インターネットのツボを押さえる!

DNSの仕組み完全解説

2

馬場達也

ゾーンデータファイルの内容と記述法

前回は、DNSの役割と仕組み、DNSを構築するためのソフトウェアについて紹介した。第2回目となる今回は、権威ネームサーバの持つゾーンデータファイルの内容と、その記述法について紹介しよう。

ゾーンデータファイルの書式はRFC 1035で規定されている

権威ネームサーバが持つゾーンデータファイルの書式は、RFC 1035によって決められており、BINDやWindows2000 Serverに付属しているMicrosoft DNS Serverもこの形式のゾーンデータファイルを使用している。

このゾーンデータファイルは、「$」で始まる制御ステートメントとリソースレコード（RR：Resource Record）で構成される。また、「;」（セミコロン）を付ければ、行末までコメントを挿入することもできる。制御ステートメントには、「$ORIGIN」「$INCLUDE」「$TTL」があり、それぞれ次のような書式になっている。

$ORIGIN <domain-name>

ゾーンデータファイル中でホストのFQDN（Fully Qualified Domain Name）を記述する場合は、FQDNの最後に「.」を付ける決まりがある。最後に「.」が付かない場合は、$ORIGINステートメントの<domain-name>で指定したドメイン名（これを「起点名」という）が末尾に付加され、FQDNとされる。この$ORIGINステートメントで指定した起点名は、次の$ORIGINステートメントまでの間に存在するリソースレコードに対して適用される。ただし、通常は暗黙的にそのゾーンのドメイン名が起点名としてセットされているので、ゾーンデータファイルの最初に$ORIGINステートメントを記述する必要はない。

$INCLUDE <file-name>

$INCLUDEステートメントを使用すると、その個所に<file-name>で指定したファイルの内容を挿入できる。

$TTL <ttl>

$TTLステートメントの<ttl>には、リソースレコードのデフォルトのキャッシュの有効期間（TTL：Time to Live）を記述する。リソースレコード内でTTLを指定しなかった場合は、このデフォルトのTTLが自動的にセットされるため、ゾーンデータファイルの最初に$TTLステートメントを記述するようにする。このTTLは、秒単位で記述する。$TTLステートメントで指定したデフォルトのTTLは、次の$TTLステートメントまでの間に存在するリソースレコードに対して適用される。$TTLステートメントはRFC 2308で導入された仕組みで、BINDでは、バージョン8.2以降に導入されている。

リソースレコードにはホストのアドレスなどが記述される

ゾーンデータファイルに記述される代表的なリソースレコードを表1に示す。このうち、主に利用されるのは、SOA、NS、A、CNAME、PTR、MXの各リソースレコードであろう。現時点で存在するリソースレコードの完全なリストは、IANA（Internet Assigned Numbers Authority）のWebページ（http://www.iana.org/assignments/dns-parameters）で確認できる。

ゾーンデータファイル中のリソースレコードは、基本的に次のような形式で記述する。

```
<owner> <ttl> <class> <type> <rdata>
```

<owner>には、このリソースレコードに関するホスト名などを記述する。行の先頭にタブやスペースなどが入ると、<owner>が省略されたとみなされ、直前のリソースレコードで記述した<owner>が自動的にセットされる。<ttl>には、このリソースレコードがローカルネームサーバなどでキャッシュされた場合の有効期間（TTL）を秒単位で記述する。<ttl>を省略した場合には、$TTLステートメントで指定したデ

フォルトのTTLがセットされる。<class>には、ネットワーククラスを記述する。インターネットでは「IN」と記述し、実際にはこれ以外には使用されていない。<class>を省略した場合には、自動的に「IN」がセットされる。<type>には、表1で示したようなリソースレコードのタイプを記述する。<rdata>の書式は、リソースレコードのタイプによって異なるので、あとで順次説明していく。

基本的に1つのリソースレコードは1行で記述するが、途中で改行したい場合は、1行目の行末に「(」を挿入して改行し、リソースレコードの最後に「)」を付けるようにすれば、複数行にわたって記述することも可能である。

それでは、代表的なリソースレコードの書式について紹介していこう。PTRレコードに関しては、ここでは説明せずに、次回詳しく取り上げる予定である。

タイプ	内容
SOA	権威の開始
NS	ドメインを管理するネームサーバ
A	ホストのIPv4アドレス
CNAME	別名（エイリアス）
PTR	ホスト名へのポインタ（逆引き用）
MX	メールの送付先となるメールサーバ
HINFO	ホストの情報
TXT	任意のテキスト
SRV	指定したサービスを提供しているサーバ
AAAA	ホストのIPv6アドレス※1
A6	ホストのIPv6アドレス※1
DNAME	IPv6アドレス逆引き用
SIG	署名（DNSSEC※2で使用）
KEY	公開鍵（DNSSEC※2で使用）
NXT	次ドメイン名（DNSSECで使用）

表1● 代表的なリソースレコードのタイプ

※1 AAAAを使用するかA6を使用するかは現在も議論が続いている
※2 DNSSEC：DNS Security Extensions

■SOAレコード

SOA（Start of Authority：権威開始）レコードは、ゾーンデータの最初に1つだけ記述する。SOAレコードの書式は例1のようになっている。

<owner>には、このゾーンのドメイン名を記述する。通常は「@」と記述し、この場合には、起点名がセットされる。<source-dname>には、このゾーンに関して権威を持っているプライマリマスタサーバのホスト名を記述する。<mbox>には、このゾーンの管理者のメールアドレスを記述する。ここではメールアドレス中の「@」を「.」に変更して記述する。また、「@」の前に「.」が存在する場合は、この「.」を「¥.」に変更する。つまり、「hostmaster@example.com」は、「hostmaster.example.com.」となり、「tatsuya.baba@example .com」は、「tatsuya¥.baba.example.com.」と記述する。

<serial>には、このゾーンデータのシリアル番号を記述する。このシリアル番号は、セカンダリマスタサーバとの間でゾーン転送を行う場合に、ゾーンデータの内容が更新されているのかどうかをセカンダリマスタサーバが判断するために使用する。つまり、ゾーンデータの内容を更新した場合には、このシリアル番号を増やさないと、セカンダリマスタサーバは、新しいゾーンデータを取得しようとしないので注意する必要がある。このシリアル番号は、「YYYYMMDDNN」（YYYY=年、MM=月、DD=日、NN=その日のリビジョン番号）の形式で記述することが推奨されている。

<refresh>には、プライマリマスタサーバのゾーンデータのシリアル番号をセカンダリマスタサーバがチェックする間隔（リフレッシュ間隔）を秒単位で指定する。ここでシリアル番号が上がっていた場合は、ゾーン転送が行われる。<retry>には、セカンダリマスタサーバが何らかの原因でゾーンデータのチェックに失敗した場合のリトライ間隔を秒単位で指定する。<expire>には、ゾーンデータの有効期間を秒単位で指定する。セカンダリマスタが、プライマリマスタにこの期間アクセスできない場合は、ゾーンデータを無効にしなければならない。そして、<minimum>には、RFC 2308に記述されているネガティブキャッシュの有効期間を秒単位で指定する。以前は<minimum>にリソースレコードのデフォルトの有効期間を設定するようになっていたが、$TTLステートメントで記述するように変更された。

```
<owner> <ttl> <class> SOA <source-dname> <mbox> (
              <serial> <refresh> <retry> <expire> <minimum> )
```

例1● SOAレコードの書式

■Aレコード

A（Address）レコードは、ホスト名に対応するIPアドレスを記述するためのリソースレコードである。Aレコードの書式は以下のようになっている。

```
<owner> <ttl> <class> A <address>
```

<owner>には、ホスト名（正規名）を記述し、<address>には、そのホストのIPv4アドレスを記述する。同じ<owner>に対して、IPアドレスの異なるAレコードを複数記述することも可能である。この場合は、

問い合わせ元に複数のIPアドレスが返されることになり、サーバの負荷分散の目的で使用することができる。

■CNAMEレコード

CNAME（Canonical Name：正規名）レコードは、ホスト名の別名（エイリアス）を記述するためのリソースレコードである。CNAMEレコードの書式は以下のようになっている。

```
<owner> <ttl> <class> CNAME
<canonical-name>
```

<owner>には別名を記述し、<canonical-name>には、ホストの正規名を記述する。<canonical-name>として記述したホストが、そのゾーンに属しているのであれば、その正規名に対するAレコードも下記のように同じゾーンデータファイル内に記述する必要がある。

```
www IN CNAME  foo
foo IN A  192.168.0.30
```

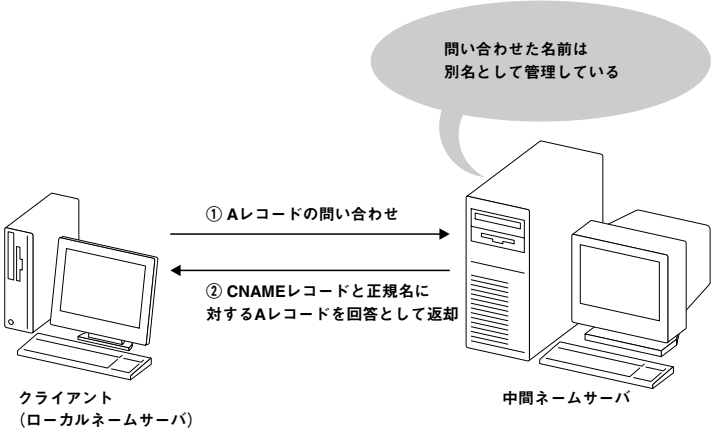

図1● 別名の名前解決に対して、CNAMEレコードと正規名に関するAレコードの内容を回答する

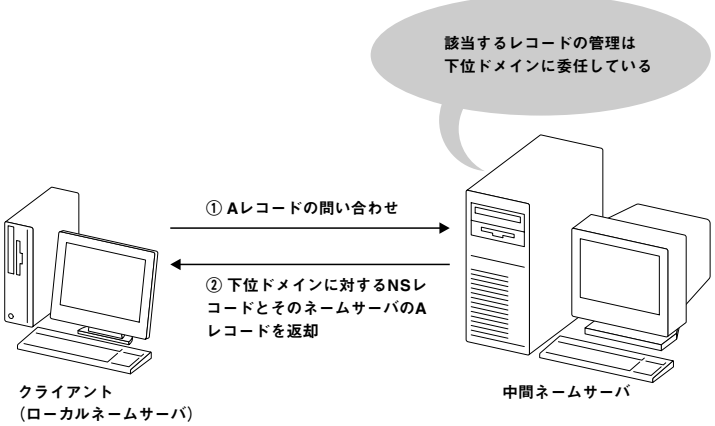

図2● 該当するリソースレコードの管理を下位ドメインに委任している場合の動作

この場合、別名に対して名前解決の問い合わせ（Aレコードの問い合わせ）があると、該当するCNAMEレコードと正規名に対するAレコードの両方が返される（図1）。

■NSレコード

NS（Name Server）レコードは、そのドメインに関して権威のあるネームサーバを記述するためのリソースレコードである。サブドメインについてNSレコードを記述した場合は、サブドメインを管理する権限を別のネームサーバに委任したことを示す。NSレコードの書式は以下のようになっている。

```
<owner> <ttl> <class> NS
<name-server-dname>
```

<owner>にはドメイン名を記述し、<name-server-dname>には、<owner>で記述されたドメインを管理する権限を与えたネームサーバのホスト名（正規名）を記述する。通常は、あるゾーンを管理するネームサーバは複数存在するため、同じ<owner>に対して複数のNSレコードが存在することになる。

下位ドメインの管理をNSレコードによってほかのネームサーバに委任した場合は、そのネームサーバのAレコードも記述しなければならない。このAレコードは、グルーレコード（glue record）と言われる。下位ドメインで管理しているリソースレコードへの問い合わせがあった場合は、その下位ドメインに関するすべてのNSレコードと、そのネームサーバに関するグルーレコード（Aレコード）を返却して、そちらに問い合わせるように指示する（図2）。

```
foo.example.com.    IN  NS  ns.foo.
example.com.
ns.foo.example.com. IN  A  192.
168.20.10 ←グルーレコード
```

■MXレコード

MX（Mail Exchanger）レコードは、メールの配送先となるメールサーバを記述するためのリソースレコードである。MXレコードの書式は次のようになっている。

```
<owner> <ttl> <class> MX
<preference> <exchange-dname>
```

<owner>にはドメイン名を記述し、<preference>にはメールサーバの優先度（数字が小さいほうが優先

される）を、<exchange-dname>にはメールサーバのホスト名（正規名）を記述する。

■HINFOレコード

HINFO（Host Information）レコードは、ホストのCPUおよびOSについての情報を記述するためのリソースレコードである。HINFOレコードの書式は次のようになっている。

```
<owner> <ttl> <class> HINFO
<cpu> <os>
```

<owner>にはホストの正規名を記述する。また、<cpu>にはホストで利用されているCPU名（実際はマシンの種類）を、<os>にはホストで利用されているOS名をそれぞれ記述する。HINFOレコードは、当初は、アプリケーションが相手ホストのハードウェアやOSの種類を判別するために用意されたが、現在では、単にホストの管理のために利用されている。しかし、ホストの情報をDNSによって公開するのはセキュリティ上問題があるため、HINFOレコードの使用は内部ネットワークに限定するべきである。

■TXTレコード

TXT（Text）レコードは、任意の文字列を記述するためのリソースレコードである。TXTレコードの書式は以下のようになっている。

```
<owner> <ttl> <class> TXT <txt-
strings>
```

<txt-strings>には、任意の文字列を記述することができる。このTXTレコードは、さまざまな目的で利用することが可能だが、BINDではアクセス制御リストを記述するためにこのTXTレコードを利用している。

リソースレコードを記述する際のチェックポイント

慣れている管理者であっても、ゾーンデータファイルを確実に記述することは難しい。ここでは、管理者がゾーンデータファイルを記述するうえで、特にまちがえやすいポイントをいくつか説明しよう。このような点については、RFC 1912で詳しく述べられているので、こちらもぜひ参照していただきたい。

①NSレコードで指定するネームサーバおよびMXレコードで指定するメールサーバの名称は、別名ではなく正規名で記述する

[まちがった例]

```
@   IN NS   ns1 ←ns1はfooの別名
ns1 IN CNAME   foo
foo IN A   192.168.0.10
```

[正しい例]

```
@   IN NS  foo ←正規名であるfooを記述する
foo IN A 192.168.0.10
```

②CNAMEレコードを連鎖させない

[まちがった例]

```
www IN CNAME foo  ←fooはbarの別名
foo IN CNAME bar
bar IN A   192.168.0.30
```

[正しい例]

```
www IN CNAME bar ←正規名であるbarを記述する
foo IN CNAME bar ←正規名であるbarを記述する
bar IN A 192.168.0.30
```

③同じownerに対して、CNAMEレコードとほかのリソースレコードを重複して記述しない

[まちがった例]

```
$ORIGIN example.com.
@   IN NS   ns1
    IN CNAME   www ←ownerがNSレコードと同じ
www IN A   192.168.0.30
```

[正しい例]

```
$ORIGIN example.com.
@   IN NS   ns1
    IN A   192.168.0.30
www IN CNAME  example.com.
```

④同一ドメインに対する親ゾーンと子ゾーンのNSレコードの内容を一致させる

サブドメインの管理をNSレコードによってほかのいくつかのネームサーバに委任している場合は、子ゾーンでも、それらのネームサーバがそのゾーンを管理していることをNSレコードによって記述しなければならない。これを行わないと、親ゾーンから管理を委任されているネームサーバが、そのゾーンに対して権威を持っていない状態（これを「lame delegation」という）になってしまい、不要なトラフィックが生じ

```
$TTL 86400
@    IN SOA    ns1.example.com. hostmaster.example.com.(
               2002071800 ;シリアル番号
               28800      ;リフレッシュ間隔（秒）
               7200       ;リトライ間隔（秒）
               604800     ;ゾーンの有効期間（秒）
               3600       ;ネガティブキャッシュの有効期間（秒）
               )
     IN NS     ns1        ;このゾーンのプライマリマスタ
     IN NS     ns2        ;このゾーンのセカンダリマスタ
     IN MX     10 mx1     ;プライマリメールサーバ
     IN MX     20 mx2     ;セカンダリメールサーバ
     IN A      192.168.0.30;Webサーバのアドレス
ns1  IN A      192.168.0.10
ns2  IN A      192.168.0.11
mx1  IN A      192.168.0.20
mx2  IN A      192.168.0.21
www  IN CNAME  example.com.        ;Webサーバの別名
sub  IN NS     ns1.sub.example.com.;サブドメインの管理を委任
     IN NS     ns2.sub.example.com.;サブドメインの管理を委任
               ns1.sub.example.com.
     IN A      192.168.20.10       ;グルーレコード
               ns2.sub.example.com.
     IN A      192.168.20.11       ;グルーレコード
```

リスト1● ゾーンデータファイルの記述例（example.comゾーン）

たり、名前解決に失敗したりする場合がある。

[親ゾーン（comゾーン）]

```
$ORIGIN com.
example  IN NS ns1.example.com.
         IN NS ns2.example.com.
```

NSレコードの内容を一致させる

[子ゾーン（example.comゾーン）]

```
$ORIGIN example.com.
@        IN NS ns1.example.com.
         IN NS ns2.example.com.
```

⑤FQDNで記述する場合は、最後に「.」を付けることを忘れないようにする

ゾーンデータファイルを記述する際に最も犯しやすいまちがいが、FQDNの最後の「.」の付け忘れである。これを忘れると、「www.example.com.」となるべきところが「www.example.com.example.com.」のように起点ドメイン名が繰り返されてしまうので注意しよう。

■ゾーンデータファイルの記述例

リスト1にゾーンデータファイルの記述例を示す。最初に、$TTLステートメントでデフォルトのTTLを指定し、次に、起点ドメイン名（ownerには、「@」と記述する）に対してSOAレコードとNSレコードを記述する。起点ドメイン名に対しては、MXレコードやAレコードを記述することもできるが、CNAMEレコードは記述できないので注意してほしい。この例では、起点ドメイン名のAレコードとして、Webサーバのアドレス（192.168.0.30）を記述し、「www」を起点ドメイン名（example.com）の別名として記述しているが、こうしておくと、「http://www.example.com/」と「http://example.com/」のどちらでもWebサーバに接続することが可能となる。

■ゾーンデータファイルのチェックツール

ここまでゾーンデータファイルの記述法について説

```
$ /usr/local/sbin/named-checkzone example.com. /var/named/example.com.zone
dns_master_load: /var/named/example.com.zone:20: ns2.example.com: CNAME and other data
zone example.com/IN: loading master file /var/named/example.com.zone: CNAME and other data
```

リスト2● named-checkzoneの実行例

```
# nslint
nslint: missing "a": mx2.example.com. -> 192.168.0.21
nslint: missing "ptr": mx2.example.com. -> 192.168.0.22
nslint: "cname" referenced by other "cname" or "mx": ns2.example.com.
```

リスト3● NSLINTの実行例

```
$ ./dnswalk -Fl example.com.
Checking example.com.
Getting zone transfer of example.com. from ns1.example.com...done.
SOA=ns1.example.com    contact=hostmaster.example.com
BAD: example.com NS ns2.example.com: CNAME (to foo.example.com)
WARN: mx2.example.com A 192.168.0.22: no PTR record
0 failures, 1 warnings, 1 errors.
```

リスト4● dnswalkの実行例

明してきたが、本当に正しく記述されているのかどうかを目でチェックするのは難しいだろう。このため、ゾーンデータファイルの内容をチェックするツールを利用することをお勧めする。

named-checkzone

「named-checkzone」は、BINDに含まれているツールであり、指定したゾーンデータファイルのチェックを行う。named-checkzoneを実行する際には、チェックしたいゾーンのドメイン名とゾーンデータファイルの名称を指定する。ゾーンデータファイルの内容を修正したら、BINDを再起動する前に、まずはこのツールで内容をチェックするとよいだろう。named-checkzoneの実行例をリスト2に示す。

NSLINT

「NSLINT」は、ローカルにあるBINDのゾーンデータファイルをチェックするツールであり、ftp://ftp.ee.lbl.gov/nslint.tar.gz から入手できる。

インストール後「nslint」と入力すると、/etc/named.confファイルから各ゾーンデータファイルの情報を取得し、それらの内容をチェックする。この際、チェック対象のゾーンデータファイルでBINDが動作している必要はない。NSLINTの実行例をリスト3に示す。

dnswalk

「dnswalk」は、指定したゾーンデータをゾーン転送によって取得し、その内容をチェックするツールである。dnswalkは、http://www.visi.com/~barr/dnswalk/から入手できる（ただし、インストールするためには、Net::DNS PerlモジュールとPerl IOモジュールが別に必要となる）。dnswalkの実行例をリスト4に示す。

次回は、IPアドレスからホスト名を得る逆引きの仕組みと、逆引き用ゾーンデータファイルの記述法を紹介する。

NTTデータ 馬場達也

●今回の内容に関連するRFC

RFC 1035 "Domain Names - Implementation and Specification"
RFC 1912 "Common DNS Operational and Configuration Errors"
RFC 2308 "Negative Caching of DNS Queries (DNS NCACHE)"