Network Working Group   S. Bortzmeyer
Internet-Draft   AFNIC
Intended status: Experimental   February 25, 2013
Expires: August 29, 2013

                   JSON format to represent DNS data
                      draft-bortzmeyer-dns-json-01

Abstract

   This document describes a profile of JSON to represent DNS data.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on August 29, 2013.

Table of Contents

1. Introduction

   The JavaScript Object Notation (JSON) format is specified in
   [RFC4627]. It is a structured data format suitable for a wide range
   of applications. It is specially popular on the Web, due to its
   JavaScript roots, but can be found in many other contexts.

   The Domain Name System (DNS) is specified in [RFC1034] and [RFC1035].
   It is one of the most important infrastructure components of the
   Internet. DNS data is today typically exchanged using two formats:
   the "zone file" format (partially) described in section 5 of
   [RFC1035] and the "wire format" of the section 4 for [RFC1035].
   Other formats have been suggested, for an easier exchange of data, or
   for using DNS in new applications, such as DNS "looking glasses" or
   gateways to get DNS data over protocols such as HTTP ([RFC2616]).

   For instance, a mechanism have been suggested for DNS data in XML, in
   [I-D.mohan-dns-query-xml].

   This document suggests using the JSON format to represent DNS data.
   Note that a similar JSON-like (rather than JSON) description of DNS
   data already exists in [getdns].

   Also note that some representations of DNS data use a data model
   which is quite close from the JSON one, even if the concrete syntax
   is different (for instance [dnspython]).

2. Requirements notation

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].

3. The format

3.1. General rules

Most data is represented by JSON objects, with their named members.
It is common to omit some of these members, to save bandwidth or by
pure lazyness. So, clients who consume this sort of JSON objects
should not assume every member is present. THIS IS AN IMPORTANT RULE
(see Section 3.6, Paragraph 2 for a discussion).

3.2. Resource records

DNS resource records are JSON objects. The following members are
common to all record types:

o Name (owner name)

o Type

o Class

o Time to live (TTL)

The other members depend on the record type. The following list
gives the resource record type mnenomic and the JSON members for this
type:

o A:

   * Address

o AAAA:

   * Address

o MX:

   * Preference

   * MailExchanger

o NS:

   * Target

o PTR:

   * Target

o CNAME:

     * Target

o SOA:

     * MaintainerName

     * MasterServerName

     * Serial

     * Refresh

     * Retry

     * Expire

     * NegativeTtl

o DNSKEY:

     * Algorithm

     * Length

     * Flags

     * Tag

o DS:

     * DelegationKey

     * DigestType

o DLV:

     * DelegationKey

     * DigestType

o NSEC3PARAM:

     * Algorihm

     * Flags

   *   Salt

   *   Iterations

o   SSHFP:

   *   Algorithm

   *   DigestType

   *   Fingerprint

o   NAPTR:

   *   Flags

   *   Order

   *   Services

   *   Preference

   *   Regexp

   *   Replacement

o   SRV:

   *   Server

   *   Port

   *   Priority

   *   Weight

o   LOC:

   *   Longitude

   *   Latitude

   *   Altitude

o   SPF:

   *   Text

Note there is no concept of resource record sets (see Section 3.6,
Paragraph 3 for a discussion).

3.3. DNS response

A DNS response is represented as a JSON object with a member named
"Query". The main members of this object (the names are self-
explanatory) are:

o QuestionSection

o AnswerSection

o AdditionalSection

o AuthoritySection

o ReturnCode (alphabetical, e.g. NOERROR, NXDOMAIN, SERVFAIL, etc)

o ID

o AA (Authoritative Answer)

o TC (TrunCation)

o RD (Recursion Desired)

o RA (Recursion Available)

o AD (Authentic Data)

o Query

The Question Section is an object with members Qname, Qtype and
Qclass. The other three sections are JSON arrays, each DNS record is
an item in the array. They may be empty arrays (for instance, if the
request returns NOERROR,ANSWER=0, the AnswerSection will be an empty
array).

The Query object has members about the query: Duration is the time
taken to process the request, Server the resolver used (preferably as
an IP address).

3.4. Zone file

A DNS zone file is represented as a JSON object with a member named
"Zone". The main member of this object is an array of resource
records.

The member "Name" cannot be ommitted in resource records (unlike the
text format of [RFC1035], JSON does not guarantee the order of
records, so the trick of "previous resource record" does not work).
But you can use relative names, and @ to denote the origin.

3.5. Examples

```
{"Query": {"Server": "127.0.0.1"},
 "AnswerSection": [
       {"Name": "bortzmeyer.fr.",
        "TTL": 3600,
        "MasterServerName": "ns3.bortzmeyer.org.",
        "MaintainerName": "hostmaster.bortzmeyer.org.",
        "Serial": 2012060801, "Expire": 604800,
        "Refresh": 10800, "Retry": 3600,
        "NegativeTTL": 10800,
        "Type": "SOA"}],
 "ReturnCode": "NOERROR",
 "AD": true,
 "QuestionSection": {"Qtype": "SOA", "Qname": "bortzmeyer.fr."}}
}
```

                An answer with a SOA resource record

```
{"Query": {"Duration": "0.167317", "Server": "127.0.0.1"},
 "AnswerSection": [
    {"Name": "facebook.com", "TTL": 6666, "Type": "AAAA",
     "Address": "2a03:2880:10:8f01:face:b00c::25"},
    {"Name": "facebook.com", "TTL": 6666, "Type": "AAAA",
     "Address": "2a03:2880:2110:3f01:face:b00c::"},
    {"Name": "facebook.com", "TTL": 6666, "Type": "AAAA",
     "Address": "2a03:2880:10:1f02:face:b00c::25"}],
 "ReturnCode": "NOERROR"}
```

             An answer with several resource records

```
{"Zone": {"Origin": "isi.edu"},
 [
    {"Type": "SOA", "Name": "@",
     "MasterServerName": "venera",
     "MaintainerName": "action.domains.",
     "Serial": 20},
```

```
        {"Type": "NS", Name": "@",
         "Target": "a.isi.edu"},
        {"Type": "NS", Name": "@",
         "Target": "venera"},
        {"Type": "NS", Name": "@",
         "Target": "vaxa"},
        {"Type": "MX", Name": "@",
         "MailExchanger": "venera",
         "Preference": 10},
        {"Type": "MX", Name": "@",
         "MailExchanger": "vaxa",
         "Preference": 20},
        {"Type": "A", Name": "a",
         "Address": "26.3.0.103"},
        {"Type": "A", Name": "venera",
         "Address": "10.1.0.52"},
        {"Type": "A", Name": "venera",
         "Address": "128.9.0.32"}
    ]
  }
```

                    The zone file of RFC 1035

3.6. Open questions

   Would it be a good idea to document a formal way to derive member
   names for the resource record JSON objects? It would allow 1) to
   document the rationale for the current names 2) to automatically
   allow representation of new DNS resource records. A possible
   candidate for such derivation is [I-D.levine-dnsextlang].

   Should we define mandatory members for some objects, in order to have
   something the consumers can rely on? It seems there is a clear
   consensus to do so, making fields with non-default values mandatory.

   In resource records objects, members such as TTL are redundant (since
   they are actually RRset-wide). Should we have a new level of
   objects, for RRsets?

   Should we use JSON schema ([I-D.zyp-json-schema] and
   [I-D.fge-json-schema-validation]) to define the profile?

   Should we add a normative reference to every RFC describing one of
   the RR types used here or simply refer to the IANA registry?

   Should we have a way to represent unknown RR types, following
   [RFC3597]?

How binary data should be represented, for types like DNSKEY? Should
we use Base64 or is the key value an escaped binary string?

4. Security considerations

These JSON documents are not signed (see [I-D.barnes-jose-use-cases])
and therefore not authentified, even if the original data was secured
with DNSSEC. If transported over an insecure transport, they can be
read by a sniffer.

Also, see the security considerations of [RFC4627].

5. References

5.1. Normative References

[RFC1034]   Mockapetris, P., "Domain names - concepts and facilities",
            STD 13, RFC 1034, November 1987.

[RFC1035]   Mockapetris, P., "Domain names - implementation and
            specification", STD 13, RFC 1035, November 1987.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4627]   Crockford, D., "The application/json Media Type for
            JavaScript Object Notation (JSON)", RFC 4627, July 2006.

5.2. Informative References

[RFC2616]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
            Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
            Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3597]   Gustafsson, A., "Handling of Unknown DNS Resource Record
            (RR) Types", RFC 3597, September 2003.

[I-D.levine-dnsextlang]
            Levine, J. and P. Vixie, "An Extension Language for the
            DNS", draft-levine-dnsextlang-05 (work in progress),
            December 2012.

[I-D.barnes-jose-use-cases]
            Barnes, R., "Use Cases and Requirements for JSON Object
            Signing and Encryption (JOSE)", draft-barnes-jose-use-
            cases-01 (work in progress), October 2012.

[I-D.mohan-dns-query-xml]

              Parthasarathy, M. and P. Vixie, "Representing DNS messages
              using XML", draft-mohan-dns-query-xml-00 (work in
              progress), September 2011.

   [I-D.zyp-json-schema]
              Galiegue, F., Zyp, K., and G. Court, "JSON Schema: core
              definitions and terminology", draft-zyp-json-schema-04
              (work in progress), January 2013.

   [I-D.fge-json-schema-validation]
              Zyp, K. and G. Court, "JSON Schema: interactive and non
              interactive validation", draft-fge-json-schema-
              validation-00 (work in progress), January 2013.

   [getdns]   Hoffman, P., "Description of the getdns API", February
              2013.

   [dnspython]
              , "dnspython: A DNS toolkit for Python", February 2013.

Author's Address

   Stephane Bortzmeyer
   AFNIC
   Immeuble International
   Saint-Quentin-en-Yvelines 78181
   France

   Phone: +33 1 39 30 83 46
   Email: bortzmeyer+ietf@nic.fr
   URI: http://www.afnic.fr/