# DNS 実装者から見た ZoneFile

2026年06月26日

株式会社インターネットイニシアティブ
技術研究所
日比野 啓

© Internet Initiative Japan Inc.

# 自己紹介

- 日比野 啓
- 所属: IIJ 技術研究所

- DNS の研究開発として、フルリゾルバ bowline を Haskell で実装
  - DNS の最新動向への追随と実証実験環境の整備
  - DNS 関連ソフトウェアの整理と部品化
  - 社内の要望に応じたカスタマイズ対応

# ZoneFile の解釈

本日は ZoneFile (Master File) の解釈の話をします。
解釈器を実装していて、興味深かった内容を紹介します。

ZoneFile

- Resource Record (RR) の定義を並べることでゾーンを定義する

```
$ORIGIN                                  example.jp.

example.jp.        3600     IN     SOA          ...

foo.bar     3600 IN   A    198.51.100.13

baz         IN   600    A    198.51.100.14

example.jp.   3600 IN   A    198.51.100.12
                         AAAA  2001:db8::12

a\.b\046c          TXT   "a.b.c" “d e”
```

# Resoure Record のリストによるゾーンの定義

example.jp.  3600 IN  A  198.51.100.12

ZoneFile

- Resource Record (RR) の定義を並べることでゾーンを定義する

<entry> := <domain-name> <rr>

<rr> :=  <TTL>  <class>  <type> <RDATA>

ドメイン名の後ろに RR の定義内容を記述

# ドメイン名の相対名

- $ORIGIN でゾーンの頂点を指定
- 最後が '.' でない名前はゾーンの頂点からの相対名
  - e.g. foo.bar.example.jp.

```
$ORIGIN                    example.jp.

foo.bar     3600 IN   A     198.51.100.13
```

<entry> := <domain-name> <rr>

<rr> := <TTL> <class> <type> <RDATA>

# 逆順でもよいフィールド

TTL と class(IN) は逆順でも良い

```
<entry> := <domain-name> <rr>
           | <rr>
<rr> :=  <TTL>   <class>  <type> <RDATA>
       |  <class>   <TTL>  <type> <RDATA>
```

```
foo.bar     3600 IN    A     198.51.100.13

baz         IN     600  A     198.51.100.14
```

# RR の省略可能な記述 (1)

- ドメイン名は省略可能
- TTL と class (IN) も、それぞれ省略可能

```
<entry> := <domain-name> <rr>
         | <rr>
<rr> := [<TTL>] [<class>] <type> <RDATA>
```

```
example.jp.  3600 IN   A       198.51.100.12
                       AAAA    2001:db8::12
```

# RR の省略可能な記述 (2)

```
example.jp.  3600 IN   A       198.51.100.12
                       AAAA    2001:db8::12
```

<entry> := <domain-name> <rr>
| <rr>
<rr> := [<TTL>] [<class>] <type> <RDATA>

ドメイン名 / class を省略した場合

- 前の行の値を引き継ぐ

TTL を省略した場合

- $TTL より前
  - 前の行の値を引き継ぐ - RFC 1035
- $TTL より後
  - $TTL で指定されたデフォルトの TTL - RFC 2308

# ( ... ) の扱い (1)

SOA でよく見る ( ... )

- RR の定義は、基本的には一行に書く
- ( … ) で囲うと、改行をまたいで RR の記述(論理行)を延長できる

```
example.jp. 3600 IN SOA ns.example.jp. admin.example.jp. (
                        1
                        3600
                        1800
                        3600000
                        3600
                    )
```

# ( ... ) の扱い (2)

RR の論理行を延長するだけなので、同じ RR 内で、何度も使える

```
example.jp.  3600  IN  SOA  ns.example.jp. admin.example.jp. (
                           1
                           3600
                        )  1800    (
                           3600000
                        )  3600    (
                        )
```

# ZoneFile 内のデータ表現

a\.b\046c　　TXT　"a.b.c" “d e”

任意のバイト(8bit)を通常のデータとして使用できる

- '\<文字>'　- エスケープ
- '\DDD' - 10進3桁 文字コード
- " " でくくられた文字列

表示できない文字や、バイナリも利用できる

# ドメイン名の表現

任意のバイトをドメイン名としても使用できる

- ドメイン名は '.' で区切られたラベルのリスト。
  最後の '.' は root。
- 最後に '.' が無いときは、ゾーンの頂点からの相対名
- ホスト名の規則に違反している文字列も使用可能

'\.' 等と書けば、ラベル内に '.' を利用することもできる。

```
example.jp.   3600 IN   A      198.51.100.12

a\.b\046c               TXT    "a.b.c" “d e“
```

# ZoneFile の字句解析

- 前処理で ( … ) を処理、論理行のリストを得る
- エスケープされていない '.' は、ラベルの区切りを意味する字句にする
- エスケープされている '.' は、通常のデータとする
- " " で括られた文字列は、文字列データ (character-string) となる

```
foo.bar\.baz  TXT  ( "foo"
                     "bar.baz" )
{- 字句の列へ -}
⇒ 'foo', '.', 'bar.baz', ' ', TXT,
'(', ' ', "foo", \n, ' ', "bar.baz", ' ', ')', \n
{- ( ... ) の前処理 -}
⇒ 'foo', '.', 'bar.baz', ' ', TXT, ' ', "foo", ' ', "bar.baz", ' ', \n
```

# ZoneFile の状態付き解釈

状態付き解釈器で以前の行の最後の値を覚える

- domain-name, TTL, class

```
$ORIGIN  example.jp.
  {- domain-name=example.jp.        ttl=<default>   zone=example.jp. -}
example.jp.   3600 IN  SOA   ...
  {- domain-name=example.jp.        ttl=3600 -}
foo.bar       3600 IN   A     198.51.100.13
  {- domain-name=foo.bar.example.jp.  ttl=3600 -}
baz           IN   600  A     198.51.100.14
  {- domain-name=baz.example.jp.      ttl=600 -}
example.jp.   3600 IN   A     198.51.100.12
  {- domain-name=example.jp.        ttl=3600 -}
                    AAAA  2001:db8::12
  {- domain-name=example.jp.        ttl=3600 -}
```

# ゾーン頂点の指定と参照

$ORIGIN で指定して、@ で参照する

$ORIGIN の指定が無いとき

- ゾーン頂点を外側から与える必要がある
- ZoneFile による表現が、
  ゾーン頂点を受けとってゾーンを返す関数 となる

```
$ORIGIN  example.jp.

@   3600  IN  SOA   ns.example.jp. admin.example.jp. ( ... )
```

# SVCB / HTTPS レコード

SVCB レコードの例

- DDR (Discovery of Designated Resolvers - RFC 9462) での利用
  - 暗号化DNSの情報を記述

```
_dns.resolver.arpa. 300 IN SVCB 1 c.example.
  alpn=h2,h3 port=443 ipv4hint=192.0.2.19 ipv6hint=2001:db8::13,2001:db8::14 dohpath="/dns-query{?dns}"
```

# SVCB / HTTPS レコード (リストの値の解釈 )

character-string への変換後に解釈する

- 変換後のデータさらに ',' で区切る
- ',' は '\' でエスケープできる

svc.example.jp. 300 HTTPS 1 ... key667=part1\,\p\a\r\t2\044part3\092,part4\092\\

part1\,\p\a\r\t2\044part3\092,part4\092\\

{- charaster-string 化 -}

⇒ part1,part2,part3\,part4\\

{- charaster-string をさらにエスケープされていない ',' で区切る -}

⇒ 'part1', 'part2', 'part3\,part4\\'

# 参考資料

- RFC 1035 - MASTER FILES
  - https://datatracker.ietf.org/doc/html/rfc1035#section-5
- RFC 2308 - SOA Minimum Field
  - https://datatracker.ietf.org/doc/html/rfc2308#section-4
- RFC 9460 - A.1. Decoding a Comma-Separated List
  - https://datatracker.ietf.org/doc/html/rfc9460#name-decoding-a-comma-separated-

フルリゾルバ bowline

- Project Page https://iijlab.github.io/dnsext/
- GitHub https://github.com/iijlab/dnsext

# 付録: Monad による、値を引き継ぐ状態付き　Parser

```
type Parser a = StateT Context ( StateT [Token] (Except (Last Error)) )

`-- 部分的な Parser への参照が、それぞれ、再起降下　parser へ展開される
record :: Parser Record
record =
   R_RR      <$> zoneRR      <|>
   R_Origin  <$> zoneOrigin  <|>
   R_TTL     <$> zoneTTL

-- Context を使って、ゾーン頂点や前の行の値を引き継ぐ
data Context =
  Context {
    cx_zone   :: Domain
    cx_domain :: Domain
    cx_class  :: CLASS
    cx_ttl    :: CtxTTL
  }
```

ありがとうございました