RFC 8427 : Representing DNS Messages in JSON

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 décembre 2018

Date de publication du RFC : Juillet 2018

https://www.bortzmeyer.org/8427.html

Le format des messages DNS circulant sur le réseau est un format binaire, pas forcément évident à analyser. Pour beaucoup d'applications, il serait sans doute préférable d'utiliser un format normalisé et plus agréable, par exemple JSON, dont ce nouveau RFC décrit l'utilisation pour le DNS.

Non seulement le format des messages DNS est du binaire (RFC 1035 [1], section 4) et non pas du texte comme par exemple pour SMTP, XMPP ou HTTP, mais en plus il y a des pièges. Par exemple, la longueur des sections du message est indiquée dans un champ séparé de la section, et peut ne pas correspondre à la vraie longueur. La compression des noms n'arrange rien. Écrire un analyseur de messages DNS est donc difficile.

Il y a un million de formats pour des données structurées mais, aujourd'hui, le format texte le plus populaire pour ces données est certainement JSON, normalisé dans le RFC 8259. L'utilisation de JSON pour représenter les messages DNS (requêtes ou réponses) suit les principes suivants (section 1.1 du RFC) :

— Tout est optionnel, on peut donc ne représenter qu'une partie d'un message. (Des profils ultérieurs de cette spécification pourraient être plus rigoureux, par exemple une base de "*passive DNS*" <https://www.bortzmeyer.org/dnsdb.html> peut imposer que QNAME, le nom de domaine demandé, soit présent, cf. section 6 du RFC.)
— Si l'information est présente, on peut recréer le format binaire utilisé par le protocole DNS à partir du JSON (pas forcément bit pour bit, surtout s'il y avait de la compression).
— Tous les noms sont représentés dans le sous-ensemble ASCII d'UTF-8, obligeant les IDN à être en Punycode (alors que JSON permet l'UTF-8).
— Les données binaires peuvent être stockées, à condition d'être encodées en Base16 (RFC 4648).

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc1035.txt

— Il n'y a pas de forme canonique : le même message DNS peut être représenté par plusieurs objets JSON différents.
— Certains membres des objets JSON sont redondants et il peut donc y avoir des incohérences (par exemple entre la longueur indiquée d'une section ou d'un message, et sa longueur réelle, cf. la section 8 sur les conséquences que cela a pour la sécurité). Cela est nécessaire pour reconstituer certains messages DNS malformés. Autrement, le format serait limité aux messages corrects.
— Le format représente des messages, pas des fichiers de zone (RFC 1035, section 5).

La section 2 du RFC donne la liste des membres (au sens JSON de « champs d'un objet ») d'un objet DNS. Voici un exemple d'un tel objet, une requête DNS demandant l'adresse IPv4 (code 1, souvent noté A) d'`example.com` :

```
{ "ID": 19678, "QR": 0, "Opcode": 0,
     "AA": 0, "TC": 0, "RD": 0, "RA": 0, "AD": 0, "CD": 0, "RCODE": 0,
     "QDCOUNT": 1, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0,
     "QNAME": "example.com", "QTYPE": 1, "QCLASS": 1
   }
```

Les noms des membres sont ceux utilisés dans les RFC DNS, même s'ils ne sont pas très parlants (`RCODE` au lieu `ReturnCode`).

On note que les différents membres qui sont dans le DNS représentés par des entiers le sont également ici, au lieu d'utiliser les abréviations courantes. Ainsi, `Opcode` est marqué 0 et pas Q ("*query*"), et `QTYPE` ("*query type*") est marqué 1 et pas A (adresse IPv4). Cela permet de représenter des valeurs inconnues, qui n'ont pas d'abréviation textuelle, même si ça rend le résultat peu lisible si on ne connait pas les valeurs des paramètres DNS <`https://www.iana.org/assignments/dns-parameters/dns-parameters.xml`> par coeur.

Les valeurs d'un seul bit (booléens) sont représentés par 0 ou 1, pas par les `false` ou `true` de JSON. (J'avoue ne pas bien comprendre ce choix.)

On note également que les longueurs des sections sont indiquées explicitement, ici `QDCOUNT` ("*Query Count*", et ne me demandez pas à quoi sert le D après le Q, le RFC 1035 ne l'explique pas). En JSON, cela n'est pas obligatoire (la longueur d'un tableau, en JSON, n'est pas spécifiée explicitement) mais, comme expliqué plus haut, cela a été décidé pour permettre de représenter des messages DNS anormaux, par exemple ayant un `QDCOUNT` de 0 et une question dans la section Question (cf. section 8 du RFC sur les conséquences que cela peut avoir pour la sécurité). De tels messages arrivent assez souvent dans le trafic DNS réel vu par les serveurs connectés à l'Internet ; attaque délibérée ou bien logiciel écrit avec les pieds ?

Et voici un exemple de réponse (`QR` = 1) DNS en JSON. La requête a été un succès (`RCODE` = 0) :

```
{ "ID": 32784, "QR": 1, "AA": 1, "RCODE": 0,
  "QDCOUNT": 1, "ANCOUNT": 2, "NSCOUNT": 1,
   ARCOUNT": 0,
   "answerRRs": [ { "NAME": "example.com.",
                                        "TYPE": 1, "CLASS": 1,
                                        "TTL": 3600,
                                        "RDATAHEX": "C0000201" },
                  { "NAME": "example.com.",
                                        "TYPE": 1, "CLASS": 1,
                                        "TTL": 3600,
                                        "RDATAHEX": "C000AA01" } ],
   "authorityRRs": [ { "NAME": "ns.example.com.",
                                            "TYPE": 1, "CLASS": 1,
                                            "TTL": 28800,
                                            "RDATAHEX": "CB007181" } ]
```

La réponse contient un ensemble d'adresses IP (`TYPE` = 1 identifie une adresse IPv4), `192.0.2.1` et `192.0.170.1`. Leur valeur est encodée en hexadécimal. C'est moins joli que si on avait mis l'adresse IP en clair mais c'est plus général : cela permet d'inclure immédiatement de nouveaux types de données, au détriment de la lisibilité pour les anciens types.

Le format de ce RFC permet aussi de décrire l'association entre une requête et une réponse (section 3 du RFC). On les met dans un objet JSON ayant un membre `queryMessage` et un `responseMessage`.

Si on représente une suite continue de messages DNS, faire un objet JSON avec son accolade ouvrante et la fermante correspondante peut ne pas être pratique. On utilise alors les séquences du RFC 7464, décrites dans la section 4 de notre RFC.

Notre RFC spécifie également (section 7) un type MIME pour le DNS en JSON, `application/dns+json`.

Notez qu'il ne s'agit pas de la première description du DNS en JSON. Par exemple, j'avais décrit un format pour cela dans le brouillon `draft-bortzmeyer-dns-json`. Ce format est mis en œuvre dans le "*DNS Looking Glass*" <https://www.bortzmeyer.org/dns-lg.html>. Mon format était plus joli, car utilisant toujours des noms plus parlants (`"Type": "AAAA"` au lieu du `"QTYPE": 28`). Notez toutefois que le RFC 8427 le permet également (`"QTYPEname": "AAAA"`). Le format plus joli ne peut de toute façon pas être utilisé systématiquement car il ne permet pas de représenter les types inconnus. Et mon format ne permet pas non plus de représenter les messages malformés. (Par exemple, le `ANCOUNT` est toujours implicite.)

Un autre exemple de représentation des données DNS est donné par les sondes RIPE Atlas <https://atlas.ripe.net/>. Le fichier des résultats d'une mesure est en JSON (ce qui permet le traitement par les outils JSON habituels comme jq <https://labs.ripe.net/Members/stephane_bortzmeyer/processing-ripe-atlas-results-with-jq>). Voici un exemple (si vous voulez un exemple complet, téléchargez par exemple le résultat de la mesure #18061873 <https://atlas.ripe.net/api/v2/measurements/18061873/results/>) :

```
       "resultset": [
      {
...
        "result": {
          "ANCOUNT": 1,
          "ARCOUNT": 0,
          "ID": 38357,
          "NSCOUNT": 0,
          "QDCOUNT": 1,
          "abuf": "ldWBgAABAAEAAAAADmN5YmVyc3RydWN0dXJlAmZyAAAcAAHADAAcAAEAAVGAABAgAUuYDcAAQQIWPv/+Jz0/",
          "rt": 103.7,
          "size": 63
        },
      },
```

On note que seule une petite partie des champs de la réponse (`ANCOUNT`, `ID`...) est exprimée en JSON, la majorité de la requête étant dans un membre `abuf` qui est la représentation binaire de la réponse.

Le service de DNS sur HTTPS de Google produit également du JSON (on le passe à jq pour qu'il soit plus joliment affiché) :

```
% curl -s https://dns.google.com/resolve\?name=laquadrature.net\&type=MX | jq .
{
  "Status": 0,
  "TC": false,
  "RD": true,
  "RA": true,
  "AD": false,
  "CD": false,
  "Question": [
    {
      "name": "laquadrature.net.",
      "type": 15
    }
  ],
  "Answer": [
    {
      "name": "laquadrature.net.",
      "type": 15,
      "TTL": 475,
      "data": "5 pi.lqdn.fr."
    }
  ]
}
```

en utilisant un format spécifique à Google <https://developers.google.com/speed/public-dns/docs/dns-over-https>. On notera que le protocole DoH, normalisé dans le RFC 8484, n'utilise **pas** JSON (et le service de Google, contrairement à ce qu'on voit parfois écrit, n'utilise pas DoH) mais le format binaire du DNS, utilisant le type MIME `application/dns-message`. Le RFC 8484 prévoit toutefois la possibilité de se servir de JSON pour le futur (section 4.2 du RFC 8484).