Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 02, 2013

J. Levine
Taughannock Networks
P. Vixie
Internet Systems Consortium

December 2012

An Extension Language for the DNS
draft-levine-dnsextlang-05

Abstract

Adding new RRTYPEs to the DNS requires that DNS servers and provisioning software be upgraded to support each new RRTYPE in Master files. This document defines a DNS extension language intended to allow most new RRTYPEs to be supported by adding entries to configuration data read by the DNS software, with no software changes needed for each RRTYPE.

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 June 02, 2013.

Copyright Notice

Table of Contents

1. Introduction

The Domain Name System[RFC1034] [RFC1035] is designed to be extensible, with new record types, known as RRTYPEs, added as needed. While it is straightforward in principle to add a new RRTYPE, in practice it can be difficult due to the software changes needed to add the new RRTYPE to the master file format read by many authoritative DNS servers, and to the provisioning software used to create and update the master files or the local equivalent.

While some new RRTYPEs, notably those for DNSSEC [RFC4033], require that DNS servers do new special purpose processing, most new RRTYPEs are, from the point of view of the DNS, just static data to return to

queries, perhaps with some additional section records if the record includes another domain name. This document defines an extension language to describe these RRTYPEs, so that server and provisioning software can parse master file records for the RRTYPEs.

2. Typical usage

The extension language is written as strings of ASCII text that describe new RR types, intended to be stored in the DNS itself. (They may also be stored in a local file with a well-known name, for debugging and local overrides, but this usage is optional.) All of the DNS software that needs to handle master file records fetches records from the DNS as needed. To support a new RRTYPE, one would add suitable records to the DNS zone where the descriptions are located, or to the local file.

DNS servers can use the extension language to parse new RRTYPE records in master files, and to translate them to the binary representation. Servers that create ASCII master files from zone data retrieved via AXFR can use the extension language to create master file records for new RRTYPEs.

Provisioning software can use the extension language to create templates for users to fill in, to create new RRTYPE records in master files to be passed to DNS servers, and to syntax check records entered by users.

In principle, provisioning software could create TYPEnn master records if the local DNS server doesn't implement the extension language, although it would be less confusing if both provisioning and server software both accept the same master record syntax.

Some DNS servers store records in ways other than master files, such as SQL databases. In principle, the extension language could be used to create new schema entries to handle new RRTYPEs, although the details are too specific to particular varieties of DNS server

software for this document to try to describe the details.

3. Extension language syntax

3.1. Lexical structure

The extension language consists of "stanzas", each of which defines an RRTYPE. In the DNS, a stanza is stored as a multi-string TXT record, with each string conceptually being a line in the stanza. In a file, it is stored as a series of lines. The first line of a stanza defines the symbolic RRTYPE name. Subsequent lines each define a field in the record.

The following ABNF imports ALPHA, DIGIT, and WSP from [RFC5234].


```
dnsextfile =  1*stanza

stanza = rrtypeline 1*fieldline

rrtypeline = 1*ALPHA ":" 1*DIGIT 0*1(WSP freetext)

fieldline = WSP ftype 0*1qualifiers 0*1(WSP freetext)

ftype = "I1" | "I2" | "I4" | "A" | "AAAA" | "N" | "S" | "B" | "X"

qualifiers = "[" qual 0*(, qual) "]"

qual = 1*ALPHA "=" 1*DIGIT | "C" | "A" | "L" | "M"

freetext = 0*(%x20-%xfe)
```

3.2. Storage in the DNS

Each extension language stanza stored in the DNS is stored as two

identical TXT records, one with a name based on the numeric RR type, one with a name based on the text name. (One record MAY be aliased to the other using a CNAME.) The numeric names are located at RRTYPE.ARPA, and the text names are located at RRNAME.ARPA. For example, if the FOO record type were type 999, the two records would be:

```
999.RRTYPE.ARPA TXT "FOO:999 Foo record" "..."
FOO.RRNAME.ARPA TXT "FOO:999 Foo record" "..."
```

3.3. Storage in a file

All the extension language stanzas stored in a file are stored as lines of ASCII text. The name of the RR type starts in the first position of the first line in the stanza. Subsequent lines in the stanza start with white space. A line that is blank or starts with a # character is a comment and is ignored.

3.4. Stanza structure

Each stanza starts with a line containing the name of the RRTYPE, a colon, and the numeric RRTYPE. The name of the RRTYPE must start in the first position on the line. When stored in a file, the RRTYPE name should not be the same as an existing RRTYPE or DNS class name (IN or CH) or bad things will happen. The RRTYPE may be followed by white space and a descriptive comment intended to be displayed to human users, but not interpreted by DNS software. Provisioning software might use the comments as prompts or labels to help a user

select the desired RRTYPE.

The rest of the lines in the stanza start with white space and describe the fields in the record. Each field is one or more octets long, and fields are stored sequentially in the record:

```
FOO:999 Foo record
   field description
   field[qual,qual] description
   field
```

...

Some fields may be followed by a comma-separated list of qualifiers in square brackets. The qualifiers further define the field, e.g., in a numeric field, the qualifiers may define symbolic names for field values or bit masks. The field and optional qualifiers may be followed by white space and a description of the field. The description is intended to be displayed to human users, and is not interpreted by DNS software. Provisioning software might use the comments as prompts or labels for templates into which users type RR data.

3.5. Field types

Each field type is defined by a token name consisting of letters and digits, starting with a letter.

3.5.1. Integer fields

Integer fields are defined by I1, I2, and I4 tokens, for fields one, two, or four octets long. The corresponding value in a master record is an unsigned integer number. A field may be followed by qualifiers defining symbolic field values.

A symbolic field value is represented as NAME=NN where NAME is the symbol and NN is the numeric value to be placed in the field. The corresponding value in a master record is the symbol. The symbol can contain any ASCII printing character other than comma, equal sign, vertical bar, angle braces, or backslash. For example, to define the type field in a CERT record [RFC4398]:

```
I2[PKIX=1,SPKI=2,PGP=2,IPKIX=4,ISPKI=5,IPGP=6,ACPKIX=7,\
 IACPKIX=8,URI=253,OID=254] Type
```

3.5.2. IP address fields

IP address fields are defined by A or AAAA tokens, for four-octet

IPv4 addresses or 16-octet IPv6 addresses. The corresponding value in a master record is an IP address written in the usual way. There are no qualifiers.

3.5.3. Domain name fields

Domain name fields are defined by N tokens. The qualifier C means the name is compressed. The qualifier A means that the DNS server should do the usual additional record processing, including related A and AAAA records if available. The qualifier M means the name is really an e-mail address, i.e., the first component is the mailbox and the rest is the actual domain name. Multiple qualifiers are permitted, e.g. N[A,C] for a compressed name with additional record processing.

The corresponding value in a master record is a domain name, written in the usual way, with \. meaning a literal dot in a record.

Names are absolute if they end with a dot, otherwise relative to $ORIGIN, the existing convention for master files.

3.5.4. String fields

String fields are defined by S tokens. The qualifier L means that the string may be long, more than 255 bytes, in which case it is stored in the record as multiple strings, with the location of the boundary between the strings undefined. The qualifier M means that there may be multiple strings, each stored as a string in the record. A string field with either qualifier must be the last field in the record.

The corresponding value in a master record is a string enclosed in single or double quotes, or multiple strings if the M qualifier is present. Embedded quotes may be escaped with a backslash, and a double backslash represents a backslash. If a non-null string contains no white space, quote characters, or backslashes, the quotes may be omitted.

3.5.5. Base-64 fields

A base64 field is defined by a B token. The qualifier C means that the field is stored in the record as a string with a preceding length byte. A base64 field without a C qualifier must be the last field in the record.

The corresponding value in a master record is a string represented as base64 [RFC3548]. The value of a base64 field without a C qualifier may include embedded spaces for readability, which are ignored.

3.5.6. Hex fields

A hex field is defined by an X token. There are no qualifiers. A hex field must be the last field in the record.

The corresponding value in a master record is a string represented as an even number of hexadecimal digits. The value may include embedded spaces for readability, which are ignored.

4. Examples

If a DNS server didn't already have support for MX records, they could be defined as:

```
MX:15 Mail exchanger
  I2 Priority (lower values are higher priority)
  N[A,C] Host name
```

The name is MX, the RRTYPE is 15, and the data includes a two-octet number and a compressed domain name, with additional section records for the domain name.

The SRV record [RFC2782] could be defined as:

```
SRV:33 Service location
  I2 Priority
  I2 Weight
  I2 Port
  N[A] Target host name
```

The name is SRV, the RRTYPE is 33. The record contains three two-octet fields for the priority, weight, and port, and a domain name. The domain name is not compressed, but the DNS server should include additional section records for it.

5. Security considerations

The extension language makes it possible to create master files that represent arbitrary DNS records. Since most DNS servers already provide ways to represent arbitrary data, this doesn't introduce any new security issues to the DNS and DNS servers, although it may create security issues in provisioning software if the provisioning system is intended to limit the kinds of records its users can define.

Extension language files with accidentally or deliberately invalid field definitions could provoke odd bugs in server or provisioning software that doesn't check the syntax before using it.

6. IANA considerations

This document requests that IANA create the RRTYPE.ARPA and RRNAME.ARPA zones. Their initial contents are as follows: [list of description of existing RRs here]

When new RR types are defined, the defining documents SHOULD request IANA to add appropriate records to RRTYPE.ARPA and RRNAME.ARPA.

This document requests that IANA create a registry of DNS Extension Language Field Types. Its initial contents are as follows

TYPE	REFERENCE
I1	(this document)
I2	(this document)
I4	(this document)
A	(this document)
AAAA	(this document)
N	(this document)
S	(this document)
B	(this document)
X	(this document)

Table 1: DNS Extension Language Field Types Registry Initial Values

7. References

7.1. References - Normative

[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.

[RFC3548] Josefsson, S., "The Base16, Base32, and Base64 Data

Encodings", RFC 3548, July 2003.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

7.2. References - Informative

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4398] Josefsson, S., "Storing Certificates in the Domain Name System (DNS)", RFC 4398, March 2006.

Appendix A. Change Log

NOTE TO RFC EDITOR: This section may be removed upon publication of this document as an RFC.

A.1. Changes from -04 to -05

DNS publication in RRYPE.ARPA and RRNAME.ARPA.

A.2. Changes from -03 to -04

More use cases.

Fix up BNF

A.3. Changes from -02 to -03

First stab at BNF

Note $ORIGIN matters

A.4. Changes from -01 to -02

Editorial nits

A.5. Changes from -00 to -01

Switch to multi-line format. Add comments for provisioning.

Authors' Addresses

John Levine
Taughannock Networks
PO Box 727
Trumansburg, NY 14886

Phone: +1 831 480 2300
Email: standards@taugh.com
URI: http://jl.ly

Paul Vixie
Internet Systems Consortium
950 Charter Street
Redwood City, CA>

Email: vixie@isc.org