serveurs de noms sont des « objets de première classe »). Ou bien ils peuvent être simplement des attributs des domaines, accessibles via le domaine (c'est le modèle de `.fr` : regardez `ssi.gouv.fr`, bien que les serveurs de noms soient dans la zone, vous ne pouvez pas demander directement des informations sur ces serveurs via whois). Le principal attribut d'un objet `Nameserver` est son adresse IP, pour pouvoir générer des colles dans le DNS (enregistrement DNS d'une adresse IP, pour le cas où le serveur de noms est lui-même dans la zone qu'il sert). Voici un exemple avec un des serveurs de noms de la zone `afilias-nst.info` :

```
% curl http://rdg.afilias.info/rdap/nameserver/b0.dig.afilias-nst.info
...
   "ipAddresses": {
      "v4": [
         "65.22.7.1"
      ],
      "v6": [
         "2a01:8840:7::1"
```

```
      ]
   },
...
```

Notez que l'adresse IP est un tableau, un serveur pouvant avoir plusieurs adresses.

La classe `Domain` correspond aux requêtes `/domain` du RFC 7482. Un objet de cette classe a des membres indiquant les serveurs de noms, si la zone est signée avec DNSSEC ou pas, l'enregistrement DS si elle est signée, le statut (actif ou non, bloqué ou non), les contacts, etc. Voici un exemple :

```
% curl http://rdg.afilias.info/rdap/domain/afilias-nst.info
...
   "nameservers": [
      {
         "ldhName": "a0.dig.afilias-nst.info",
...
   "secureDNS": {
      "zoneSigned": false
   },
   "status": [
      "clientTransferProhibited -- http://www.icann.org/epp#clientTransferProhibited",
      "serverDeleteProhibited -- http://www.icann.org/epp#serverDeleteProhibited"
   ]
...
```

La classe `IP network` rassemble les objets qu'on trouve dans les réponses aux requêtes `/ip` du RFC 7482. Un objet de cette classe ne désigne en général pas une seule adresse IP mais un préfixe, dont on indique la première (`startAddress`) et la dernière adresse (`endAddress`). Personnellement, je trouve cela très laid et j'aurai préféré qu'on utilise une notation préfixe/longueur. Voici un exemple :

```
% curl http://rdap.db.ripe.net/ip/131.111.150.25
...
{
  "handle" : "131.111.0.0 - 131.111.255.255",
  "startAddress" : "131.111.0.0/32",
  "endAddress" : "131.111.255.255/32",
  "ipVersion" : "v4",
  "name" : "CAM-AC-UK",
  "type" : "LEGACY",
  "country" : "GB",
...
```

La dernière classe normalisée à ce stade est `autnum` (les AS), en réponse aux requêtes `/autnum` du RFC 7482. Elle indique notamment les contacts de l'AS. Pour l'instant, il n'y a pas de format pour indiquer la politique de routage (RFC 4012). Un exemple d'un objet de cette classe :

```
% curl  http://rdap.db.ripe.net/autnum/20766
{
  "handle" : "AS20766",
  "name" : "GITOYEN-MAIN-AS",
  "type" : "DIRECT ALLOCATION",
...
    "handle" : "GI1036-RIPE",
    "vcardArray" : [ "vcard", [ [ "version", { }, "text", "4.0" ], [ "fn", { }, "text", "NOC Gitoyen" ], [ "kind
      "label" : "Gitoyen\n21 ter rue Voltaire\n75011 Paris\nFrance"
    }, "text", null ], [ "email", { }, "text", "noc@gitoyen.net" ] ] ],
...
```

Comme, dans la vie, il y a parfois des problèmes, une section de notre RFC, la section 6, est dédiée aux formats des erreurs que peuvent indiquer les serveurs RDAP. Le code de retour HTTP fournit déjà des indications (404 = cet objet n'existe pas ici, 403 = vous n'avez pas le droit de le savoir, etc) mais on peut aussi ajouter un objet JSON pour en indiquer davantage, objet ayant un membre `errorCode` (qui reprend le code HTTP), un membre `title` et un membre `description`. Voici un exemple sur le serveur RDAP expérimental de l'ARIN :

```
% curl -v http://rdappilot.arin.net/restfulwhois/rdap/autnum/99999
< HTTP/1.0 404 Not Found
< Date: Thu, 26 Mar 2015 16:02:07 GMT
< Server: Apache/2.2.3 (CentOS)
...
{
...
  "errorCode" : 404,
  "title" : "AUTNUM NOT FOUND",
  "description" : [ "The autnum you are seeking as '99999' is/are not here." ]
```

Plus positive, la possibilité de demander de l'aide à un serveur RDAP, en se renseignant sur ses capacités, avec la requête `/help`. Son résultat est décrit dans la section 7 mais les serveurs RDAP actuels ne semblent pas encore utiliser cette possibilité. L'exemple qui suit est donc purement théorique :

```
{
     "rdapConformance" :
     [
       "rdap_level_0"
     ],
     "notices" :
     [
       {
         "title" : "Authentication Policy",
         "description" :
         [
           "Access to sensitive data for users with proper credentials."
         ],
         "links" :
         [
           {
             "value" : "http://example.net/help",
             "rel" : "alternate",
             "type" : "text/html",
             "href" : "http://www.example.com/auth_policy.html"
           }
         ]
       }
     ]
   }
```

Et les résultats des recherches ouvertes (section 3.2 du RFC 7482), qui peuvent renvoyer plusieurs objets ? Ce sont des tableaux JSON, dans des membres dont le nom se termine par `Results`. Par exemple, en cherchant les noms de domaines commençant par `ra` :

```
% curl http://rdg.afilias.info/rdap/domains\?name=ra\*|more
   "domainSearchResults": [
      {
         "ldhName": "RAINSTRAGE.INFO",
         ...
         "objectClassName": "domain",
         "remarks": [
```

```
            {
               "description": [
                  "Summary data only. For complete data, send a specific query for the object."
               ],
               "title": "Incomplete Data",
               "type": "object truncated due to unexplainable reasons"
            }
...
         "ldhName": "RADONREMOVAL.INFO",
...
         "ldhName": "RANCONDI.INFO",
...
```

Vous avez peut-être noté dans le tout premier exemple le membre `events` (section 4.5 du RFC). Ces événements comme `created` ou `last-changed` donnent accès à l'histoire d'un objet enregistré. Ici, nous apprenons que le domaine `kornog-computing.info` a été enregistré en 2007.

Certaines valeurs qui apparaissent dans les résultats sont des chaînes de caractères fixes, stockées dans un nouveau registre IANA <https://www.iana.org/assignments/rdap-json-values/rdap-json-values.xhtml>. Elles sont conçues pour être utilisées dans les `notices`, `remarks`, `status`, `roles` et quelques autres. Parmi les remarques, on trouvera le cas où une réponse a été tronquée (section 9 du RFC), comme dans l'exemple ci-dessus avec la mention "*Incomplete Data*". Parmi les statuts, on trouvera, par exemple `validated` (pour un objet vérifié, par exemple un nom de domaine dont on a vérifié les coordonnées du titulaire), `locked` (pour un objet verrouillé), `obscured` (qui n'est pas un statut dans le base du données du registre mais simplement la mention du fait que le serveur RDAP a délibérement modifié certaines informations qu'il affiche, par exemple pour protéger la vie privée), etc. Pour les rôles, on trouvera `registrant` (titulaire), `technical` (contact technique), etc.

Pour ceux qu'intéressent les questions d'internationalisation, la section 12 contient d'utiles mentions. L'encodage des données JSON **doit** être de l'UTF-8. Et, comme indiqué plus haut, les IDN peuvent être sous la forme Punycode ou bien directement en UTF-8.

Et la vie privée, un problème permanent avec whois, où il faut toujours choisir entre la distribution de données utiles pour contacter quelqu'un et les risques pour sa vie privée ? La section 13 revient sur cette question. Un point important : RDAP est un protocole, pas une politique. Il ne définit pas quelles règles suivre (c'est de la responsabilité des divers registres) mais il fournit les outils pour mettre en œuvre ces règles. Notamment, RDAP permet de marquer des parties de la réponse comme étant connues du registre, mais n'ayant délibérement pas été envoyées (avec les codes `private` et `removed`) ou bien comme ayant été volontairement rendues peu ou pas lisibles (code `obscured`).

Vous avez vu dans les exemples précédents que les réponses d'un serveur RDAP sont souvent longues et, a priori, moins lisibles que celles d'un serveur whois. Il faudra souvent les traiter avec un logiciel qui comprend le JSON. Un exemple très simple et que j'apprécie est jq <http://www.bortzmeyer.org/jq.html>. Il peut servir à présenter le résultat de manière plus jolie :

```
% curl  https://rdap.centralnic.com/domain/centralnic.pw  | jq ""
{
  "cnic_recordGenerated": "2015-03-26T16:11:20.0Z",
  "lang": "en",
  "notices": [
    {
      "links": [
        {
          "type": "application/json",
          "media": "screen",
```

```
          "rel": "item",
          "href": "https://www.centralnic.com/registry/labs/rdap#service-level",
          "value": "https://www.centralnic.com/registry/labs/rdap#service-level",
          "title": "Service Level"
        }
      ],
      "description": [
        "The RDAP service is currently an alpha quality product. CentralNic makes no warranty of it
s correctness, usefulness or reliability. CentralNic reserves the right to change or remove part or
 all of the service at any time with no prior warning. No service level is offered."
      ],
      "title": "Service Level"
    },
    {
      "links": [
        {
...
```

(Essayez ce même serveur RDAP sans jq !)

Mais on peut aussi se servir de jq pour extraire un champ particulier, ici le pays :

```
% curl -s  http://rdap.db.ripe.net/ip/131.111.150.25 | jq ".country"
"GB"

% curl -s  http://rdap.db.ripe.net/ip/192.134.1.1 | jq ".country"
"FR"
```

Il y a évidemment d'autres logiciels que jq sur ce créneau, comme JSONpath <http://goessner.net/articles/JsonPath/>, jpath <http://blogs.perl.org/users/perlancar/2014/12/day-1-slici
html> ou, pour les programmeurs Python, `python -m json.tool`.

Un dernier mot, sur le choix de JSON pour le format de sortie, alors que le protocole standard d'avitaillement des objets dans les bases Internet, EPP (RFC 5730) est en XML. L'annexe E de notre RFC, qui discute ce choix, donne comme principaux arguments que JSON est désormais plus répandu que XML (cf. l'article « *"The Stealthy Ascendancy of JSON"* <https://devcentral.f5.com/articles/the-stealthy-ascendancy-of-json> ») et que c'est surtout vrai chez les utilisateurs (EPP étant utilisé par une population de professionnels bien plus réduite).