特集

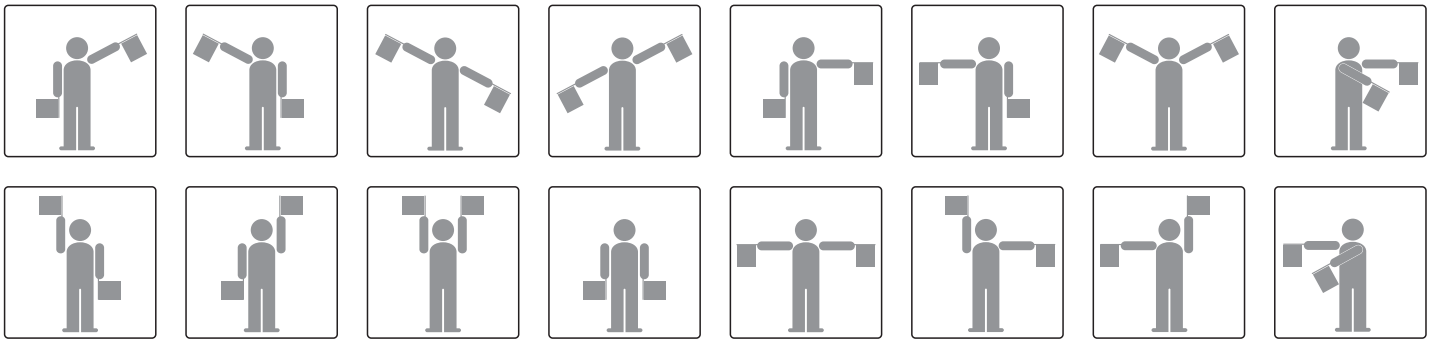
BIND設定のツボを伝授

DNS 快適運用ガイド

Part 1 DNSサーバ構築の基本手順
7つのステップでBINDを動かそう

Part 2 役割分担でセキュリティとパフォーマンスをアップ
問い合わせ“渋滞回避”のテクニック

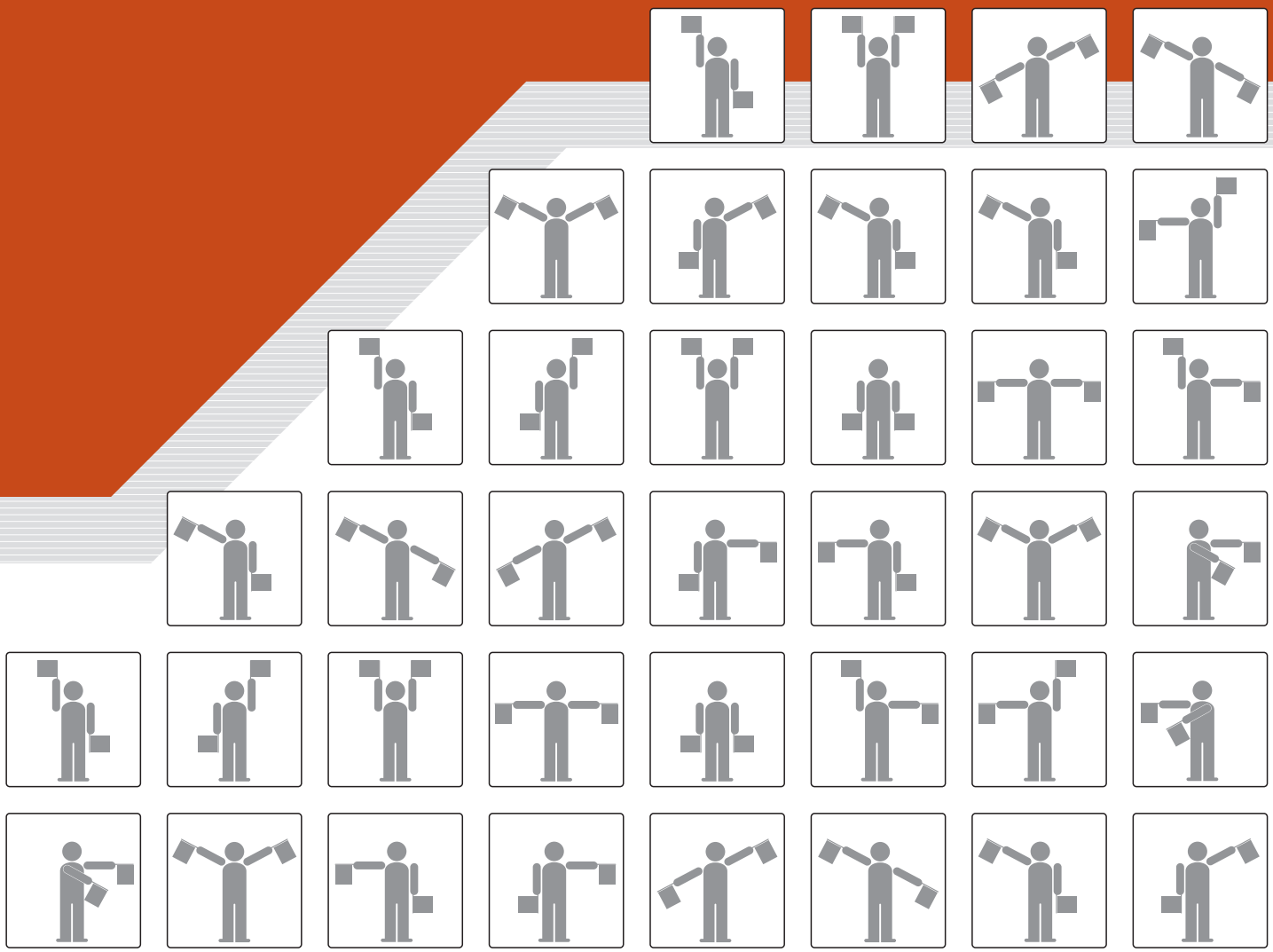
Part 3 起動トラブルもこれで解決！
BINDエラーメッセージ完全解説



最良の名前解決システムを手に入れる

DNS（Domain Name System）は、ホスト名をIPアドレスに変換するだけでなく、メール配送やサービス検索などさまざまな目的で使用されており、インターネットになくてはならない重要なシステムとなっている。最近では、個人でドメインを取得する人も多くなり、ネームサーバ（DNSサーバ）を自身で管理するようになった読者も多いだろう。しかし、DNSの仕組みや設定がわかりづらいため、まちがった設定のまま運用してしまい、これが障害の原因となっている場合も多い。本特集では、多くのサイトで利用されているオープンソースのネームサーバBIND（Berkeley Internet Name Domain）のインストールから設定、起動までを徹底的に解説し、管理者が正しくネームサーバを構築できるようにする。

馬場達也



Part 1 DNSサーバ構築の基本手順

7つのステップでBINDを動かそう

BINDはISC（Internet Software Consortium）から提供されているオープンソースのネームサーバソフトウェアであり、現在多くのサイトのネームサーバ（DNSサーバ）で使用されている。Part1では、Red Hat Linux上でBINDを動作させるまでの7つの手順を解説する。

STEP 01 RPMパッケージからBINDをインストール

BINDのソフトウェアは、ISCのWebサイト（http://www.isc.org/）から入手できる。本稿執筆時点でのBINDの最新版はBIND 9.2.2である。Red Hat Linux 8.0およびRed Hat Linux 9には、BIND 9.2.1がRPM（Red Hat Package Manager）パッケージとして付属している。しかし、セキュリティ上問題があるわけではないので、今回は、OSに付属のパッケージを利用することにする。最初に、手元のLinuxマシンにBINDがインストールされているかどうかを次のコマンドで確認しよう。

```
［Red Hat Linux 8.0の場合］
$ rpm -qa ¦ grep ^bind

bind-9.2.1-9 ……… BIND本体
bind-devel-9.2.1-9 …… BIND開発者用パッケージ
bind-utils-9.2.1-9 …… BINDユーティリティプログラム

［Red Hat Linux 9］
$ rpm -qa ¦ grep ^bind

bind-9.2.1-16 ……… BIND本体
bind-devel-9.2.1-16 …… BIND開発者用パッケージ
bind-utils-9.2.1-16 …… BINDユーティリティプログラム
```

上記のうち、「bind-9.2.1-*」と「bind-utils-9.2.1-*」の2つが表示されれば、BINDがインストールされていることになる。「bind-devel-9.2.1-*」は開発者用パッケージなので、BINDを動作させるためには必要ない。BINDがインストールされていない場合には、次のようにしてインストールを行う。まず、Red Hat Linux 8.0またはRed Hat Linux 9のインストールディスク（DISC1）をCD-ROMドライブに挿入し、rootになって次のコマンドを入力し、Linuxファイルシステムに CD-ROMをマウントする。

```
# mount -t iso9660 -r /dev/cdrom /mnt/cdrom
```

そして、BINDのRPMパッケージを次のようにしてインストールする。CD-ROMがない場合は、Red Hat Linux 8.0の場合は「ftp://ftp.redhat.com/pub/redhat/linux/8.0/en/os/i386/RedHat/RPMS/」から、Red Hat Linux 9の場合は「ftp://ftp.redhat.com/pub/redhat/linux/9/en/os/i386/RedHat/RPMS/」から入手することもできる。

```
［Red Hat Linux 8.0の場合］
# rpm -ivh /mnt/cdrom/RedHat/RPMS/bind-utils-9.2.1-9.i386.rpm
# rpm -ivh /mnt/cdrom/RedHat/RPMS/bind-9.2.1-9.i386.rpm
# umount /mnt/cdrom

［Red Hat Linux 9の場合］
# rpm -ivh /mnt/cdrom/RedHat/RPMS/bind-utils-9.2.1-16.i386.rpm
# rpm -ivh /mnt/cdrom/RedHat/RPMS/bind-9.2.1-16.i386.rpm
# umount /mnt/cdrom
```

また、BINDでは、ルートネームサーバのIPアドレスを得るためのファイル（ルートヒントファイル）が必要となる。このルートヒントファイルは、/var/named/ディレクトリに「named.ca」や「named.cache」「named.root」という名前で存在するが、存在しない場合は、「ftp://ftp.rs.internic.net/domain/named.root」から入手して「/var/named」ディレクトリにコピーしておこう。

STEP 02 named.confファイルでBINDを設定

それではBINDの設定を始めよう。本稿では、正引きゾーンである「example.com」ゾーンと、その逆引きゾーンである「0.135.

163.in-addr.arpa」ゾーンを設定すると仮定して説明を行う。「example.com」ゾーンには表1に示すマシンが存在し、このうち、「ns1.example.com」および「ns2.example.com」が、現在設定を行おうとしているネームサーバである。

BINDでは、BINDの設定ファイルであるnamed.confファイルと、ゾーンの情報を記述するためのゾーンデータファイルを設定する必要がある。ここでは、最初に、BINDの設定ファイルであるnamed.confファイルの設定を行う。Red Hat Linuxでは、named.confファイルは「/etc」ディレクトリに置かれる。

プライマリネームサーバ（ns1.example.com）におけるnamed.confファイルの設定例をリスト1に、セカンダリネームサーバ（ns2.example.com）におけるnamed.confファイルの設定例をリスト2に示す。named.confファイルの設定を行う場合には、rootになってから行おう。

リスト1およびリスト2の「//」で始まる行はコメントである。named.confファイルでは、コメントの記述形式として、C++形式の「//」、C形式の「/*～*/」、シェル形式の「#」が使用できる。

また、named.confファイルでは、いくつかのステートメントを記述できるが、基本的には、optionsステートメントとzoneステートメントを記述することになる。それぞれのステートメントの記述方法は以下のとおりである。

optionsステートメントには、ネームサーバ全体にかかわる設定を記述する。optionsステートメントの書式は次のようになっている。

```
options {
      <option>;
     [<option>; ...]
};
```

<option>には、optionsステートメント中で記述可能なサブステートメントを記述する。optionsステートメント中で記述すべきサブステートメントには、directoryサブステートメントがある。directoryサブステートメントには、ゾーンデータファイルなどを置いておくディレクトリを指定する。Red Hat Linuxでは、通常「/var/named/」を指定する。

zoneステートメントは、BINDが管理するゾーンごとに、そのゾーンに閉じた設定を記述する。zoneステートメントの書式は次のようになっている。

ホスト名	役割	IPアドレス
ns1.example.com	プライマリネームサーバ	163.135.0.10
ns2.example.com	セカンダリネームサーバ	163.135.0.11
mx1.example.com	プライマリメールサーバ	163.135.0.20
mx2.example.com	セカンダリメールサーバ	163.135.0.21
www.example.com	Webサーバ	163.135.0.30

表1● example.comにはこれらのホストが存在する

```
// optionsステートメント
options {
        directory "/var/named/";
};
// ルートゾーン用のzoneステートメント
zone "." {
        type hint;
        file "named.root";
};
// 0.0.127.in-addr.arpaゾーン用のzoneステートメント
zone "0.0.127.in-addr.arpa" {
        type master;
        file "0.0.127.in-addr.arpa.zone";
};
// example.comゾーン用のzoneステートメント
zone "example.com" {
        type master;
        file "example.com.zone";
};
// 0.135.163.in-addr.arpaゾーン用のzoneステートメント
zone  "0.135.163.in-addr.arpa" {
        type master;
        file "0.135.163.in-addr.arpa.zone";
};
```

リスト1● プライマリネームサーバの「named.conf」ファイルの設定例。zoneステートメントのtypeサブステートメントで「master」と設定する

```
// optionsステートメント
options {
        directory "/var/named/";
};
// ルートゾーン用のzoneステートメント
zone "." {
        type hint;
        file "named.root";
};
// 0.0.127.in-addr.arpaゾーン用のzoneステートメント
zone "0.0.127.in-addr.arpa" {
        type master;
        file "0.0.127.in-addr.arpa.zone";
};
// example.comゾーン用のzoneステートメント
zone "example.com" {
        type slave;
        masters { 163.135.0.10; };
        file "example.com.bak";
};
// 0.135.163.in-addr.arpaゾーン用のzoneステートメント
zone  "0.135.163.in-addr.arpa" {
        type slave;
        masters { 163.135.0.10; };
        file "0.135.163.in-addr.arpa.bak";
};
```

リスト2● セカンダリネームサーバのnamed.confファイルの設定例。zoneステートメントのtypeサブステートメントで「slave」と設定する

```
zone <zone-name> <zone-class> {
    <zone-options>;
   [<zone-options>; ...]
};
```

<zone-name>にはゾーン名（ドメイン名）を記述し、<zone-class>には、そのゾーンのネットワーククラス（インターネットの場合は「IN」）を記述する。<zone-class>を省略すると、ネットワーククラスとして自動的に「IN」が設定される。<zone-options>には、zoneステートメント中で記述可能なサブステートメントを記述する。zoneステートメント中で記述すべきサブステートメントとしては、次のものがある。

■ typeサブステートメント

ゾーンデータファイルの原本を管理するプライマリネームサーバの場合は「master」、プライマリネームサーバなどからゾーン転送により取得したゾーンデータのコピーを管理するセカンダリネームサーバの場合は「slave」と記述する。また、ルートゾーンに対しては「hint」と記述する。

■ mastersサブステートメント

セカンダリネームサーバの場合のみ設定し、ゾーン転送要求を行う先のネームサーバのIPアドレスを記述する。

■ fileサブステートメント

optionsステートメントのdirectoryサブステートメントで指定したディレクトリに存在するゾーンデータファイルのファイル名を指定する。セカンダリネームサーバの場合は、ゾーン転送によって取得したゾーンデータを格納するファイルのファイル名を指定する。ルートゾーンの場合は、ルートヒントファイルのファイル名を指定する。

STEP 03 ゾーンデータファイルを作成

次に、プライマリネームサーバ上では、リスト3およびリスト4のように正引きゾーンおよび逆引きゾーンのゾーンデータファイルを作成する。それぞれのファイルは、named.confファイルのdirectoryサブステートメントで指定したディレクトリに、fileサブステートメントで指定したファイル名で作成する。リスト1の例では、正引きゾーンのゾーンデータファイルは「/var/named/example.com.zone」、逆引きゾーンのゾーンデータファイルは「/var/named/0.135.163.in-addr.arpa.zone」となる。なお、セカンダリネームサーバでは、ゾーン転送によって自動的にゾーンデータファイルが作成されるため、ここでゾーンデータファイルを作成しておく必要はない。

ゾーンデータファイル中では、「;」（セミコロン）を付ければ、そのあとにコメントを挿入することもできる。また、ゾーンデータファイルの内容は、「$」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。次にその記述法について説明する。

制御ステートメントには、$ORIGIN、$INCLUDE、$TTLなどがあり、それぞれの書式は次のとおりである。

```
$TTL 86400
@       IN      SOA     ns1.example.com.    hostmaster.example.com. (
                                2003051000 ; 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.example.com.    ; プライマリネームサーバ
        IN      NS      ns2.example.com.    ; セカンダリネームサーバ
        IN      MX      10 mx1.example.com. ; プライマリメールサーバ
        IN      MX      20 mx2.example.com. ; セカンダリメールサーバ
        IN      A       163.135.0.30        ; Webサーバ
ns1     IN      A       163.135.0.10        ; プライマリネームサーバ
ns2     IN      A       163.135.0.11        ; セカンダリネームサーバ
mx1     IN      A       163.135.0.20        ; プライマリメールサーバ
mx2     IN      A       163.135.0.21        ; セカンダリメールサーバ
www     IN      CNAME   example.com.         ; Webサーバの別名
```

リスト3● 正引き用ゾーンデータファイルのexample.comゾーンにおける記述の例。ここでは「/var/named/example.com.zone」ファイルとして作成している

■$ORIGIN制御ステートメント

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。ホスト名の最後に「.」が付かない場合は、$ORIGIN制御ステートメントで指定したドメイン名（これを「起点名」という）がそのあとに付加され、FQDNとなる。$ORIGIN制御ステートメントの書式は「$ORIGIN <domain-name>」のようになっている。

<domain-name>には、起点名を記述する。$ORIGINで指定した起点名は、次の$ORIGIN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は$ORIGINを記述しなくてもよい。

■$INCLUDE制御ステートメント

$INCLUDE制御ステートメントを使用すると、その個所に別のファイルの内容を挿入することができる。$INCLUDE制御ステートメントの書式は「$INCLUDE <file-name>」のようになっている。<file-name>には、挿入したいファイルの名称を記述する。

■$TTL制御ステートメント

$TTL制御ステートメントには、リソースレコードのデフォルトのキャッシュの有効期限（TTL値）を記述する。各リソースレコードにおいてTTLが記述されなかった場合は、このデフォルトのTTL値がセットされるため、ゾーンデータファイルに必ず記述しよう。$TTL制御ステートメントの書式は「$TTL <ttl>」のようになっている。

<ttl>には、デフォルトのTTL値を記述する。このTTL値は、秒単位で記述するが、BIND 9では、1m（1分）、1h（1時間）、1d（1日）、1w（1週間）というような表記も可能である。$TTLで指定したデフォルトの有効期間は、次の$TTL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。

リソースレコードは、基本的に次のような形式で記述される。

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

<owner>は、行の最初に記述し、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。行の最初にタブやスペースなどが入ると、<owner>が省略されたと見なされ、その前のリソースレコードで記述された<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のキャッシュの有効期間を秒単位で記述する。<ttl>を省略した場合には、直前の$TTLでセットされたデフォルトの有効期間がセットされる。

<class>には、ネットワーククラスを記述し、インターネットでは「IN」と記述する。<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、リソースレコードのタイプを記述する。

<rdata>には、リソースレコードのタイプによって異なる内容を記述する。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できる。

■SOAレコード

SOA（Start of Authority）レコードは、権威のあるゾーンの最初に記述され、このネームサーバが、そのゾーンに関して権威を持っていることを示す。ゾーンデータ中で、SOAレコードは最初に

```
$TTL 86400
@       IN      SOA     ns1.example.com.   hostmaster.example.com. (
                                2003051000 ; 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.example.com.   ; プライマリネームサーバ
        IN      NS      ns2.example.com.   ; セカンダリネームサーバ
10      IN      PTR     ns1.example.com.   ; プライマリネームサーバ
11      IN      PTR     ns2.example.com.   ; セカンダリネームサーバ
20      IN      PTR     mx1.example.com.   ; プライマリメールサーバ
21      IN      PTR     mx2.example.com.   ; セカンダリメールサーバ
30      IN      PTR     example.com.       ; Webサーバ
```

リスト4●　逆引き用ゾーンデータファイルの0.135.163.in-addr.arpaゾーンにおける記述の例。「/var/named/0.135.163.in-addr.arpa.zone」ファイルとして作成している

1つだけ記述する。SOAレコードの書式は次のようになっている。

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

<owner>には、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、起点名が自動的にセットされる。

<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ネームサーバのホスト名を記述する。このホスト名は、後述するCNAMEレコードで定義された別名ではなく、Aレコードで定義された正規名である必要がある。

<mbox>には、このゾーンの管理者の連絡先（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。しかし、「@」の前に「.」が存在する場合は、この「.」を「¥.」に変更する。例えば、「hostmaster@example.com」は「hostmaster.example.com」となり、「tatsuya.baba@example.com」は「tatsuya¥.baba.example.com」となる。

<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、YYYYMMDDNN（YYYY=年、MM=月、DD=日、NN=その日のリビジョン番号）で表記することが多い。

<refresh>には、セカンダリネームサーバが、ゾーン転送のために、プライマリネームサーバのゾーンデータのシリアル番号をチェックする間隔（リフレッシュ間隔）を秒単位で指定する。

<retry>には、セカンダリネーム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に行うリトライの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。ゾーン転送によりゾーンデータを取得してからこの有効期間が過ぎると、セカンダリネームサーバは取得したゾーンデータを無効にしなければならない。

<minimum>には、否定応答のキャッシュ（ネガティブキャッシュ）の有効期間を秒単位で指定する。

■Aレコード

A（Address）レコードは、ホストの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「<owner> <ttl> <class> A <address>」のようになっている。

<owner>には、ホスト名を記述し、<address>には、そのホストのIPv4アドレスを記述する。<owner>に記述したホスト名は、特に「正規名」と呼ばれる。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る。これは、サーバの負荷分散の目的で使用される。

■CNAMEレコード

CNAME（Canonical Name）レコードは、Aレコードで記述したホスト名（正規名）に対して、エイリアス（別名）を付与するためのリソースレコードである。CNAMEレコードの書式は次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、その別名に対する正規名を記述する。

■MXレコード

MX（Mail Exchange）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>にはメールアドレスの「@」以降の部分にあたるドメイン名を記述し、<preference>にはメールサーバの優先度、<exchange-dname>にはメールサーバのホスト名を記述する。<exchange-dname>には、後述するCNAMEレコードで定義された別名ではなく、Aレコードで定義された正規名を記述する必要がある。優先度は、値の小さいほうが優先されるため、プライマリメールサーバの優先度をセカンダリメールサーバの優先度よりも小さく設定する。

■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。NSレコードの書式は次のようになっている。

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

<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を記述する。この<name-server-dname>には、後述するCNAMEレコードで定義された別名ではなく、Aレコードで定義された正規名を記述する必要がある。

■PTRレコード

正引きでは、Aレコードを使用して、ホスト名からIPアドレスへの変換を行うが、逆引きでは、PTR（Pointer）レコードを使用して、IPアドレスからホスト名への変換を行う。PTRレコードの書式は次のようになっている。

```
<owner> <ttl> <class> PTR <dname>
```

<owner>には、ドメイン名形式に変換されたIPアドレスを記述し、<dname>には、そのIPアドレスに該当するホスト名をFQDNで記述する。<owner>には、IPアドレスを左右逆に記述して、最後に「in-addr.arpa.」というドメイン名を付加して記述する。例えば、「192.168.0.10」の場合は、「10.0.168.192.in-addr.arpa.」のようになる。逆引きゾーンには、正引きゾーンで記述したAレコードに対するPTRレコードをすべて記述する。

STEP 04

ゾーンデータファイルのアクセス権をチェック

Red Hat Linux 8.0では、BINDのデーモンであるnamedは、「named」というユーザーの権限で動作する。このため、次に示すように、ゾーンデータファイルの所有者を「named」に変更し、さらにアクセス権を設定して、named以外のユーザーが書き込みや読み込みをできないようにしておく。

```
# chown named.named /var/named/*.zone
# chmod 600 /var/named/*.zone
# chown named.named /var/named/named.root
# chmod 600 /var/named/named.root
```

STEP 05

付属ツールで設定内容をチェック

設定が完了したら、各設定ファイルの内容をチェックする。BIND 9には、「named.conf」ファイルの内容をチェックする「named-checkconf」と、ゾーンデータファイルの内容をチェックする「named-checkzone」というツールが付属している。

named.confファイルのチェックを行うためには、root権限で次のようにコマンドを発行し、エラーが出力されないことを確認する。

```
# /usr/sbin/named-checkconf /etc/named.conf
```

次に、ゾーンデータファイルのチェックを行うために、ゾーンファイルごとにroot権限で次のようにコマンドを発行し、エラーが出力されないことを確認する。

```
# /usr/sbin/named-checkzone example.com. /var/named/example.com.zone
zone example.com/IN: loaded serial 2003051000
OK
# /usr/sbin/named-checkzone 0.135.163.in-addr.arpa. /var/named/0.135.163.in-addr.arpa.zone
zone 0.135.163.in-addr.arpa/IN: loaded serial 2003051000
OK
```

STEP 06

BINDを起動してサービス開始

次に、root権限で次のコマンドを発行してnamedを起動させる。

```
# /etc/rc.d/init.d/named start
namedを起動中:                                [  OK  ]
```

また、OS起動時にnamedを起動させるようにするために、次のように設定しておく。

```
# /sbin/chkconfig named on
```

BINDが正常に立ち上がったかどうかをログファイルで確認してみよう。BINDのメッセージログは、デフォルトでは「/var/log/messages」ファイルに記述される。正常に起動されれば、次のように出力されるはずである。

```
$ cat /var/log/messages ¦ grep named
May  10 14:24:49 server named[29770]: starting BIND 9.2.1-u named
May  10 14:24:49 server named[29770]: using 1 CPU
May  10 14:24:49 server named[29773]: loading configuration from '/etc/named.conf'
May 10 14:24:49 server named[29773]: no IPv6 interfaces found
May 10 14:24:49 server named[29773]: listening on IPv4 interface lo, 127.0.0.1#53
May 10 14:24:49 server named[29773]: listening on IPv4 interface eth0, 163.135.0.10#53
May 10 14:24:49 server named[29773]: command channel listening on 127.0.0.1#953
```

```
May 10 14:24:49 server named[29773]: zone 0.135.163.in-addr.arpa/IN: loaded serial 2003051000
                                     ↑ 0.135.163.in-addr.arpaゾーンがロードされた
May 10 14:24:49 server named[29773]: zone example.com/IN: loaded serial 2003051000
                                     ↑ example.comゾーンがロードされた
May 10 14:24:49 server named[29773]: running
                                     ↑ namedが起動
May 10 14:24:49 server named[29773]: zone example.com/IN: sending notifies (serial 2003051000)
May 10 14:24:49 server named[29773]: zone 0.135.163.in-addr.arpa/IN: sending notifies (serial 2003051000)
```

また、次のようにpsコマンドでBINDのデーモンであるnamedが起動していることを確認する。

```
$ ps -ef ¦ grep named
named   29771     1  0 14:24 ?        00:00:00 named -u named
```

STEP 07 digコマンドでネームサーバの動作をチェック

設定したネームサーバが正常に動作していることを確認しよう。digコマンドを発行して、「ANSWER SECTION」に、指定したホスト名に対するIPアドレスが含まれていれば、正常に動作している（リスト5）。また、「AUTHORITY SECTION」に設定したNSレコードが、「ADDITIONAL SECTION」に設定したネームサーバのAレコードがすべて含まれていることを確認しよう。

```
$ dig @localhost www.example.com. ····· www.example.comのIPアドレスを問い合わせる場合

; <<>> DiG 9.2.1 <<>> @localhost www.example.com.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60392
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        86400   IN      CNAME   example.com.
example.com.            86400   IN      A       163.135.0.30 ····· www.example.comのIPアドレス

;; AUTHORITY SECTION:
example.com.            86400   IN      NS      ns2.example.com. ····· 設定したネームサーバ
example.com.            86400   IN      NS      ns1.example.com. ····· 設定したネームサーバ

;; ADDITIONAL SECTION:
ns1.example.com.        86400   IN      A       163.135.0.10 ····· ネームサーバのIPアドレス
ns2.example.com.        86400   IN      A       163.135.0.11 ····· ネームサーバのIPアドレス

;; Query time: 5 msec
;; SERVER: 127.0.0.1#53(localhost)
;; WHEN: Sat May 10 13:37:21 2003
;; MSG SIZE  rcvd: 159
```

リスト5●　設定したネームサーバが正常に動作していることを確認する

Part 2 役割分担でセキュリティとパフォーマンスをアップ

問い合わせ“渋滞回避”のテクニック

ネームサーバのセキュリティとパフォーマンスを向上させるためには、権威のあるゾーンデータを管理する機能と、ほかのゾーンの名前解決を行ってその結果をキャッシュする機能を別のサーバで行うことが推奨されている。Part2では、これらの機能を実現するための設定を紹介する。

効率的な名前解決のかぎとなる2つのネームサーバ

Authoritative-onlyネームサーバは、ほかのネームサーバからの問い合わせに対して、自身が管理するゾーンの情報のみを返答するネームサーバである（図1）。クライアントから再帰問い合わせ要求を受けても、自身が管理するゾーンの中に目的のレコードがなければ、ほかのネームサーバに問い合わせを行わずにクライアントにエラーを返却する。

これに対して、Caching-onlyネームサーバは、自身ではゾーンを管理しないローカルネームサーバとして動作し、クライアントからの問い合わせに対して、目的のレコードがキャッシュに存在する場合はその情報を返答し、キャッシュに存在しない場合は、ほかのネームサーバに反復問い合わせを行って名前解決を行い、その結果を返答する（図2）。名前解決によって得られた結果は、キャッシュとして指定された期間にわたり保持する。

プライマリとして動作するAuthoritative-onlyネームサーバの設定例をリスト6に、セカンダリとして動作するAuthoritative-onlyネームサーバの設定例をリスト7に、そして、Caching-onlyネームサーバの設定例をリスト8に示す。

アクセス制御をacl名定義で実現

ゾーンデータの各ステートメントの中で、Part1で説明しなかったものを次に説明する。

aclステートメントでは、optionsステートメントやzoneステートメントで記述するアクセス制御用のサブステートメントで使用するアドレスグループ（ACL名）を定義する。aclステートメントの書式は次のようになっている。

```
acl <acl-name> {
   <match-element>;
   [<match-element>; ...]
};
```

<acl-name>には、アドレスグループの名前（ACL名）を記述し、<match-element>には、IPアドレスや、ネットマスク付きのネットワークアドレスを記述する。

次のacl名についてはデフォルトで定義されている。

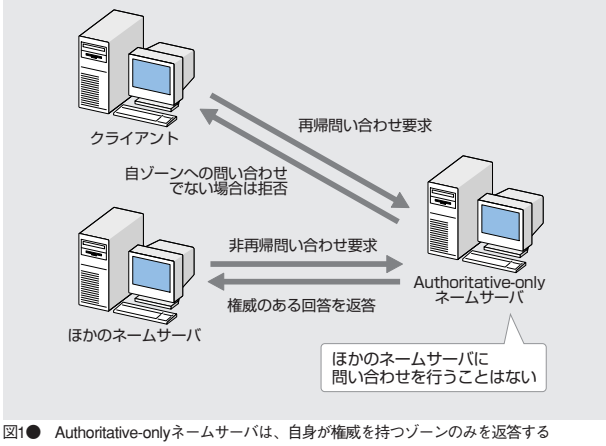


図1● Authoritative-onlyネームサーバは、自身が権威を持つゾーンのみを返答する

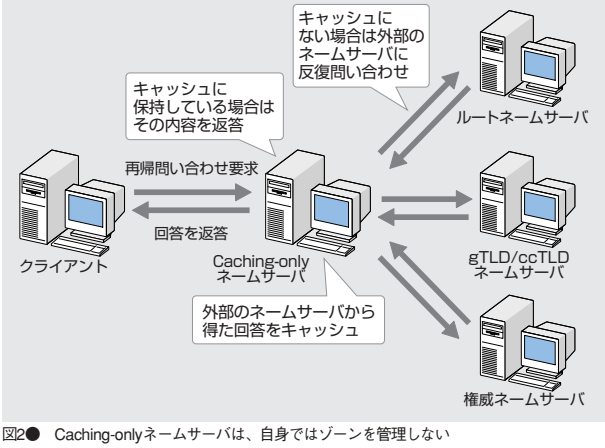


図2● Caching-onlyネームサーバは、自身ではゾーンを管理しない

●any…すべてのIPアドレスにマッチする
●localhost…ローカルシステムで使用されているIPアドレスにマッチする
●localnets…ローカルシステムが接続されているネットワーク上のIPアドレスにマッチする
●none…どのIPアドレスにもマッチしない

optionsステートメントで再帰問い合わせを制限

次に、optionsステートメント中で記述するサブステートメントについて説明する。

■recursionサブステートメント

Authoritative-onlyネームサーバは、自身が管理していない情報の問い合わせを受けた場合に、外部のネームサーバに問い合わせを行うことはせず、自身の管理するゾーンへの問い合わせに対する返答に専念する。そのため、クライアントからの再帰問い合わせ要求には答えないように設定する。BINDでは、デフォルトで再帰問い合わせ要求を受け付ける設定になっているので、「options」ステートメントで「recursion no;」と設定し、再帰問い合わせ要求を拒否するように設定する。これを行わないと、再帰問い合わせ要求があった場合に、ほかのネームサーバに問い合わせを行ったり、その結果をキャッシュしたりと、余計な処理が生じてしまうことになり、権威ネームサーバとしての本来の機能に支障を来たす。

クライアントから名前解決の問い合わせを受けるCaching-onlyネームサーバの場合は、再帰問い合わせ要求を受け付けるので、このサブステートメントを設定しないか、または「recursion yes;」と設定する。

■allow-recursionサブステートメント

再帰問い合わせを受け付けるホストのIPアドレス（ネットワークアドレス）を記述する。デフォルトでは、すべてのホストからの再帰問い合わせ要求を受け付ける設定になっている。「recursion no;」として再帰問い合わせを受け付けないようにし

```
// aclステートメント
acl "bogus" {          ……問い合わせの送信元として起こりえないアドレスを設定
        0.0.0.0/8;     ……予約アドレス
        1.0.0.0/8;     ……予約アドレス
        2.0.0.0/8;     ……予約アドレス
        169.254.0.0/16 ……リンクローカルアドレス
        192.0.2.0/24;  ……テストアドレス
        224.0.0.0/3;   ……マルチキャストアドレス
        10.0.0.0/8;    ……プライベートアドレス
        172.16.0.0/12; ……プライベートアドレス
        192.168.0.0/16;……プライベートアドレス
};
// optionsステートメント
options {
        directory "/var/named/";
        recursion no;  ……再帰問い合わせを受け付けない
        allow-recursion { none; }; ……再帰問い合わせを受け付けない
        blackhole { bogus; }; ……上記で設定したアドレスからの問い合わせを無視する
};
// example.comゾーン用のzoneステートメント
zone "example.com" {
        type master;
        file "example.com.zone";
        allow-query { any; }; ……すべてのクライアントからの問い合わせを受け付ける
        allow-transfer { 163.135.10.11; }; ……セカンダリネームサーバのみゾーン転送を許可する
        allow-update { none; }; ……ダイナミックアップデートを受け付けない
        notify yes;    ……NOTIFYメッセージを送信する
};
// 0.135.163.in-addr.arpaゾーン用のzoneステートメント
zone  "0.135.163.in-addr.arpa" {
        type master;
        file "0.135.163.in-addr.arpa.zone";
        allow-query { any; }; ……すべてのクライアントからの問い合わせを受け付ける
        allow-transfer { 163.135.10.11; }; ……セカンダリネームサーバのみゾーン転送を許可する
        allow-update { none; }; ……ダイナミックアップデートを受け付けない
        notify yes;    ……NOTIFYメッセージを送信する
};
```

リスト6● Authoritative-onlyネームサーバ（プライマリ）における「named.conf」ファイルの設定例。再帰問い合わせ要求を受け付けないように設定する

```
// aclステートメント
acl "bogus" {  ..... 問い合わせの送信元として起こりえないアドレスを設定
        0.0.0.0/8; ..... 予約アドレス
        1.0.0.0/8; ..... 予約アドレス
        2.0.0.0/8; ..... 予約アドレス
        169.254.0.0/16 ..... リンクローカルアドレス
        192.0.2.0/24; ..... テストアドレス
        224.0.0.0/3;  ..... マルチキャストアドレス
        10.0.0.0/8; ..... プライベートアドレス
        172.16.0.0/12;..... プライベートアドレス
        192.168.0.0/16; ..... プライベートアドレス
};
// optionsステートメント
options {
        directory "/var/named/";
        recursion no; ..... 再帰問い合わせを受け付けない
        allow-recursion { none; }; ..... 再帰問い合わせを受け付けない
        blackhole { bogus; }; ..... 指定したアドレスからの問い合わせを無視する
};
// example.comゾーン用のzoneステートメント
zone "example.com" {
        type slave;
        masters { 163.135.0.10; };
        file "example.com.bak";
        allow-query { any; }; ..... すべてのクライアントからの問い合わせを受け付ける
        allow-transfer { none; }; ..... ゾーン転送のリクエストを受け付けない
        allow-update { none; };  ..... ダイナミックアップデートを受け付けない
        notify no;  ..... NOTIFYメッセージを送信しない
};
// 0.135.163.in-addr.arpaゾーン用のzoneステートメント
zone  "0.135.163.in-addr.arpa" {
        type slave;
        masters { 163.135.0.10; };
        file "0.135.163.in-addr.arpa.bak";
        allow-query { any; }; ..... すべてのクライアントからの問い合わせを受け付ける
        allow-transfer { none; }; ..... ゾーン転送のリクエストを受け付けない
        allow-update { none; }; ..... ダイナミックアップデートを受け付けない
        notify no;  ..... NOTIFYメッセージを送信しない
};
```

リスト7● Authoritative-onlyネームサーバ（セカンダリ）における「named.conf」ファイルの設定例。再帰問い合わせ要求とゾーン転送要求を受け付けないように設定する

```
// aclステートメント
acl "internal" { ..... 内部ネットワークで使用しているアドレスを設定
        localhost;
        10.0.0.0/24;
        192.168.0.0/24;
};
// optionsステートメント
options {
        directory "/var/named/";
        allow-query { internal; }; ..... 問い合わせ元を制限する
        allow-recursion { internal; }; ..... 再帰問い合わせ要求を制限する
};
// ルートゾーン用のzoneステートメント
zone  "." {
        type hint;
        file "named.root";
};
// localhostゾーン用のzoneステートメント
zone  "localhost" {
        type master;
        file "localhost.zone";
        allow-transfer { none; }; ..... ゾーン転送のリクエストを受け付けない
        allow-update { none; };  ..... ダイナミックアップデートを受け付けない
        notify no; ..... NOTIFYメッセージを送信しない
};
// 0.0.127.in-addr.arpaゾーン用のzoneステートメント
zone  "0.0.127.in-addr.arpa" {
        type master;
        file "0.0.127.in-addr.arpa.zone";
        allow-transfer { none; }; ..... ゾーン転送のリクエストを受け付けない
        allow-update { none; }; ..... ダイナミックアップデートを受け付けない
        notify no; ..... NOTIFYメッセージを送信しない
};
```

リスト8● Caching-onlyネームサーバにおける「named.conf」ファイルの設定例。問い合わせを受け付けるIPアドレスを内部に制限する

ている場合は、このサブステートメントを記述する必要はないが、再帰問い合わせを一部のホストにのみ許可したい場合に、このサブステートメントを使用して制限する。

●blackholeサブステートメント

外部に公開されたサーバは、さまざまな不正アクセスを受けるようになる。特に、ネームサーバはその被害を受けやすいサーバの1つである。このような不正アクセスは、身元を隠すために、インターネットで使用されていないアドレスを使用して行われることが多い。そこで、この「blackhole」サブステートメントを使用して、インターネットで使用されていないアドレスからの問い合わせパケットを処理せずに破棄するように設定する。この設定は、1999年に発行されたAusCERT（Australian Computer Emergency Response Team）の文書（AL-1999.004-AUSCERT ALERT Denial of Service（DoS）attacks using the Domain Name System（DNS））でも推奨されている。

問い合わせやゾーン転送も制限

次のサブステートメントは、optionsステートメントまたはzoneステートメントのどちらにも記述することができる。

■allow-queryサブステートメント

問い合わせを受け付けるホストのIPアドレス（ネットワークアドレス）を記述する。デフォルトでは、どのホストからの問い合わせも受け付けるようになっている。外部に公開するネームサーバの場合は制限をしないので、何も記述しないか、「any」と設定する。また、Caching-onlyネームサーバの場合は、このサブステートメントを使用して、問い合わせを内部のネットワークのみに制限するのがよい。このサブステートメントは、optionsステートメント中で記述するとすべてのゾーンに設定が適用され、zoneステートメント中で記述すると、そのゾーンにのみ設定が適用される。

■allow-transferサブステートメント

ゾーン転送を許可するセカンダリネームサーバのIPアドレスまたはセカンダリネームサーバの持つTSIG（Transaction Signature）鍵を記述する。デフォルトでは、どのホストからのゾーン転送要求も受け付けてしまうので、必ずこの設定をしておくようにする。この設定を行わないと、むだなゾーン転送により、ネームサーバに余計な負荷がかかるだけでなく、ゾーン情報をすべて知られてしまい、不正アクセスを行うための情報として利用されてしまう可能性がある。optionsステートメント中で記述するとすべてのゾーンに設定が適用され、zoneステートメント中で記述すると、そのゾーンにのみ設定が適用される。

zoneステートメントでダイナミックアップデートを禁止

次に、zoneステートメント中で記述するサブステートメントについて説明する。

■allow-updateサブステートメント

ダイナミックアップデートを許可するホストのIPアドレス（ネットワークアドレス）を記述する。デフォルトでは、どのホストからの要求も受け付けないようになっている。通常は、何も記述しないか、「none」と設定し、ダイナミックアップデートを許可しないように設定する。ダイナミックアップデートを使用したい場合は、TSIGなどの相手認証機能を利用し、第三者が不正にゾーン情報を更新できないように設定する必要がある。

■notifyサブステートメント

ゾーンデータが更新されたことを知らせるためのNOTIFYメッセージを、セカンダリネームサーバに通知するかどうかを指定する。デフォルトでは、NOTIFYメッセージを通知するように設定されている。プライマリネームサーバでは、NOTIFYメッセージをセカンダリネームサーバに通知する必要があるので「yes」と設定する。しかし、セカンダリネームサーバでは、ほかのセカンダリネームサーバが、そのセカンダリサーバをゾーン転送の要求先（マスタ）として設定していないのであれば、NOTIFYメッセージを送信する必要はないので、「no」と設定しておく。

Column

DNS Cache Poisoningを防ぐ

DNSへの攻撃として代表的なものに、DNSのキャッシュ機能を悪用した「DNS Cache Poisoning」（「キャッシュ汚染」や「キャッシュへの毒入れ」と訳される）というものがある。この攻撃は、攻撃者が用意したネームサーバに不正なゾーン情報を設定し、その不正なゾーン情報をキャッシュをさせたいネームサーバに対して再帰問い合わせを行って、そのネームサーバから攻撃者が用意したネームサーバに対して問い合わせが発行するように仕向け、その結果、不正なゾーン情報をキャッシュさせてしまうというものである。

不正なゾーン情報のキャッシュに成功すると、誤ったキャッシュ情報を使用して名前解決を行ってしまい、問い合わせを行ったクライアントは、攻撃者の意図したサイトへ誘導されてしまう可能性がある。このため、外部からアクセスされるネームサーバは、キャッシュを行わないように設定しなければならない。

Part 3 起動トラブルもこれで解決！

BINDエラーメッセージ完全解説

Part1、Part2では、BIND 9を動作させるまでの設定について紹介してきたが、なかなか思うように起動せず、管理者を悩ませることも多い。Part3では、BINDを起動する際に発生するエラーとその原因、対処法を説明する。

ERROR 01 named.confファイル

ログファイルに「'{' expected near 'foo'」または「'}' expected near 'bar'」「'}' expected near end of file」などと出力される

```
May 10 21:08:06 server named[32253]: /etc/named.conf:16: unknown option 'zone'
May 10 21:08:06 server named[32253]: /etc/named.conf:25: unknown option 'zone'
May 10 21:08:06 server named[32253]: /etc/named.conf:33: '}' expected near end of file
May 10 21:08:06 server named[32253]: loading configuration: unexpected token
```

原因と対処法

かっこを付け忘れている

このエラーは、named.confファイルにおいて、かっこを付け忘れているのが原因である。各ステートメントや、サブステートメント中で指定したIPアドレスなどは、すべてかっこ「{}」でくくるようにする。

ERROR 02 named.confファイル

ログファイルに「missing ';' before 'foo'」と出力される

```
May 10 21:01:24 server named[32210]: /etc/named.conf:19: missing ';' before 'file'
May 10 21:01:24 server named[32210]: loading configuration: failure
```

原因と対処法

行の最後の「;」を記述し忘れている

このエラーは、「named.conf」ファイルにおいて、行の最後に「;」を記述し忘れているのが原因である。各ステートメントの最後や、サブステートメント中で指定したIPアドレスの最後には、必ず「;」を記述するようにする。

ERROR 03 ゾーンデータファイル

ログファイルに「zone example.com/IN: has 0 SOA records」または「zone example.com/IN: has no NS records」と出力される

```
May 10 15:05:35 server named[30086]: zone example.com/IN: could not find NS and/or SOA records
May 10 15:05:35 server named[30086]: zone example.com/IN: has 0 SOA records
```

原因と対処法

SOAレコードまたはNSレコードが定義されていない

このエラーは、ゾーンデータファイルにおいて、SOAレコードが記述されていないか、または、そのゾーンに対するNSレコードが1つも記述されていないのが原因である。SOAレコードは、ゾーンデータファイルの先頭に必ず記述し、NSレコードは、該当するゾーンを提供するネームサーバを、プライマリネームサーバとセカンダリネームサーバで記述する必要がある。BIND 9では、このエラーがある場合は、ゾーン全体が無効となるので注意が必要である。

ERROR 04 ゾーンデータファイル

ログファイルに「no TTL specified; using SOA MINTTL instead」と出力される

```
May 10 14:04:18 server named[29676]: example.com.zone:2: no TTL specified; using SOA MINTTL instead
```

原因と対処法

デフォルトTTLが記述されていない

このエラーは、ゾーンデータファイルにおいて、$TTLが記述されていないのが原因である。BIND 9では、$TTLを見つけられないと、デフォルトのTTLとして、SOAレコードの<minimum>値を使用する。この値は、以前はデフォルトのTTL値として使用されていたが、現在は、ネガティブキャッシュのTTL値として再定義されている。ネガティブキャッシュのTTLには、通常のTTL値より短い値（3,600秒など）が設定されるので、ゾーンデータファイルの先頭に「$TTL 86400」のようにして、必ず$TTLを記述しよう。

ERROR 05 ゾーンデータファイル

ログファイルに「CNAME and other data」と出力される

```
May 10 13:44:48 server named[29473]: dns_master_load: example.com.zone:20: www.example.com: CNAME and other data
May 10 13:44:48 server named[29473]: zone example.com/IN: loading master file example.com.zone: CNAME and other data
```

原因と対処法

CNAMEレコードとほかのレコードを重複して書いている

このエラーは、該当するゾーンデータファイルに、次のようにCNAMEレコードとほかのレコードが重複して書かれていることが原因である。CNAMEレコードで定義した別名に対しては、ほかのどのレコードも記述してはならないため、CNAMEレコード以外のレコードを削除する必要がある。BIND 9では、このエラーがある場合は、そのゾーン全体が無効となるので注意が必要である。次のように対処する。

```
www    IN    CNAME   example.com.
       IN    TXT     "This is an alias name for example.com."
```

↑ 削除するかコメントとして記述する

ERROR 06 ゾーンデータファイル

ログファイルに「multiple RRs of singleton type」と出力される

```
May 10 16:00:57 server named[30614]: dns_master_load: example.com.zone:21: www.example.com: multiple RRs of singleton type
May 10 16:00:57 server named[30614]: zone example.com/IN: loading master file example.com.zone: multiple RRs of singleton type
```

原因と対処法

同じタイプのレコードが複数定義されている

このエラーは、次のように、同じowner名に対して1つしか記述することができないレコードタイプを持つリソースレコード（CNAMEレコードやSOAレコードなど）を複数記述した場合などに出力される。

```
www    IN    CNAME   www1
       IN    CNAME   www2
www1   IN    A       163.135.10.30
www2   IN    A       163.135.10.31
```

この場合は、次のように記述する。

```
www    IN    CNAME   www1
www1   IN    A       163.135.10.30
       IN    A       163.135.10.31
```

BIND 9では、このエラーがある場合はゾーン全体が無効となるので注意が必要である。

ERROR 07 ゾーンデータファイル

ログファイルに「extra input text」と出力される

```
May 10 15:51:45 server named[30471]: dns_rdata_fromtext: example.com.zone:16: near 'Linux': extra input text
May 10 15:51:45 server named[30471]: zone example.com/IN: loading master file example.com.zone: extra input text
```

原因と対処法

通常より多い引数が指定されている

このエラーは、ゾーンデータファイルの指定された行に設定されたレコードで、規定よりも多く引数が指定されているのが原因である。例えば、HINFOレコードでは、ホストのCPUとOSの2つの引数しか記述できないが、次に示す例のように間にスペースが入ると別の引数として認識されてしまい、このエラーが発生する。

```
ns1    IN    HINFO    Pentium 4 1.8GHz Red Hat Linux 8.0
```

この場合は、次のようにCPUとOSの記述をそれぞれ「"」でくくって記述するようにする。

```
ns1    IN    HINFO    "Pentium 4 1.8GHz" "Red Hat Linux 8.0"
```

BIND 9では、このエラーがある場合はゾーン全体が無効となるので注意が必要である。

ERROR 08 ゾーンデータファイル

ログファイルに「out of range」と出力される

```
May 10 15:39:08 server named[30326]: dns_rdata_fromtext: example.com.zone:2: near '20030510000': out of range
May 10 15:39:08 server named[30326]: zone example.com/IN: loading master file example.com.zone: out of range
```

原因と対処法

値が既定をオーバーしている

このエラーは、ゾーンデータファイル内のレコードで指定した数値が、規定された範囲をオーバーしているのが原因である。SOAレコードのシリアル番号やMXレコードの優先度などの記述ミスが原因と考えられる。SOAレコードのシリアル番号は、0～4294967295、MXレコードの優先度は、0～65535の範囲でなければならない。SOAレコードのほかの値（<refresh> <retry> <expire> <minimum>）で規定された範囲をオーバーした場合は、このエラーは発生せず、規定された範囲の最大値が設定される。

BIND 9では、このエラーがある場合は、ゾーン全体が無効となるので注意が必要である。

NTTデータ　馬場達也

Column

ゾーンデータのチェックツールを有効活用

DNSのゾーンデータが正しく記述されているかどうかをチェックするツールとして、Part1では、BIND 9に付属している「named-checkzone」を紹介した。しかし、named-checkzoneでチェックする項目は、十分であるとは言い難い。そこで、さらに詳しくゾーンデータをチェックするツールを紹介する。

NSLINT

「NSLINT」は、BIND 9に付属している、ローカルのゾーンデータファイルをチェックするツールである。NSLINTは、BIND 9のソースファイルのcontribディレクトリに含まれている。また、「ftp://ftp.ee.lbl.gov/nslint.tar.gz」から最新版を入手することも可能である。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、その内容を自動的にチェックする。

Dlint

「Dlint」は、指定したゾーンデータをゾーン転送によって取得し、チェックするDNSデバッガである。Dlintは「http://www.domtools.com/dns/dlint.shtml」から入手できる。また、「DlintWeb」というのもあり、http://www.domtools.com/dlint/ にアクセスしてチェック対象のゾーンを入力すると、そのゾーンをネットワーク経由でチェックしてくれる。

DNSreport

「DNSreport」は、オンラインでゾーンデータをチェックしてくれるサイトである。「http://www.dnsreport.com/」にアクセスして、チェック対象のゾーンを入力すると、ゾーンデータだけでなく、メールサーバやWebサーバの状態もチェックしてくれる。