Network Working Group
Internet-Draft
Intended status: Experimental
Expires: March 25, 2017

P. Hoffman
ICANN

September 21, 2016

Representing DNS Messages in JSON
draft-hoffman-dns-in-json-08

Abstract

Some applications use DNS messages, or parts of DNS messages, as data. For example, a system that captures DNS queries and responses might want to be able to easily search those without having to decode the messages each time. Another example is a system that puts together DNS queries and responses from message parts. This document describes a standardized format for DNS message data in JSON.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The DNS message format is defined in [RFC1035]. DNS queries and DNS responses have exactly the same structure. Many of the field names and data type names given in [RFC1035] are commonly used in discussions of DNS. For example, it is common to hear things like "the query had a QNAME of 'example.com'" or "the RDATA has a simple structure".

There are hundreds of data interchange formats for serializing structured data. Currently, JSON [RFC7159] is quite popular for many types of data, particularly data that has named sub-fields and optional parts.

This document uses JSON to describe DNS messages. It also defines how to describe a paired DNS query and response, and how to stream DNS objects.

1.1. Design of the Format

There are many ways to design a data format. This document uses a specific design methodology based on the DNS format.

- o The format is based on JSON objects in order to allow a writer to include or exclude parts of the format at will. No object members are ever required.

- o This format is purposely overly-general. Protocols and applications that use this format are expected to use only a subset of the items defined here.

- o All members whose values that are always 16 bits or shorter (even booleans) are represented by JSON integers.

- o The encoding for the DNS object is ASCII as described in [RFC0020]. This is done to prevent an attempt to use a different encoding such as UTF-8 for octets in names or data.

- o Values for domain names and RDATA can be expressed using the decimal escaping ("\DDD") defined in [RFC1035], although this can lead to processing problems due to the escaping rules for JSON.

- o Names of items that have string values can have an "HEX" or "B64" appended to them to indicate a non-ASCII encoding of the value. Names that end in "HEX" have values stored in base16 encoding (hex with uppercase letters) defined in [RFC4648]. This is particularly useful for RDATA that is binary. Names that end in "B64" have values stored in base64url encoding defined in [RFC4648]. This is particularly useful for RDATA that very long (such as cryptographic keys) or entire records.

- o All field names used in [RFC1035] are used in this format as-is. Names not defined in [RFC1035] generally use "camel case".

- o The same data may be represented in multiple object members multiple times. For example, there is a member for the octets of the DNS message header, and there are members for each named part of the header. A message object can thus inadvertently have inconsistent data, such as a header member whose value does not match the value of the first bits in the entire message member.

- o The design explicitly allows for the description of malformed DNS messages. This is important for systems that are logging messages seen on the wire, particularly messages that might be used as part of an attack. A few examples of malformed DNS messages include:

   * an RR that has an RDLENGTH of 4 but an RDATA whose length is longer than 4 (if it is the last RR in a message)

   * a DNS message whose QDCOUNT is 0

   * a DNS message whose QDCOUNT is large but there are insufficient bytes after the header

   * a DNS message whose length is less than 12 octets, meaning it doesn't even have a full header

o An object in this format can have zero or more of the members defined here; that is, no members are required by the format itself. Instead, profiles that use this format might have requirements for mandatory members, optional members, and prohibited members from the format. Also, this format does not prohibit members that are not defined in this format; profiles of the format are free to add new members in the profile.

This document defines DNS messages, not zone files. A later specification could be written to extend it to represent zone files.

2. JSON Format for DNS Messages

The following gives all of the members defined for a DNS message. It is organized approximately by levels of the DNS message.

2.1. Message Object Members

o ID - Integer whose value is 0 to 65535

o QR - Integer whose value is 0 or 1

o Opcode - Integer whose value is 0 to 15

o AA - Integer whose value is 0 or 1

o TC - Integer whose value is 0 or 1

o RD - Integer whose value is 0 or 1

o RA - Integer whose value is 0 or 1

o AD - Integer whose value is 0 or 1

o CD - Integer whose value is 0 or 1

o RCODE - Integer whose value is 0 to 15

- o QDCOUNT - Integer whose value is 0 to 65535
- o ANCOUNT - Integer whose value is 0 to 65535
- o NSCOUNT - Integer whose value is 0 to 65535
- o ARCOUNT - Integer whose value is 0 to 65535
- o QNAME - String of the name of the first Question section of the message; see Section 2.5 for a desciption of the contents
- o compressedQNAME - Object that describes the name with two optional values: "isCompressed" (with a value of 0 for no and 1 for yes) and "length" (with an integer giving the length in the message)
- o QTYPE - Integer whose value is 0 to 65535, of the QTYPE of the first Question section of the message
- o QTYPEname - String whose value is from the IANA RR TYPEs registry, or that has the format in [RFC3597]; this is case-sensitive, so "AAAA" not "aaaa"
- o QCLASS - Integer whose value is 0 to 65535, of the QCLASS of the first Question section of the message
- o QCLASSname - String whose value is "IN", "CH", "HS", or has the format in [RFC3597]
- o questionRRs - Array of zero or more resource records or rrSet obects in the Question section
- o answerRRs - Array of zero or more resource records or rrSet obects in the Answer section
- o authorityRRs - Array of zero or more resource records or rrSet obects in the Authority section
- o additionalRRs - Array of zero or more resource records or rrSet obects in the Additional section

2.2. Resource Record Object Members

A resource record is represented as an object with the following members.

- o NAME - String of the NAME field of the resource record; see Section 2.5 for a description of the contents

- o compressedNAME - Object that describes the name with two optional values: "isCompressed" (with a value of 0 for no and 1 for yes) and "length" (with an integer giving the length in the message)

- o TYPE - Integer whose value is 0 to 65535

- o TYPEname - String whose value is from the IANA RR TYPEs registry, or that has the format in [RFC3597]; this is case-sensitive, so "AAAA" not "aaaa"

- o CLASS - Integer whose value is 0 to 65535

- o CLASSname - String whose value is "IN", "CH", "HS", or has the format in [RFC3597]

- o TTL - Integer whose value is 0 to 4294967295

- o RDLENGTH - Integer whose value is 0 to 65535. Applications using this format are unlikely to use this value directly, and instead calculate the value from the RDATA.

- o RDATA - String of the octets of the RDATA field of the resource record. This will most likely be given as RDATAHEX or RDATAB64, not as an ASCII string with \DDD encoding. (This format does not have a way of expressing RDATA by the fields for each DNS record type. Instead, it assumes that a processor of these records probably already knows how to split up an RDATA using master file format.)

- o rrSet - List of objects which have RDLENGTH and RDATA members.

A Question section can be expressed as a resource record. When doing so, the TTL, RDLENGTH, and RDATA members make no sense.

2.3. The Message and Its Parts as Octets

The following can be members of a message object. These names will most likely be given as "HEX" or "B64" to indicate encoding that is not plain ASCII. All these items are strings.

- o messageOctets - The octets of the message

- o headerOctets - The first 12 octets of the message (or fewer, if the message is truncated)

- o questionOctets - The octets of the Question section

- o answerOctets - The octets of the Answer section

- authorityOctets - The octets of the Authority section

- additionalOctets - The octets of the Additional section

The following can be a member of a resource record object.

- rrOctets - The octets of a particular resource record

The items in this section are useful in applications to canonically reproduce what appeared on the wire. For example, an application that is converting wire-format requests and responses might do decompression of names, but the system reading the converted data may want to be sure the decompression was done correctly. Such a system would need to see the part of the message where the decompressed labels resided, such as in one of the items in this section.

2.4. Additional Message Object Members

The following are members that might appear in a message object:

- dateString - The date that the message was sent or received, given as a string in the standard format described in [RFC3339], as refined by Section 3.3 of [RFC4287]

- dateSeconds - The date that the message was sent or received, given as the number of seconds since 1970-01-01T00:00Z in UTC time; this number can be fractional

- comment - An unstructured comment as a string

2.5. Name Fields

Names are represented by JSON strings. The rules for how names are encoded are described in Section 1.1. The contents of these fields are always uncompressed, that is after [RFC1035] name compression has been removed.

There are three encodings for names:

- If the member name does not end in "HEX" or "B64", the value is a domain name encoded as ASCII. Non-ASCII octets in the domain name can be expressed using the decimal escaping ("\DDD"). Periods indicate separation between labels.

- If the member name ends in "HEX", the value is the wire format for an entire domain name stored in base16 encoding.

o If the member name ends in "B64", the value is the wire format for an entire domain name stored in base64url encoding.

3. JSON Format for a Paired DNS Query and Response

A paired DNS query and response is represented as an object. Two optional members of this object are names "queryMessage" and "responseMessage", and each has a value that is an message object. This design was chosen (as compared to the more obvious array of two values) so that a paired DNS query and response could be differentiated from a stream of DNS messages whose length happens to be two.

4. Streaming DNS Objects

Streaming DNS objects is performed using [RFC7464].

5. Examples

5.1. Example of the Format of a DNS Query

The following is an example of a query for the A record of 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
}
```

As stated earlier, all members of an object are optional. This example object could have one or more of the following members as well:

```
"answerRRs": []
"authorityOctetsHEX": ""
"comment": "Something pithy goes here"
"dateSeconds": 1408504748.657783
"headerOctetsHEX": "4CDE00000001000000000000"
"QNAMEHEX": "076578616D706C6503636F6D00",
"compressedQNAME": { "isCompressed": 0 },
"messageOctetsHEX":
     "4CDE00000001000000000000076578616D706C6503636F6D0000010001"
"messageOctetsB64": "TN4AAAABAAAAAAAAB2V4YW1wbGUDY29tAAABAAE="
"questionOctetsHEX": "076578616D706C6503636F6D0000010001"
"questionRRs": [ { "NAMEHEX": "076578616D706C6503636F6D00",
               "TYPE": 1, "CLASS": 1, "hostNAME" : "example.com." } ]
"questionRRs": [ { "NAME": "example.com.", "TYPE": 1,
               "CLASS": 1, } ]
```

5.2. Example of the Format of a Paired DNS Query and Response

The following is a paired DNS query and response for a query for the A record of example.com.

```
{
  "queryMessage": { "ID": 32784, "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 },
  "responseMessage": { "ID": 32784, "QR": 1, "AA": 1, "RCODE": 0,
                      "QDCOUNT": 1, "ANCOUNT": 1, "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" } ]
                    }
}
```

The Answer section could instead be given with an rrSet:

```
   "answerRRs": [ { "NAME": "example.com.",
                    "TYPE": 1, "CLASS": 1,
                    "TTL": 3600,
                    "rrSet": [ { "RDATAHEX": "C0000201" },
                               { "RDATAHEX": "C000AA01" } ] ],
```

6. Local Format Policy

Systems using this format in this document will likely have policy about what must be in the objects. Those policies are outside the scope of this document.

For example, private DNS systems such as those described in [I-D.dulaunoy-dnsop-passive-dns-cof] covers just DNS responses. Such a system might have a policy that makes QNAME, QTYPE, and answerRRs mandatory. That document also describes two mandatory times that are not in this format, so the policy would possibly also define those members and make them mandatory. The policy could also define additional members that might appear in a record.

As another example, a program that uses this format for configuring what a test client sends on the wire might have a policy of "each record object can have as few members as it wants; all unstated members are filled in from previous records".

7. IANA Considerations

7.1. MIME Type Registration of application/dns+json

To: ietf-types@iana.org
Subject: Registration of MIME media type application/dns+json

MIME media type name: application

MIME subtype name: dns+json

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type.

Security considerations: This document specifies the security considerations for the format.

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: This document.

Applications that use this media type: Systems that want to exchange DNS messages.

Additional information:

Magic number(s): n/a

File extension(s): This document uses the mime type to refer to protocol messages and thus does not require a file extension.

Macintosh file type code(s): n/a

Person & email address to contact for further information:
Paul Hoffman, paul.hoffman@icann.org

Intended usage: COMMON

Restrictions on usage: n/a

Author: Paul Hoffman, paul.hoffman@icann.org

Change controller: Paul Hoffman, paul.hoffman@icann.org

8. Security Considerations

As described in Section 1.1, a message object can have inconsistent data, such as a message with an ANCOUNT of 1 but that has either an empty answerRRs array or an answerRRs array that has 2 or more RRs. Other examples of inconsistent data would be resource records whose RDLENGTH does not match the length of the decoded value in the RDATAHEX member, or a record whose various header fields do not match the value in headerOctetsHEX, and so on. A reader of this format must never assume that all of the data in an object are all consistent with each other.

Numbers in JSON do not have any bounds checking. Thus, integer values in a record might have invalid values, such as an ID value that is negative, or greater than or equal to 2^16, or has a fractional part.

DNS has its own escaping mechanism for non-ASCII octets using "\DDD". Using DNS escaping in this JSON format could lead to security issues if the receiver does not correctly handle the double-escaping.

9. Acknowledgements

Some of the ideas in this document were inspired by earlier, abandoned work such as ([I-D.daley-dnsxml], [I-D.mohan-dns-query-xml], and [I-D.dulaunoy-dnsop-passive-dns-cof]. The document was also inspired by early ideas from Stephane Bortzmeyer. Many people on the DNSOP WG mailing list contributed very useful ideas (even though this was not a WG work item.)

10. References

10.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <http://www.rfc-editor.org/info/rfc3597>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7464] Williams, N., "JavaScript Object Notation (JSON) Text Sequences", RFC 7464, DOI 10.17487/RFC7464, February 2015, <http://www.rfc-editor.org/info/rfc7464>.

10.2. Informative References

[I-D.daley-dnsxml]
Daley, J., Morris, S., and J. Dickinson, "dnsxml - A standard XML representation of DNS data", draft-daley-dnsxml-00 (work in progress), July 2013.

[I-D.dulaunoy-dnsop-passive-dns-cof]
Dulaunoy, A., Kaplan, A., Vixie, P., and H. Stern, "Passive DNS - Common Output Format", draft-dulaunoy-dnsop-passive-dns-cof-01 (work in progress), November 2015.

[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.

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <http://www.rfc-editor.org/info/rfc20>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, DOI 10.17487/RFC4287, December 2005, <http://www.rfc-editor.org/info/rfc4287>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

Author's Address

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org