The Posadis DNS server

user documentation for Posadis 0.60.5

Tutorial and reference manual for Posadis
Updated for Posadis 0.60.5
Meilof Veeningen <meilof@myrealbox.com>

Table of Contents

Part 1

This part will provide a general introduction into the Domain Name System as well as task-based information on how to configure the Posadis DNS server.

Contents of this part:

Do I need DNS?

The Domain Name System, or DNS for short, is something like the telephone directory for the Internet: of you type in a domain name such as “www.posadis.org” in, for example, your browser, it will be looked up and converted to an Internet number, the Internet equivalent of a telephone number, which is used by your computer to make a connection.

A DNS server is a program that answers DNS requests from clients. Whether you need DNS depends on your particular situation.

You *need* a Domain Name Server if you want to serve information about an Internet domain name. You can, for a small or bigger fee, let somebody else do DNS, but you can do it yourself by running your own DNS server as well.

You *can use* a Domain Name Server if you want to connect a local network to the Internet. Setting up a DNS server will speed up the domain name lookup process for your local network. Additionally, you can give names to computers on your local network. The latter goal van also be achieved by using a file called “hosts” on your client PC's. Your operating system vendor has more information about that. Using “hosts” instead of DNS can safe you time if you have a relatively small and static network, though it is not as powerful and doesn't have as much applications as setting up a local DNS zone.

The Domain Name System

This chapter will provide a brief introduction into the Domain Name System, which is pretty much a crucial part of a manual that attempts to describe a Domain Name Server. This is the reason that, even when I know most of you won't do it, I strongly advise you to *read* this introduction: it isn't *that* long, and I've created some nice images that should make the reading process more bearable :)

Now, we will introduce the Domain Name System by means of a few models that will describe the Domain Name System. These models will each emphasize other aspects of the system.

The Authoritative DNS tree

The first thing we are going to note is that items in the Domain Name System are identified by *domain names*. Let's look at an example domain name:

```
www.acdam.net
```

What we see now is that this domain name consists of three *labels*: “www”, “acdam”, and “net”. Now, we're going to say that that the Domain Name System is a *hierarchical* system, in which “net” is a *subdomain* of “.”, the root domain. Likewise, “acdam.net” is a subdomain of “net” and “.”, and “www.acdam.net” is thus a subdomain of “acdam.net”, “net” and “.”. This means we can represent the Domain Name System by means of a tree structure, as in Figure 1.

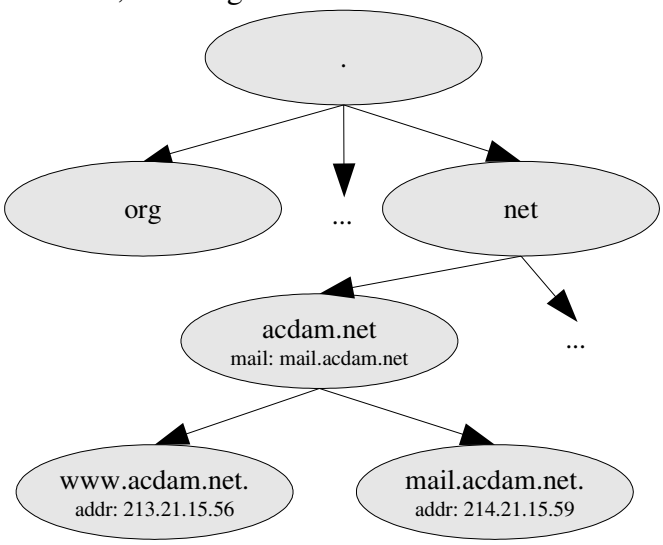

Figure 1 - The authoritative DNS tree showing data stored in domain names.

Another thing we're going to note from this figure, is the way data is stored in the Domain Name System: by means of data that is connected to the domain names. We're not going to go into great detail about how exactly this data is stored, but it is important to note that each domain name can have an arbitrary number of properties attached to it, of various types. For

example, the domain name “www.acdam.net” has a piece of data of type “addr” connected to it, with the value “213.21.15.56”. Our interpretation of this information would be that “www.acdam.net” has the IP address 213.21.15.56. Likewise, “acdam.net” has a piece of data of the “mail” type attached to it, with the value “mail.acdam.net”. We interpret this by saying that mail.acdam.net is the domain name of one of the mail servers that receive mail for the acdam.net domain.

This way of storing data means, that if we want to find out about the addresses of “www.acdam.net”, we will want to find information of the type “addr” for the domain name “www.acdam.net”.

The last thing we say about the data storage, is that *everything* in DNS is stored by means of these data chunks, connected to domain names: internal maintenance information needed by DNS as well as data meant for users to look up.

A distributed system

Now, you can imagine that given the enormous amount of domain names in existence, it wouldn't be very practical to store *all* of these domain names in one place. Indeed, DNS is a *distributed database*: different parts of the DNS tree are stored on different DNS servers. A model of this can be found in Figure 2.

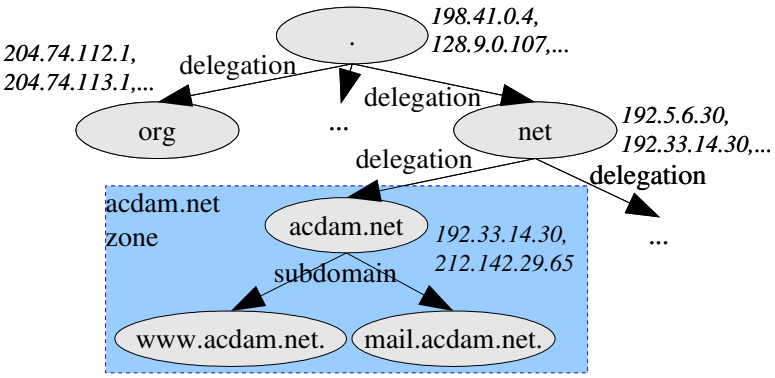

Figure 2 - The Domain Name System as a hierarchical database. The IP numbers in italics are the DNS servers for the various zones.

In this model, we see that there are a few DNS servers that store the information about the root domain “.”, which are 192.41.0.4 and 128.9.0.107, among others. These DNS servers are called the *root nameservers*. The root nameservers are called *authoritative* for “.” because they store its information.

Now, we see that these DNS servers don't store information about the subdomains of “.”, “org” and “net”: the root nameservers have *delegated the authority* about these domain names to other nameservers. In this case, the root nameservers are *only* authoritative for “.”, and not for its subdomains. In case of the “acdam.net” domain, however, we see a different picture: the two nameservers for “acdam.net”, 192.33.14.30 and 212.142.29.65, are authoritative for both “acdam.net” and its two subdomains. In general, a *zone* is the collection of the root domain for the zone and its non-delegated subdomains, so “.” is a zone of its own where

“acdam.net” is in a zone with “www.acdam.net” and “mail.acdam.net”. Of course, “acdam.net” *could* also have delegated the “europe.acdam.net” domain, for example, to ACDAM's department in Europe, and in that case “europe.acdam.net” would not have been in the “acdam.net” zone.

One thing to notice here, is that “192.33.14.30” is apparently both authoritative for “net” and “acdam.net”. This is indeed a possibility, and in fact, many nameservers are authoritative for (much) more than one zone.

Caching

We're not ready yet, though. What we can do now, however, is think about how the process of finding out information in the DNS tree (this process is referred to as *resolving*) would happen in the authoritative tree. Say we want to find the address of “www.acdam.net”. Now, a logical starting point, if we do not know about the structure of the tree, is to start at the root. We will always need the addresses of the root nameservers, but if we have them, we can figure the rest out ourselves. At first, we ask one of the root nameservers about the address of “www.acdam.net”. Now, because “net” is a delegated subdomain of “.”, the root nameserver will answer: 'I don't know, but you can ask the “net” nameservers', and it will give back the addresses of the “net” nameservers. This is called a *referral*. Likewise, the “net” DNS servers will refer us to the “acdam.net” nameservers, which will in turn return the right answer for us.

Now, to go through this whole resolving process for each domain name we need to look up would seem to be a bit inefficient. It would be nice to store a sort of 'mental image' of the DNS tree, so that when we have to lookup “mail.acdam.net”, we know we can query the “acdam.net” DNS servers directly. And in fact, if we have looked up the address for “www.acdam.net”, we can assume it remains the same for some time so we don't need to look it up every time we need it.

Client applications like your every-day web browser aren't smart enough to do this however; in fact, they are unable to resolve a domain name themselves: they expect to receive a direct answer to every query they send

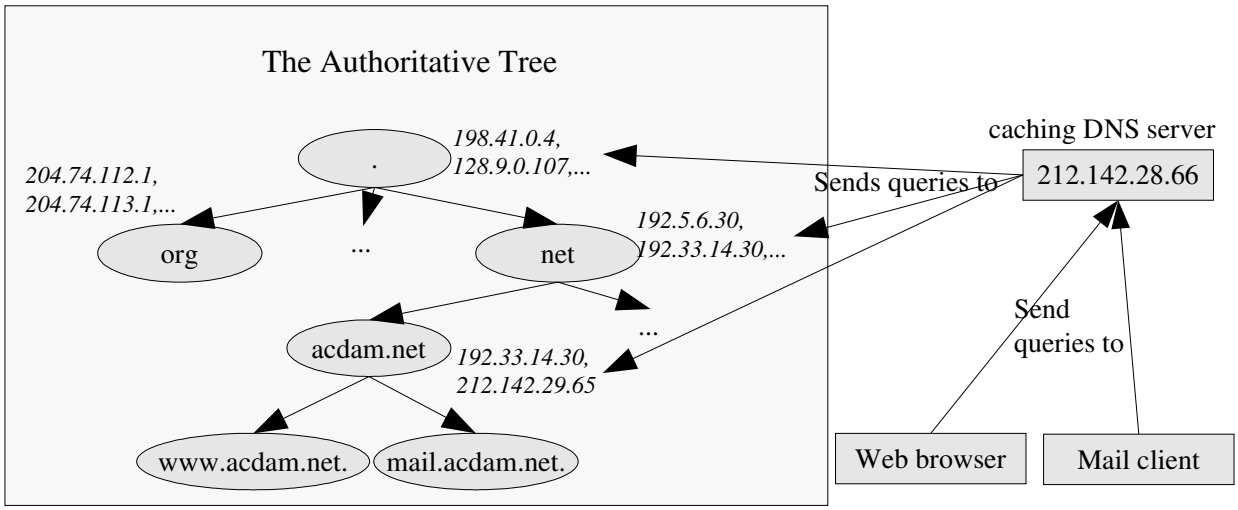

Figure 3 - Caching DNS servers

to a DNS server. That's why we have *caching DNS servers* (Figure 3).

What we see is that normal Internet applications send their DNS queries to these caching DNS servers, which will perform the heavy task of querying the authoritative DNS servers for them. Additionally, these caching DNS servers will store answers it receives as well as an image of the DNS tree internally to speed up its resolving process. This is referred to as *caching*, thus the name of these DNS servers. Pretty much each Internet provider runs its own caching nameservers for its own clients so that they have a DNS server they can reach quickly and reliably.

One thing to note about this model, is that DNS servers *can* combine authoritative and caching DNS service. Usually, in this case, they provide caching service only for selected clients, for example computers on the local network, and authoritative service to everybody.

Forwarding

Even though the caching DNS servers your ISP provides are usually quite fast, especially when they can answer directly from their cache, querying them still takes non-trivial time, and sometimes, depending on your connection speed and DNS server, *very* non-trivial time. So it would be nice to have a DNS server running on your local network, which intercepts the queries and responses from the DNS server of your ISP, and caches them so that it can return them *directly* the next time. This set-up is shown in Figure 4.

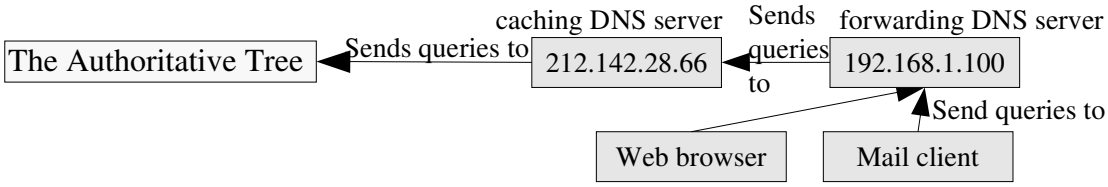

Figure 4 - Forwarding

Now, if your ISP's DNS server performs *really* badly, you can always choose to run a complete caching nameserver, but in many cases a simple forwarding DNS server works better because it has the added advantage of using the large collection of cached domain names of the caching DNS server (which presumably also has a quicker Internet connection, causing it to be able to resolve domain names more quickly).

So, what do I do next?

Now that you've read what I wanted to tell you about caching and forwarding, you're ready to set up your own forwarding or caching DNS server. Strictly speaking, in order to do that, you wouldn't have had to read this description at all, but I hope this chapter has given you a better idea on what you're doing.

- To install your DNS server and set general parameters, refer to page 12.
- To set up a caching DNS server, refer to page 14.
- To set up a forwarding DNS server, refer to page 14.

Authoritative DNS servers

If you want to set up an authoritative DNS server though, this isn't quite all you need to know. When looking back to Figure 2, we saw that “acdam.net” had two nameservers. Obviously, we will want to make sure both DNS servers send out the same data for the zone. But how do we assure both servers have the same zone data? Though you keep the data on all DNS servers in sync yourself, DNS also has a system to do this automagically by means of *zone transfers*. This is displayed in Figure 5.

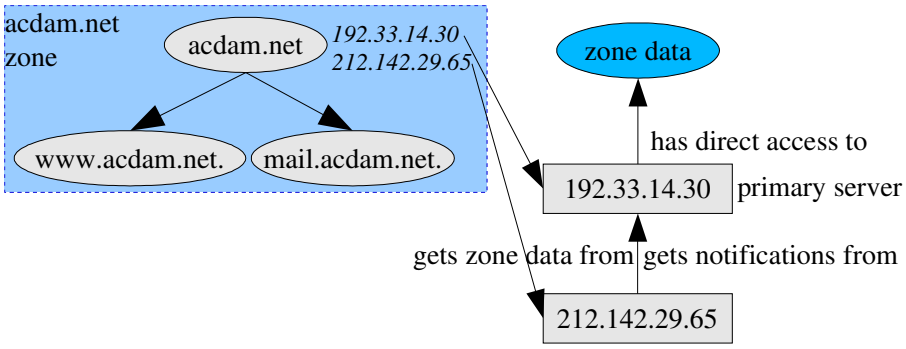

Figure 5 - Zone transfers between authoritative DNS servers

Now, we see that though there are two DNS servers for “acdam.net”, there is only one server, called the *primary server*, which has direct access to the zone data (that is, it usually has the zone data stored on disk). Now, the other DNS server gets its own copy of the zone by doing a zone transfer from the primary server. This is a completely automatic process.

Whenever something chances in the zone, the secondary will need to get an updated version of the zone as well. If an update is necessary, the primary server will notify the secondary server, saying “you need to do a zone transfer again”. The secondary will re-transfer the zone, and it can start answering queries with the new zone data right away.

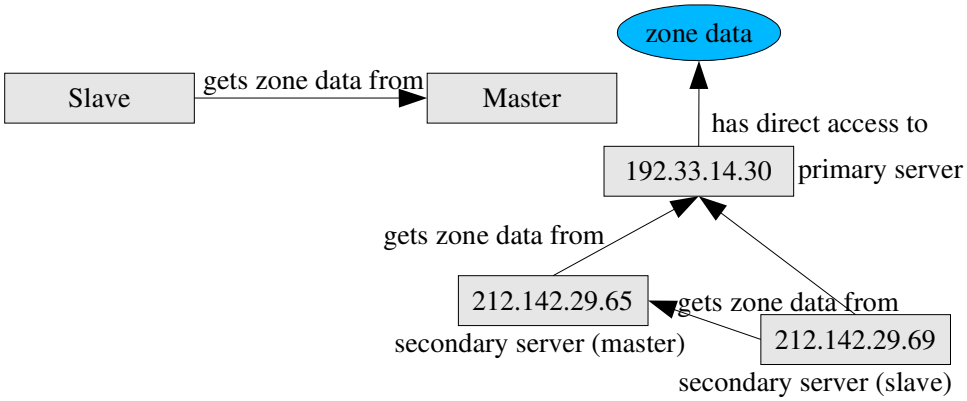

Figure 6 - Relationship between slaves and masters (left), and a set-up with two secondary DNS servers (right).

Now, let's make the set-up a bit more complicated by adding another DNS server for “acdam.net” (Figure 6). Now, the secondary server *could* just get its data from the primary server itself, but it might just as well get the data from the other secondary server. For example, if the two secondary DNS servers are on the same network, a zone transfer between them will presumably be a lot faster than a zone transfer from the primary DNS

server. In the figure, 212.142.29.69 gets its zone data from *both* other DNS servers: it will randomly try one of them, and if it fails, it will try the other.

Now, 212.142.29.65 and 212.142.29.69 are both secondary DNS servers because they do not have direct access to the zone data. Still, there is a difference between them two, so we will need to introduce one other piece of terminology, the *master-slave* relationship, also shown in Figure 6. By definition, a *slave* gets its zone data from its *master*. In the “acdam.net” case, 192.33.14.30 only serves zone transfers, so it is a master. 212.142.29.69 only gets zone transfers, so it's a slave. And 212.142.29.65 both serves and gets zone transfers, so we call it a *slave master*.

Reverse-mapping

While one common application of the Domain Name System is forward-mapping of domain names to IP numbers, the DNS is also used for *reverse-mapping*: the conversion of IP numbers to domain names. This done by means of a special part of the DNS tree: the in-addr.arpa tree.

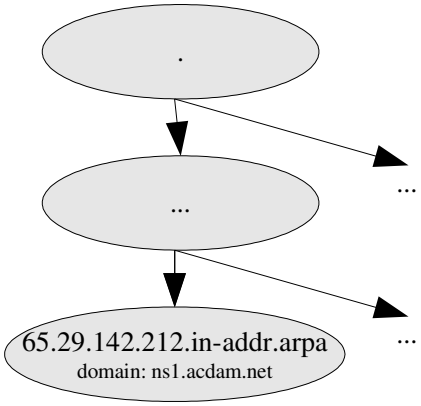

Figure 7 - Reverse-mapping of 212.142.29.65

If we want to do reverse-mapping for an IP number, we do a DNS query to a special domain name built from that IP number. For example, as one can see in Figure 7, the reverse-mapping domain name for the IP number 212.142.29.65 would become “65.29.142.212.in-addr.arpa”, and an “domain” query for that domain name would give us a domain that maps to that IP number.

So, what way do I go now?

Well, I've told pretty much everything you need to know to start setting up your own authoritative DNS server. The one thing I haven't covered yet, how pieces of data are formatted, will be covered in the chapters about primary DNS.

- To install your DNS server and set general parameters, refer to page 12.
- To register an Internet domain name, refer to page 17.
- To set up a primary DNS server, refer to page 18.
- To set up a secondary DNS server, refer to page 21.
- To set up reverse-mapping for your IP numbers, refer to page 22.

Installing your DNS server

The Posadis DNS server consists of a few packages that, depending on your particular needs, you will or will not want to install. These include:

- Poslib is the base library for DNS-related tasks that all Posadis tools depend on. You will want to download this.
- Posadis is the core of the Posadis DNS server, which you'll want because after all, that's what this book is all about...
- Posadis-monitor is a module for Posadis that automatically monitors a directory with zone files for changes. If it's available for your OS, I recommend installing it. Posadis-monitor is included in the source release of Posadis.
- If you want to become a primary master for some zones, you can optionally get Mfedit, the Posadis Master File Editor, a graphical tool to edit DNS master files. This tool can really make editing master files much easier and error-proof. It contains a built-in syntax checker and has context help about how to set up DNS data.
- Finally, an important diagnostic utility for use with DNS is a DNS query program. Posadis contains a command-line driven tool called posask which you can use, but if you want something graphical, you can use Dnsquery.

Note that for some operating systems, packages might be available that combine these software items. For example, for Windows, an installer is available that includes all of the above.

Configuring Posadis

Posadis finds out what it needs to do by reading the Posadis configuration file, posadisrc. Your Posadis distribution will provide an example posadisrc with some examples and some initial settings. Windows users will find this file in the “Config” subdirectory of the directory in which Posadis was installed, whereas Posadis for Unix will use “/etc/posadisrc”. In this file, lines starting with a semicolon (“;”) are considered comments; other lines are configuration settings.

Now, the posadisrc has a good deal of options, and they will all be explained in due time. At this point, there are only a few things we want to do.

At first we need to determine to what ports and network interfaces we want to listen. If Posadis runs without problems the first time, there is no problem, but you may find Posadis to say something like “Could not listen to network interface x.x.x.x”. If this is the case, you're already running an other DNS server (or you might not have sufficient permissions to start a server at a privileged port). This happens especially if you're running Microsoft Internet Connection Sharing, which as an internal DNS server that cannot be disabled. In this case, you need to comment out the “listen

any" line, and add another listen line in front. Read the comments in the Posadisrc for more instructions.

If you want to run an authoritive DNS server, you will also want to make sure the data directory is set up well. The data directory, determined by the "datadir" setting in your Posadis configuration file, is used by Posadis to put in temporary files, so you'll need to use a directory Posadis can safely write its data there.

Using the monitor module to set up zones

If you're planning to run an authoritative DNS server, you'll need to do something else as well: you need to choose how you want to configure your zones. In this tutorial part, we will assume you use the "monitor" module to do this. There are other ways of doing this that are slightly different though; jump to page for more information.

On Windows, the "monitor" module is already enabled in your Posadisrc. On Unix, you need to uncomment the "Loadmodule monitor" line. Note that for the module to work, you'll need to have FAM installed.

Adding a zone to your Posadis configuration is as simple as adding a file to your configuration directory. The monitor module will automagically detect the creation of new zone files, changes in your zone files and removal of zone files while Posadis is running. You just need to find out where your configuration directory is. On Windows, it is the "Config" subdirectory of your Posadis installation by default. On Unix, it is "/etc/posadis" by default. You can change this by setting the "Configfile" configuration setting in your Posadisrc.

Zone files do need to have special names though:

- Zone files for primary zones have the format <zone>.prm or db.<zone>.
- Zone files for other zone types have the format <zone>.znf or zn.<zone>.

On Windows, creating a zone file is as easy as right-clicking in the Explorer and choosing "New zone master file" (for primary zones) or "New zone definition file" for other zones. On Unix, it is as simple as touching it or opening it with your favourite editor – for primary zone files, this favourite editor is obviously the Posadis Master File Editor :)

Starting Posadis

Well, nothing much to tell really. Just start the Posadis executable. Once you start using Posadis at a regular basis though, you'll want Posadis to start at boot-time. Instructions on how to do this can be found way further in Part 2, namely on page 33.

Setting up a caching DNS server

Setting up a caching DNS server is actually quite easy. Actually, Posadis is a caching DNS server by default, so you don't need to do that much yourself. If you just run Posadis with the example Posadisrc, Posadis is a caching DNS server all by itself.

Well, that was pretty easy, wasn't it? :)

One thing to notice though is that since you will usually be running your DNS for internal use, you can, just for safety, make Posadis not listen to your Internet IP number. You can comment out the “listen any” line in your Posadisrc and add something like this instead:

```
listen 127.0.0.1, 192.168.1.100
```

This will prevent Internet clients from looking up domain names with your caching DNS server, which costs you network traffic, and which can make your network more vulnerable to outside attacks.

If you also need to listen to your Internet IP number (e.g. you run an authoritative DNS server as well), it would be a good idea to disable recursion for Internet clients:

```
allow_recursion 192.168.1.*, 127.0.0.1
```

This will only allow recursion to the IP numbers 192.168.1.1-192.168.1.255 and 127.0.0.1.

You can run a caching name server to provide DNS service to your local network. Especially if you're on a large network or your ISP has a slow DNS server, this can really help you speed up DNS resolution. Additionally, you can define internal domain names (such as “router” or “johnspc”). Instructions on how to do that can be read later on in this manual.

➢ To set up domain names for local networks, refer to page 15.

Forwarding

Setting up a forwarding DNS server works just like setting up a caching DNS server, except that you will also need to specify what DNS servers to forward queries to. This involves adding a “cache-forward” structure to your Posadisrc, for example:

```
cache-forward .
  212.142.28.66
  212.142.28.67
```

In this case, Posadis will forward queries to the two DNS servers specified here. Note that you will need to keep the “cache-ns” structure in the Posadisrc intact.

Setting up domain names for local network computers

Now that you have set up caching, you can also set up domain names for local network computers.

➢ To set up domain names for workgroup computers, refer to page 18.

Set up your workgroup computers to use the DNS server

If you set up recursive DNS for your network, you will need to tell your workgroup computers to do DNS lookups on your server. I will very briefly describe how to do that here.

On Unix systems, this usually involves editing /etc/resolv.conf and entering something like:

```
nameserver 192.168.1.100
```

Many Unix operating systems, particularly Linux distributions, have their own tools to edit the DNS configuration, so you can also use them.

On Windows systems, you will want to edit the TCP/IP properties of your local ethernet card, and you can enter DNS server addresses somewhere there. They will also ask you to enter a host name, but you can quite safely ignore that, or just enter something your host will want to call itself.

Domain names for your local network

If you use your own DNS server, why not define internal-only domain names as well? Setting up local domain names isn't radically different from setting up interet domain names, except that you do it on a DNS server which is also a recursive DNS server for your local network. We're going a bit ahead of business nowm but once you know how to do primary DNS (hint: page 18), you can for example, create a “db.localnet” zone master file, as you can see in Figure 8.

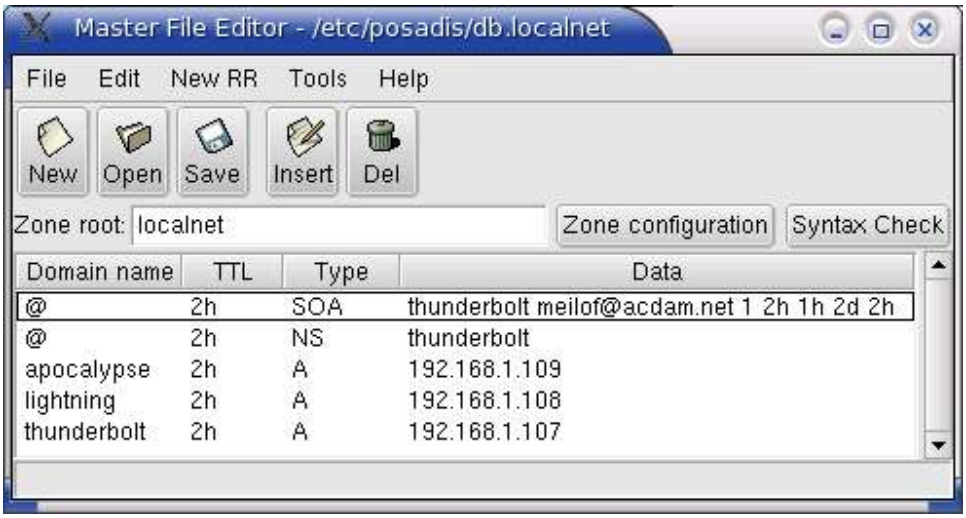

Figure 8 - DNS for your local network

So, you can now type in “thunderbold.localnet” to reach the internet gateway PC. But wouldn't it be great if you could just enter “apocalypse”, “lightning” or “thunderbolt”? Luckily, your operating system knows how to do that. If you run Unix, adding the following to your /etc/resolv.conf is usually enough:

```
search localnet
```

Now, if internet queries for “apocalypse”, “lightning” and “thunderbolt” fail, your resolver will try appending “localnet” to them, and then it will find

the correct adresses. In Windows, a similar result can be established by entering something in the “search domain” (or just “domain”) box.

You can also set up *reverse-mapping* for your local network to convert IP numbers back to domain names. Instructions for this follow later.

- To set up primary DNS, refer to page 18.
- To set up reverse-mapping, refer to page 22.

Registering an Internet domain name

If you want to register your own domain name, you'll need to do this at a *registrar*. Where you can do that differs from top-level domain to top-level domain, and apart from that, you will probably need to choose a registrar from many tens or even hundreds of available registrars. One good starting point, as usual, is the Open Directory Project (also used by Google Directory):

```
http://www.dmoz.org/Computers/Internet/Domain_Names/
```

Here, you'll find a long list of available registrars. I'm not here to endorse any hosting company, but if you really want me to name someone, I'd say Gandi, www.gandi.net, which is the registrar for the Posadis.{org,net,com} domains. You can get your own domain name there for 12 euro a year, which sounds like quite a bargain.

Technically, there isn't much to tell about registering your domain name: you will usually just need to provide a list of the domain names of your nameservers, and their IP numbers. This implies that if you register your domain name, you will already need to have your authoritative DNS servers running. Note that it is a Good Thing(tm), and some registrars might enforce this, that you have at least two DNS servers for your zone (apart from your own DNS server, which is called the primary, you will also need so-called *secondary* DNS servers).

Your very own domain name

One problem you might encounter is that your registrar will only let you enter domain names as nameservers for your zone, and not its IP numbers , whereas if you want to run your own DNS server, it hasn't got a domain name elsewhere yet. Luckily, like for most problems in life, there is a solution to this one: your ISP has probably given you your own domain name. There are some FTP and IRC servers requiring every person who connects to have his own domain, so your ISP has probably set this up for you; you just have to find out what your domain name is.

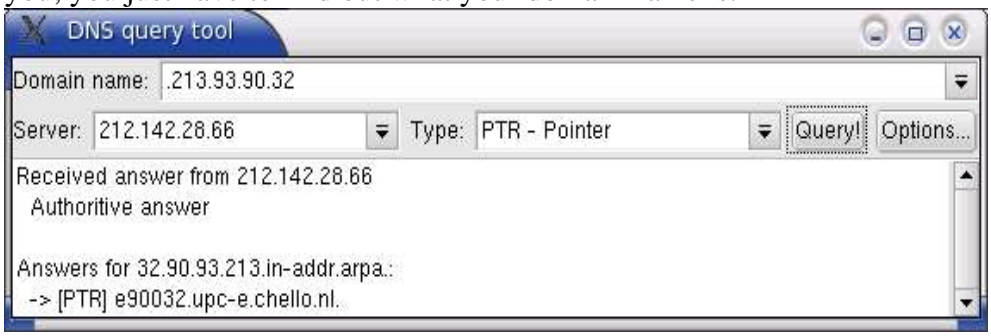

Figure 9 - Finding out your domain name

Luckily, you can do this with the Posadis DNS query tool. An example screenshot is shown in Figure 9. In this case, “e90032.upc-e.chello.nl” is apparently the domain name for the IP number 213.93.90.32. This method should only be used as a temporary solution.

Setting up a primary DNS server

Using primary DNS, you can set up Posadis to serve an internet domain name, which you need to register first. Alternatively, you can set up primary DNS if you want to assign domain names to computers on your local network. This works in just the same way, just as long as your workgroup computers are configured to use the primary DNS server (or one of its slaves) for resolving.

A primary DNS server, as you may remember from Figure 6 (page 10), is the one DNS server for a zone that stores the actual zone data. That zone data is stored in a so-called *master file*. This master file is stored in a format can be read by most DNS servers, including BIND and MS-DNS. Posadis comes with the Posadis Master File Editor, with which you can easily edit the data for your zone.

To create a zone, simply create a new zone master file in the Posadis configuration directory. If you'd want to create a master file for the “acdam.net” zone, you'd have to name the file either “db.acdam.net” or “acdam.net.prm” in order for Posadis to recognize it. Under Windows, you can create a new Master File from within the Explorer by choosing “New...” in the File menu.

Zone transfer configuration

If, apart from your primary DNS server, there are also secondary DNS servers for the zone, you should let know Posadis about them. By default, Posadis only allows zone transfers to slaves for security resons, so you'll need to let Posadis know what the slave DNS servers are. You can do this from the Master File Editor by clicking the “Zone configuration” button, and entering the slaves in the dialog box as shown in Figure 10.

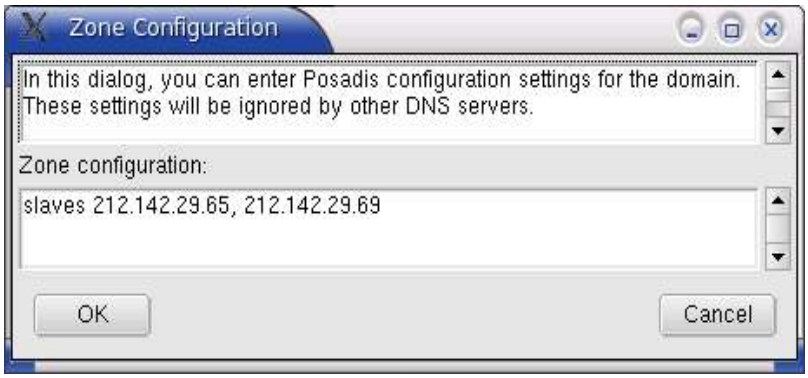

Figure 10 - Setting up the slaves for your domain.

Adding another

```
allow_xfr_to 127.0.0.1
```

line will allow Posadis to also answer zone transfer queries to local queries, which can come in quite handy for testing purposes.

Free secondary DNS services

The more DNS servers you run, the less implications it will have when one of them fails, and the less bandwith and processor load you use per DNS server. There are free secondary DNS services that will act as a secondary DNS server for your zone, for example twisted4life.com (pretty good IMHO), and secondary.org, but you can of course run your own secondary DNS server as well.

Secondary services will need to provide a list of secondary DNS servers to add to the “slaves” list. Refer to the documentation of your secondary DNS service for more instructions.

Contents of a zone

Now, let's create a zone master file for the zone. You can do this by choosing “File->New” from the Posadis Master File Editor menu, or by opening a newly created, empty master file. The Master File editor will now open up a dialog in which you can create a zone skeleton. You can use the rudimentary zone data that is created thus as a starting point for your zone.

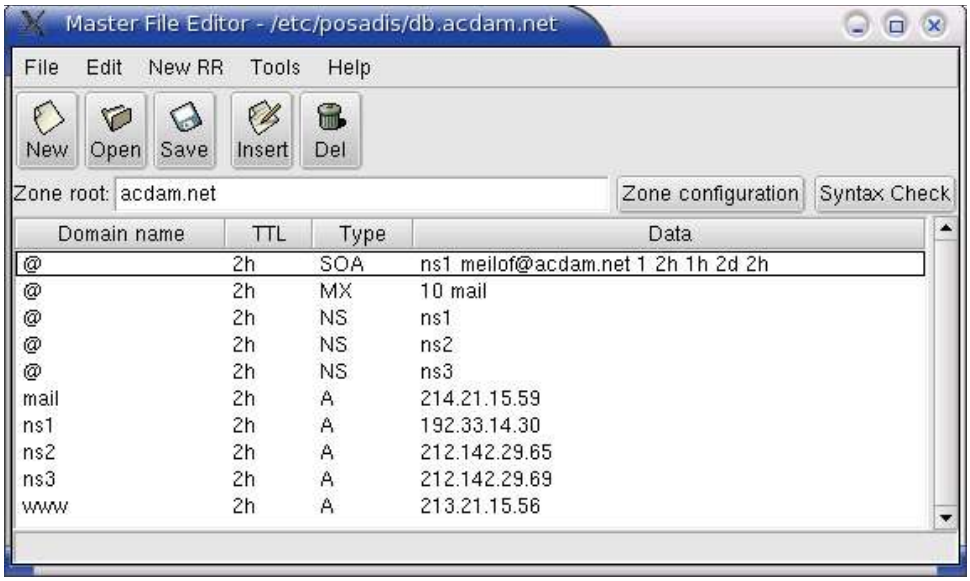

Figure 11 - Configuration of the acdam.net zone

A basic zone configuration is shown in Figure 11. In the Master File Editor, we see a list that contains the following:

- Domain name – The domain name for the zone.

Remark. Relative domain names
Domain names on zone master files are displayed relative to the zone root (that is, in the “acdam.net” zone, “ns1” means “ns1.acdam.net”, and “@”, the *origin sign*, means “acdam.net”). If you need absolute domain names (for domains outside your zone), make sure they have a trailing dot, for example “dns.secondary.com.”.

- TTL – The the time caching DNS servers may store the data. The default, “2h” (meaning two hours), is usually fine.
- Type – The type of data stored in this record.
- Data – Type-dependent data.

This is what a zone contains:

- The SOA record contains general zone information. If you want more information, double-click on the SOA record, and leave your mouse on one of the text fields to get the scoop on that parameter (for the moment you can leave them use as-is).
  One parameter is very important though: the third one, “1” in our example, which is the zone serial number. Each time you change your zone, and you want your secondary DNS servers to pick the zone up from you, you will need to increase this number otherwise secondary DNS servers won't know someothing has changed. In fact, Posadis will warn you if it re-loads a changed master file with still the same serial number, saying that you really should increase this number.
- NS records, attached to the zone root, list the DNS servers for the zone by their domain names. Note that the domain name is relative to the zone root (see the remark on relative domain names on page 19). Each domain name in your zone you use here should also have A records. If you use a third-party secondary DNS service, you'll also want to add their DNS server domain names (for example “dns.secondary.com.”) here as well.
- A MX record attached to a domain name points to mail exchange servers for the domain, that is, the server that receives mail for the domain. So if we want to receive mail for “<username>@acdam.net”, we will make a MX record for “acdam.net.”. The numeric value is a preference value which says what mail server to try first: mail servers with low priority will be tried first.
- A records determine addresses for domain names. You will need A records for domain names you referred to in SOA, MX and NS records (press the “Syntax check” button to see which domains you have forgotten), and you will also want to add them for other domains you intend to use, for example the “www” and “ftp” subdomains and so on.
- If you want to define IPv6 addresses for your hosts, you use the AAAA record type rather than, or in addition to, A records.

There are a few things we're not going to do now, but which might come in handy later:

- To delegate authority for subzones, refer to page 29.
- To learn how to set up wildcard domain names, refer to page 25.
- To learn how to set up alias domain names, refer to page 25.

Setting up a secondary DNS server

Setting up a secondary DNS server using Posadis is, luckily, a very simple process.

What you'll want to do is create a zone file in the Posadis configuration directory. If you'd want to create a zone file for “acdam.net”, that file should be called either “zn.acdam.net” or “acdam.net.znf”. If you use Windows, you can use the “File->New” menu of the Windows Exmplorer to create them.

Now, you'll need to edit this file with a text editor. You can add comments to the file using the semicolon: any text after a semicolon will be ignored. The first non-comment line in your file should be:

```
zone secondary
```

Now, you'll want to add a line defining the master DNS servers (the servers to get zone transfers from) Posadis will use:

```
masters 192.33.14.30, 212.142.29.65
```

Additionally, if your secondary DNS server should also serve zone transfers itself, it should also add a line defining slaves that are allowed to do zone transfers for the zone:

```
slaves 212.142.29.69
```

For more details, see the section on zone transfer configuration on page 18.

Reverse-mapping domain names

So far, we-ve done forward-mapping from domain names to IP numbers. DNS also provides a way to do reverse-mapping from your IP number to your domain name, as we've seen on page 11. On page 10, we also saw the format of the reverse-mapping domain names: an IP number “a.b.c.d” has a reverse-mapping domain name “d.c.b.a.in-addr.arpa”. Though it is good to know that these are just common domain names stored somewhere in the DNS tree, it is not very handy to have to type in this longish reverse-mapping domain name every time. That's why in posadis, when you mean “d.c.b.a.in-addr.arpa”, you can just type in “.a.b.c.d”. So, the reverse-mapping domain name for 212.142.29.65 would become 212.142.9.65. Makes sense, doesn't it?[1] And if you want to refer to the zone which contains reverse-mapping for 212.142.29.*, the “29.142.212.in-addr.arpa” zone, you can use (you guessed it) “.212.142.29.*”.

Now it's time to tell you how the domain name the IP number points to is stored: this is done by means of “PTR” records connected to these odd-looking domains. In fact, we have already used this information on page 17, when we tried to find our very own domain name (Figure 9).

If you just have one or two internet IP numbers, you don't need to do reverse-mapping: your ISP has done some reverse-mapping, and you really don't need to set it up yourself.

If you register a larger range of IP numbers though, for example a C-class network (that is, for example, 213.93.90.0 to 213.93.90.255), you will want to create your own reverse-mapping zone. Your network registrar will have the scoop on how you can tell them to delegate authority to your reverse-mapping zone. A reverse-mapping zone is just a normal zone except that it mainly contains PTR records instead of the A records a normal zone would mainly consist of.

One general rule to keep in mind is that it is very uncommon for a reverse-mapping domain to have more than one PTR record connected to it, so particularly there is no need to specify *all* domain names for a given IP number.

In many cases, in fact, you won't even *care* about what domain a reverse-mapping domain maps to. This is for example the case if you intend to connect a large network of client PCs to the network. For these cases, Posadis has the revmap and formap modules, that can automagically generate large reverse-mapping zones, saving you a lot of work.

- To learn how to set up a master file, refer to page 18.
- To learn about the revmap and formap modules, refer to the reference section.

1 Though this works in Posadis, other DNS servers won't know what you mean if you use them elsewhere. When communicating to other DNS servers, Posadis *does* use the normal reverse-mapping domain name format, though.

Your local network

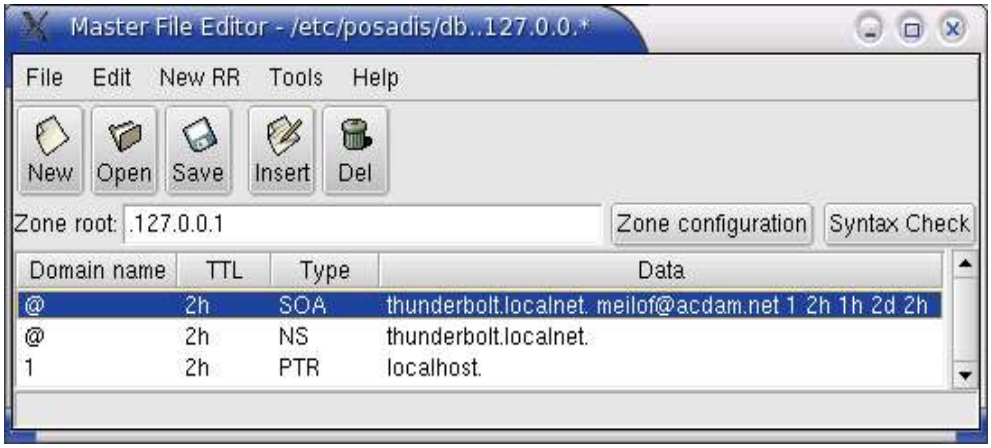

Figure 12 - Reverse-mapping localhost

It is common practice to at least create a reverse-mapping zone for the 127.0.0.1 IP address, and make it point to the well-known “localhost” domain name. We could create a primary zone as shown in Figure 12.

Remark. Filenames for reverse-mapping zones

The Linux Operating System (on which these screenshots were teken) happens to support filenames with two dots after each other and aterisks (“*”) in it, but you might not be so lucky. In that case, for the filename, you will need to resort to filenames such as “db.0.0.127.in-addr.arpa” for the 127.0.0.* reverse-mappin domain name.

Now, similarly, we can implement reverse-mapping for the local network we set up on page 15 as shown in Figure 13.

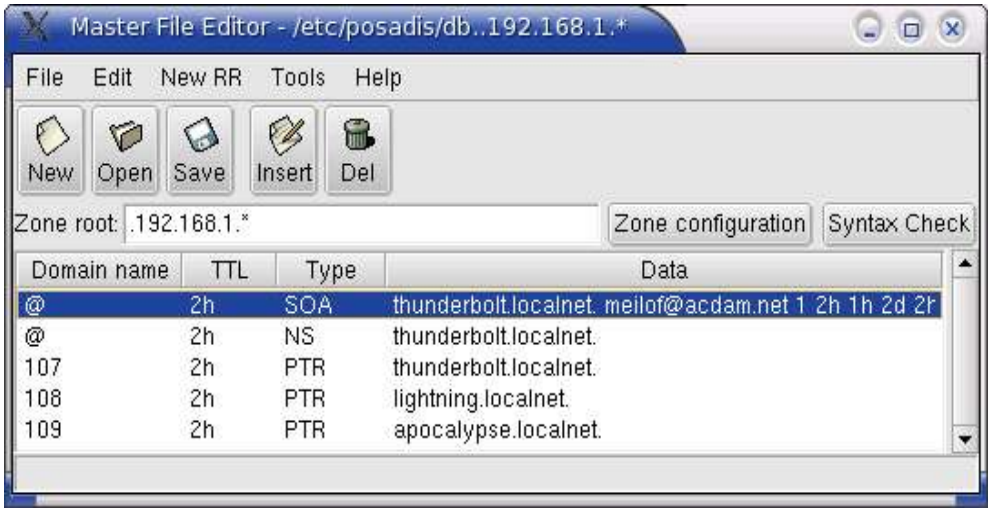

Figure 13 - Reverse-mapping a local network

So, you ask, is reverse-mapping really nessecary? Well, it might not be inevitable, but in many cases it is a good idea. Servers, particularly mail and news servers, can use reverse-mapping queries to find out to what domain you belong, and they can use that knowledge to determine whether or not to allow queries. FTP servers can be configured to only allow connections from IP numbers that can be reverse-mapped to a domain name. Also, there are tools available to analyze log files, which use reverse-mapping to convert the IP numbers you find in log files to the more readable domain

names they are mapped to. In these scenario's, you will probably want reverse-mapping. And why wouldn't you do it? It isn't that difficult, and besides, you'll have to agree with me it is great fun!

IPv6 reverse-mapping

And if IPv6 is more your thing, of course you can do IPv6 reverse mapping as well. If you want me to be entirely honest with you, there are basically two methods to do this, but only one of them is supported by Posadis. This is the method using the “ip6.int” domain, which is the companion to the AAAA resource record for IPv6 addresses.[2]

This reverse-mapping method is very similar to in-addr.arpa reverse mapping. Every hexadecimal label in the domain becomes a label of the domain name (in reverse order of course), resulting in quite long domain names. Also, you will need to drop short notations for IPv6 addresses such as the “dead::beef” notation, or “22” for a label that really is “0022”. For example, the reverse-mapping domain name for the IPv6 address “ae4:d1e4:22:f0::deaa” would become:

```
a.a.e.d.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.f.0.0.2.2.0.0.4.e.0
.0.4.e.a.0.ip6.int.
```

You can just attach PTR records to this domain name like you do in IPv4 reverse mapping. As you will understand, this reverse-mapping can become quite a pain after a while. Similar to the IPv4 reverse mapping case, Posadis does have a nonstandard extension in its domain name handling, so that a domain name of “.ae4:d1e4:22:f0::deaa” has the meaning of the reverse-mapping domain name of that IPv6 address.

2 There's a more advanced alternative out there, which consists of the A6 resource record type and the ip6.arpa reverse-mapping domain. Recent 0.50.x versions of Posadis support this method, and support for it might be added in Posadis 0.60.x in the future. This method has however been marked experimental by the IETF, so I'm not really that much in a hurry currently.

Advanced zone configuration topics

This chapter will mention two often combined techniques you can use in your zone data files: aliases and wildcards.

Aliases

Aliases do just what you expect: you can let one domain name point to another. This is done by means of the CNAME record. For example, in the configuration in Figure 14, any queries for “web.acdam.net” will be answered with the answers for “web.acdam.net”. For example, the address of “web.acdam.net” would become 213.21.15.56.

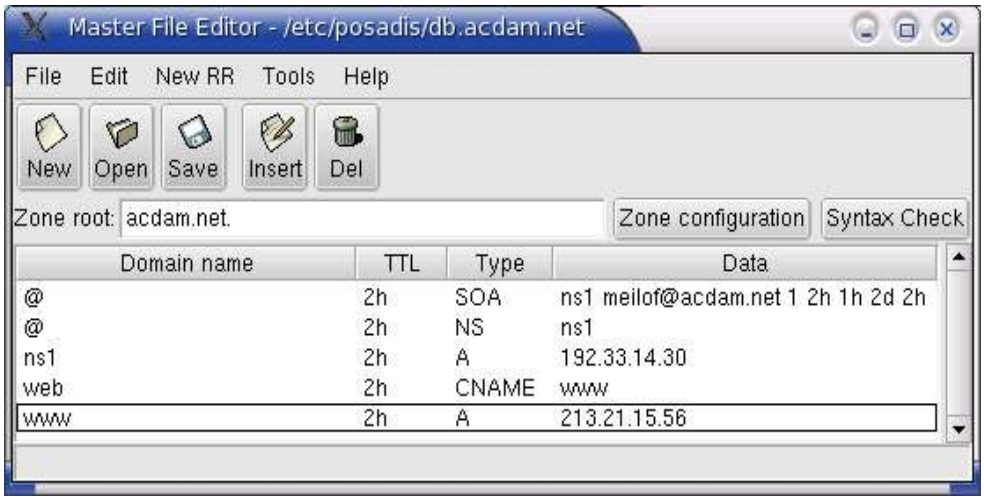

Figure 14 “web” is an alias to “www”.

When dealing with CNAMEs, there are a few things to note:

- The domain name with the CNAME may not have other records attached to it.
- The domain name a domain name refers to *can* be outside the zone itself.
- The domain name a domain name refers to *should* not be an alias itself. These setups will usually work, but they often cause unnessecary to be done while looking up the domain name.
- DNS query programs will show you which CNAMEs have been followed to answer the query.

Wildcards

Using wildcards, you can define records that apply to all non-existent subdomains of a domain. Wildcard records are just records with an “*”-label, as shown in Figure 15. In this case, all queries for non-existent subdomains will be redirected to the “www” subdomain. Things to note:

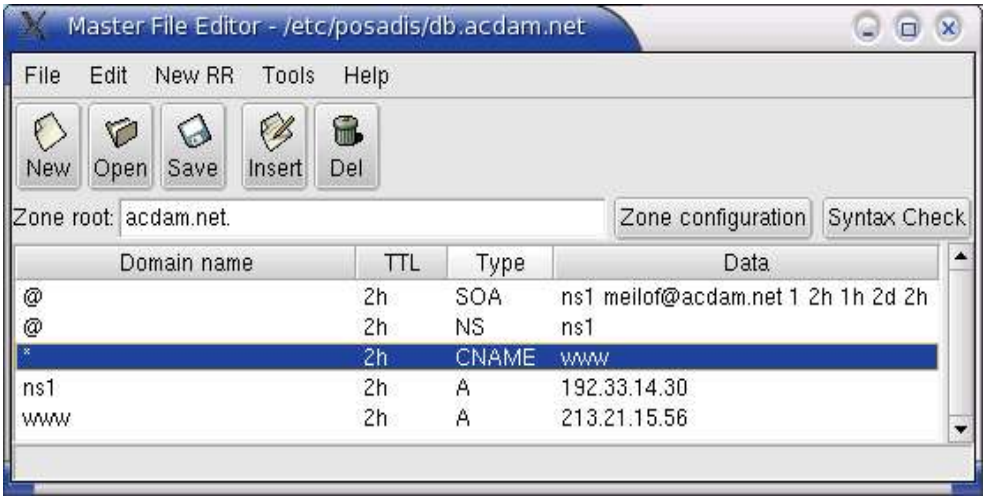

Figure 15 - A wildcard record

- Though wildcards are often used in combination with CNAMEs, this is not nessecary; for example, one can also directly attach “A” records to wildcard domains.
- Wildcards apply only to *nonexistent* domain names. For example, if “www.acdam.net” doesn't have a MX record, but “*.acdam.net” has, the DNS server will *not* answer with the MX record for “*.acdam.net”.
- Wildcards do not apply to subdomains of existent subdomains; for example, a “*.acdam.net” wildcard will *not* apply to “a.www.acdam.net” if “www.acdam.net” exists.
- When doing a query, there is no way of telling whether the query was answered by means of a wildcard or by means of existent records. You can however directly query a DNS server for wildcard records, for example, one can query “*.acdam.net” for addresses.

Using initial cache files for blocking and blacklisting

This chapter will discuss a feature called *initial cache files* and its uses for blocking access to specific domain names and for e-mail black- and whitelisting.

Initial cache files apply to the situation where you run a caching DNS server, and you want to feed custom data into your cache. This custom data will be loaded when Posadis starts up, and it can not be overrided by data Posadis might receive from other servers later on.

For example, you can set the address of “www.google.nl” to 127.0.0.1, thus in effect blocking access to that domain name. Obviously, this protection *is* circumventable by using another DNS server, but it can useful to block access to banner websites as well (analogous to setting their addresses to 127.0.0.1 in the operating system's host file as is done by some banner-blocking tools).

Another use of initial cache files is for blacklisting spammers in your mailserver. Many mail servers can use DNS lookups to check for known spammers. For example, at the time of writing the Open Relay Database [www.ordb.org] offers this kind of service: when a mail server receives an e-mail from another mail server with IP address 1.2.3.4, it will do a lookup for 4.3.2.1.relays.ordb.org, and if there is an A record attached, it will block the e-mail. You can of course create your own zone with blacklisting information, but using initial cache files, you can use the information from for example the Open Relay Database, and add your own hosts by feeding custom domain names into the cache.

Creating cache files

An initial cache files is basically just a master file, except that it doesn't have to have an initial SOA record. In fact, you can use the Posadis Master File Editor to create and edit cache files.

To create an initial cache file, just create a new zone file for the root (“.”) domain, and then check the “Is cache data file” box in the tools menu. You can then remove the SOA record that the Master File Editor created, and just add records you want added to your cache.

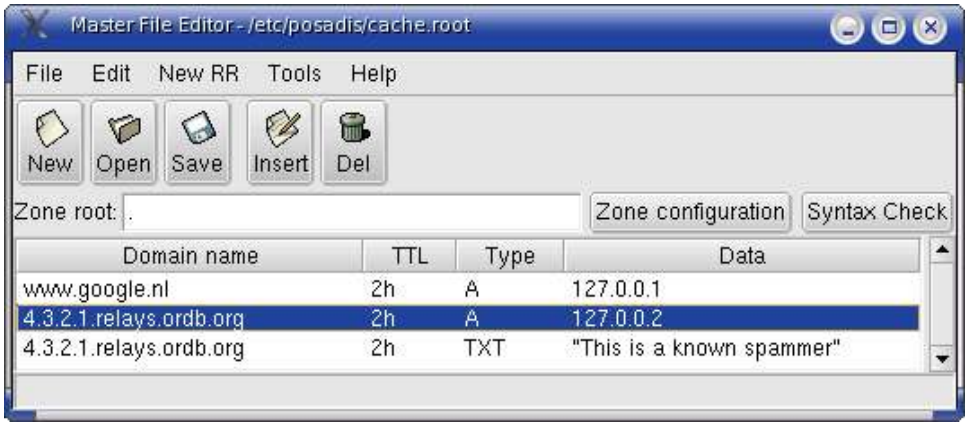

Figure 16: Using initial cache files to block domain names and blacklist spammers.

In Figure 16, we see an example ow how to block access to the domain name “www.google.nl”, and how to block mail coming from the IP address “1.2.3.4” if our mail server is set up to use the Open Relay Database.

Configuring initial cache files

You can make Posadis load your initial cache files by means of the “initial_cache_file” configuration option. For example, if the cache file would be stored as “cache.root” in the Posadis configuration directory, the following line would make Posadis load it:

```
initial_cache_file cache.root
```

You can make Posadis load an arbitrary number of initial cache files. Unlike zone master files, initial cache files will *not* be re-loaded regularly, so any changes to initial cache files can only be applied by restarting Posadis.

Delegating subdomains

Imagine this: you have a primary zone called “acdam.net”, and now you give control to “europe.acdam.net” and all of its subdomains to your friends in Europe. To do this, you'll want to *delegate* the subdomain to them (we showed how this works in Figure 2 on page 7). This is exactly what happened when you registered your internet domain name: “acdam.net” was delegated to you by the “net” nameservers.

Anyway, here's what you add to the “acdam.net” DNS master file:

Domain name	TTL	Type	Data
europe	2h	NS	ns1.europe
europe	2h	NS	ns.acdameurope.com.
ns1.europe	2h	A	212.142.28.44

What we did here is:

- listing the nameservers for the “europe” zone
- defining the addresses for the nameservers of “europe.acdam.net” that are in the “europe.acdam.net” zone themselves.

This data is collectively called the *glue data* for the “europe.acdam.net” zone.

This is about all it takes! Now the guys in europe can start running their own DNS servers authoritative for the “europe.acdam.net”, and they can start answering queries.

Maintaining your DNS server

Even if you would dearly love to say goodbye to DNS configuration after your DNS server is finally up and running, your involvement probably won't stop there. This chapter will discuss some common tasks you will encounter once your DNS server is up and running.

Updating your primary zone data

If you run a primary zone, you will want to add, remove or edit domain names in it over time. The most important thing to keep in mind is that, any time you make a change to your zone, you need to update the serial number field of the SOA record of your zone (this is the third field of the record). Updating this field will make sure secondary DNS servers will notice something changed in your zone. In fact, if you change zone data without updating the serial, Posadis will give you a warning.

Updating secondary zone data

Usually, when the primary finds out the zone data has changed, and when it has succesfully re-loaded the zone master file, it will send out so-called “DNS Notify” messages to all of its known slaves to tell them they should do a zone transfer to get the latest zone data. Thus, maintainers of secondary zones usually won't have to do anything: as soon as the primary has new zone data, the secondary will know that it needs to do an update.

If your master doesn't support DNS Notify though (Posadis servers do), the secondary checks after the interval specified by the refresh parameter of the SOA RR, whether the zone data is still current. This means that a secondary will wait for at most that time before it updates its zone data if you change anything on the primary. To make sure the data is updated directly, it is best to just re-start the DNS server.

Maintaining your cache

If you maintain a cache, there usually isn't that much to do. After, that is, you have found yourself a reasonable value for the “max_cache_items” configuration value. This is the maximum number of domain names that get cached by the Posadis DNS server. If you intend to run a serious cache, you will usually want to fine-tune this value to find the right tradeoff between being able to answer much answers from cache (thus speeding up resolution and saving bandwidth), and preventing Posadis from occupying too much memory. For this case, Posadis can provide some rudimentary cache statistics if you enable the “cache_statistics” setting in your Posadisrc. If you have, Posadis will periodically (that is, each minute) log some information on the usage of the cache to its log file. A typical log entry would look something like:

```
2004/04/04 22:36:45|info: [server] Cache statistics:
cd=257 dcd=88 ch=1 dch=1 cm=39 dcm=26
```

The fields have the following meaning:

- cd: Number of domain names stored in the cache
- dcd: Change in the number of domains in the cache over the last minute
- ch: The number of queries we could answer from cache
- dch: Change over the last minute
- cm: The number of queries we could *not* answer from cache
- dcm: Change over the last minute

Problem-solving strategies

In a perfect world, DNS servers all work flawlessly and co-operate in a friendly manner, and you usually won't see any problems (and if you do see them, they will make sure they automagically solve themselves within a minute or so). Sadly though, this world doesn't yet entirely classify as being perfect, and that's why you might be experiencing problems with your DNS server.

If you have a problem with your DNS server (for example, clients are unable to visit a specific website, or can't connect to their mail or ftp server), there are a few things you'll want to do.

Check the logs!

First, you will want to check the logs of the DNS server to see whether it has itself a clue why things don't work. In the Posadis configuration file, you can specify a file Posadis will log to, and Unix systems can just check their syslog since Posadis also logs there. For example, you might find Posadis to log something like:

```
2003/07/30 15:03:23|error: [server] Error in zone
transfer from 192.168.1.2#53 for acdam.net.:
Erroneous answer: SRVFAIL!
2003/07/30 15:03:23|error: [server] No servers could
be reached for acdam.net.: delaying zone transfer
```

In this case, you can see fairly easy what the problem is: the nameserver 192.168.1.2 gave an incorrect answer to the zone transfer query from our DNS server. In this case, the thing to do is to contact the people who manage the 192.168.1.2 nameservers, and tell them to wake up and configure their DNS server the right way.

There are a few other common problems Posadis will log messages about:

- If a syntax occurs in a master file, Posadis will start anyway, but it will answer all queries for the zone with a SRVFAIL (“server failure”) message. Posadis will tell you exactly where it found the syntax error so that you can easily fix it.
- If you want to serve zone transfers, you will need to specify a data directory for Posadids to put its zone data files in. If you get errors about data directories, will want to check whether the directory actually exists, and whether the user Posadis runs as has write access to it.

Query logging

If you have a problem that you think might be related to DNS, but you're not entirely sure what exactly the query was that caused the problem, the best way to find out is by enabeling query logging. You can enable query logging by setting the query_logging configuration setting to true. You probably won't want to have that feature enabled in normal operation though because you will notice your log file growing very quickly, so you'll want to turn it off after you've found the problem, and then restart Posadis. If you run Unix, you can temporarily enable query logging by sending a SIGUSR1 signal to Posadis, and you can disable it by sending another SIGUSR1:

```
linux# /etc/init.d/posadis status
 5730 pts/2    S      0:00 posadis -p /
var/posadis/pidfile -f
linux# kill -USR1 5730
```

After you'll enable query logging, you will find such log messages as:

```
2003/07/30 15:29:54|info: [query] Query from
127.0.0.1#402: OPCODE QUERY, q={acdam.net.,AXFR}
2003/07/30 15:30:11|info: [query] Query from
127.0.0.1#88: OPCODE QUERY, q={www.google.nl.,A}
2003/07/30 15:30:12|info: [query] Query from
127.0.0.1#88: OPCODE QUERY, q={www.google.nl.,A}[3]
```

Posadis will log all normal queries, zone transfers and NOTIFY messages.

If you see a query that you think might have caused the problem, it is time to move on to the next step:

Using the DNS query tool

If you want to test whether the DNS server gives back the results you expected, you can use the DNS query tool like we did in Figure 17.

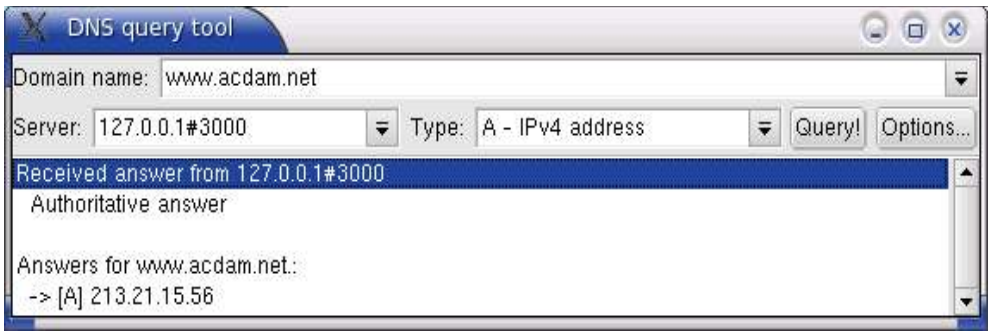

Figure 17 - A succesful query for the address of www.acdam.net

Now in this case, the query actually succeeded. But if you find the DNS server to give a “Server failure”, then usually there is an error in the master file (if it's a primary), or the zone transfer failed (if it's a secondary). If you didn't get the resut you expected, you can query for the “SOA” type for the root domain of the zone to see whether the DNS server is serving the latest data, which you can see from the serial number.

3 This second query is due to the fact that the DNS server didn't answer within one second because looking up the domain name took some time. After a second, the client decided to re-try the query.

Starting Posadis at boot-time

If you're happy with Posadis, you'll want to start it whenever your computer starts. The procedure on how to do this differs from OS to OS.

For Windows, you'll want to use the Posadis service. This works for at least Windows 2000 and XP. I've heard rumours Windows 98 and other older versions also contain rudimentary service support, but then again I don't know any details. The thing to do is to run the “Install/Remove Posadis service” entry from the Posadis Start Menu group. If all goes well, it will say something like “Posadis service installed”. It is now in the service list, and it will be started when you boot your PC. You can remove it again by re-running the program, which will now tell you the service has been removed. Alternatively, you can add a shortcut to Posadis.exe to the “Startup” folder of your Start Menu if your Windows edition doesn't support services. You will then see a “MS-DOS prompt” window Posadis runs in though.

The Posadis RPM will automagically register itself as a service. This works on at least Mandrake, Suse and RedHat Linux. These distributions each have separate tools to configure, start and stop services.

For others running a UNIX-like operating system, we provide an init script which can start and stop Posadis. This script can be found in the “tools” subdirectory; it's called posadis-init. The procedure for installing it depends on your Unix flavour. Reading “/etc/init.d/README” might help you to install this script. This will usually involve copying the init script to “/etc/init.d” and renaming it to “posadis”, and creating appropriate links in “/etc/rc.d”. The K Desktop Environment offers a tool to help you do this, called “ksysv”.

If you have the “chkconfig” script, you can install the Posadis service like this:

```
cp tools/posadis-init /etc/rc.d/init.d/
/sbin/chkconfig posadis reset
```

You can remove the service again if you don't want it anymore:

```
/etc/init.d/posadis stop
/sbin/chkconfig --del posadis
```

You can control the behaviour of Posadis by doing one of the following:

```
/etc/init.d/posadis start
/etc/init.d/posadis stop
/etc/init.d/posadis restart
/etc/init.d/posadis reload
/etc/init.d/posadis status
```

Reload will make Posadis re-load its configuration files and re-issue zone transfers, whereas restart will make Posadis re-read its posadisrc as well. Status will show you whether Posadis is already running.

Part 2

This part of the Posadis Manual will contain reference documentation for Posadis. Basically, it is just a print-out of the Posadis manual pages shipped with Posadis for Unix.

Contents of this part:

posadisrc

Updated: 0.60.5

NAME

posadisrc - Configuration file for Posadis

DESCRIPTION

This is the main configuration file of the Posadis Domain Name Server. It contains both the configuration directives used by Posadis and a list of the zones hosted by the DNS server.

The default Posadis configuration file is called */etc/posadisrc*. You can specify a custom configuration file using the "-c" command-line option.

The configuration file is basically just a plain text file. Each configuration option is on a separate, unindented line, followed by its configuration value. Everything after a ';' is ignored, so you can thus embed comments in your configuration files.

For example, you can specify configuration options/commands like this:

```
Listen tcp/3000 ; listen to port 3000 for tcp
Listen ::1      ; listen to ipv6 port 53 for tcp/udp
```

LOADING MODULES

Posadis has a modular structure for the authoritative zones, so if you intend to use any functionality that is not in the Posadis core itself, for example, to use "revmap", "formap" or "localhost" zone types, you will need to load the appropriate module first. This can be done by using the "loadmodule" statement in your posadisrc:

```
LoadModule "revmap"
LoadModule "localhost"
```

Modules can also supply other functionality, for example, the "monitor" module can automagically load all master files in a specified directory. The default location for libraries is "/prefix/lib/posadis/", where prefix is your installation prefix, /usr by default. You can add directories to the library search path using the following:

```
; add this to library search path
Libdir "/home/meilof/mymodule-1.0/lib/"
```

If you want to know the default module location, run "posadis --version". Creating Posadis documents is covered later.

The following modules are included in the core Posadis distribution:

monitor - Loads all master files in a specified directory, and monitors them for changes (for Unix, this requires the SGI File Alteration Monitor).

revmap - Provides a convenient way for automated IPv4 reverse mapping (.in-addr.arpa.). Provides "formap" and "revmap" zone types.

localhost - Returns back the IP number of the querier. Provides "localhost" zone type.

CACHE CONFIGURATION

You will need to tell Posadis where the root nameservers are, because Posadis doesn't know them itself. This can be done by the "cache-ns" setting. Basically, you'll just want to copy this if you want to use the standard root nameserver of the ICANN (as of january 2003):

```
cache-ns .
  A.ROOT-SERVERS.NET.@198.41.0.4
  B.ROOT-SERVERS.NET.@128.9.0.107
  C.ROOT-SERVERS.NET.@192.33.4.12
  D.ROOT-SERVERS.NET.@128.8.10.90
  E.ROOT-SERVERS.NET.@192.203.230.10
  F.ROOT-SERVERS.NET.@192.5.5.241
  G.ROOT-SERVERS.NET.@192.112.36.4
  H.ROOT-SERVERS.NET.@128.63.2.53
  I.ROOT-SERVERS.NET.@192.36.148.17
  J.ROOT-SERVERS.NET.@192.58.128.30
  K.ROOT-SERVERS.NET.@193.0.14.129
  L.ROOT-SERVERS.NET.@198.32.64.12
  M.ROOT-SERVERS.NET.@202.12.27.33
```

You can specify initial nameserver lists for other domains as well though that is not nessecary. Also, Posadis features a tool called "posadis-getroots" which can get the latest root nameservers for you, in the format described above. Use the program without arguments to get the latest root nameservers of ICANN, or use "posadis-getroots orsc" to get the latest root nameservers of the Open Root Server Confederation. Alternatively, enter IP numbers for other alternative roots to get the complete list for that.

Allright, now for the cache configuration. Basically, there are two ways of caching: forwarding and resolving. If you just want resolving, you should be done by now because Posadis has now enough information. If you want to do forwarding instead of caching, which is a good idea if you have a slow internet connection, you can configure forwarding for a zone like this:

```
cache-forward .
  192.168.1.1       ; nameserver to forward to
  192.168.1.2       ; another nameserver
```

This will apply to all subdomains of "." that are not part of an authoritative zone (they may be subzones of authoritative zones though). If you would want to disable forwarding for a given subdomain, use:

```
cache-forward acdam.net
```

For all subdomains of acdam.net, Posadis will try to resolve the domain name instead of forwarding queries.

OPTIONS

Libraries:

libdir - Specify an additional directory to look in for libraries. Posadis looks by default in the "lib/posadis" subdirectory of your prefix, so this would usually be "/usr/lib/posadis/".

loadmodule - Load a Posadis module. Modules can introduce new zone types and configuration settings, and thus provide new functionality to Posadis.

General:

listen - Specify an interface to listen to. You can specify an IP number (both IPv4 and IPv6), a port number (to listen on any IPv4 interface), or both (separated by a '#'). You can prefix this by "udp/" or "tcp/" to listen to only UDP or TCP, respectively.

configdir - The directory to look in for zone data and master files.

update_ttl - Interval after which to re-check zone files for changes when using the "zonefile" and "allzonefiles" statements. Defaults to 10m (10 minutes).

datadir - Directory to store temporary files in (nessecary both for serving and retrieving zone transfers).

user (Unix only) - User name to switch to after opening configuration files and binding to sockets. Note that master files should be readable by this user as well as root.

group (Unix only) - Group name to switch to after opening configuration files and binding to sockets.

Logging:

logfile - File to log messages in. By default, nothing is logged. Posadis for Unix will also log to syslog.

logfile_per_day - If set to true, Posadis will create a new logfile each day at midnight (named according to the value of 'logfile', but with the data appended). Defaults to false though.

do_query_logging - If set to true, Posadis will log each query (normal queries, zone transfers and NOTIFY messages) it receives. Useful for debugging purposes; defaults to false.

Resolving:

max_cache_items - The maximum number of domain names stored in the cache. If new domain names are added, older ones will be removed to prevent breaking this limit. Defaults to 1000.

resolv_patience - The number of operations taken at most to complete one recursive query. Applies to zones for which we do resolving for, only, and

defaults to 15.

allow_recursion - If given, only these hosts are allowed to do recursion. Can be an address or a range.

refuse_recursion - If given, hosts on this list are not allowed to do recursive queries, even if they are in the allow_recursion list as well.

cache_statistics - If this is set to true, Posadis will periodically (that is, every 60 seconds) display statics about the cache: the number of domain names and records stored, and the number of cache hits/misses. Defaults to false.

initial_cache_files - Specifies master files to initially populate the cache with. These files are looked for in the Posadis configuration directory. Initial cache files are not periodically reloaded.

cache_to_file - If set to true, Posadis will keep the contents of its cache over reboots by saving the cache to disk when it quits, and reloading it when it boots.

Resources:

max_threads - Maximum number of simultaneous threads. Defaults to 50.

max_tcp_connections - Maximum number of simultaneous TCP connections. By default set to -1, which means no limit. Of course, the max_threads limit applies to tcp connections as well.

tcp_priority - Hosts to always allow a TCP connection from. Can be a host name or an address range such as 192.168.*.

max_cname_recursion - Maximum number of recursive aliases (CNAMEs) to follow. Defaults to ten, which should really be enough for any sane configuration.

ADDING ZONES

If you want to run an authoritative DNS server, you will want to add zones to your Posadis configuration. Now, in the good tradition of Unix software there are three different ways of doing this, each with their separate advantages and disadvantages:

Using the "monitor" module: each zone is a file in your configuration directory, and changed, edited and added zone files will be detected by Posadis. On Unix, this requires FAM to be installed.

Using zone files: each zone is in a file in your configuration directory, and Posadis will poll the files for changed, though it isn't able to add or remove zones.

Using the "zone" statement: zones can be defined directly in the Posadisrc.

Using the "monitor" module

This is the recommended method. In this set-up, you have a configuration directory with zone files in them: each file corresponds to a zone you want to configure. These files have special names: *db.<zone>* or *<zone>.prm* for primary zones, and *zn.<zone>* and *<zone>.znf* for non-primary zones.

You can set the configuration directory by means of the *Configdir* setting.

By default, this is "/etc/posadis" for Unix, and the "Config" subdirectory of your Posadis install for Windows. Look for a line with the text "Configdir" in your Posadisrc to change this.

As soon as you add, change or remove a zone file in your configuration directory, Posadis will automagically re-load zone. For primary zones, this zone file is just a master file in which configuration commands can be embedded (see the "Master files and RR types" section). For non-primary zones, this file has the following syntax:

```
zone <type>
command1 value11, value12
command2 value2
...
```

The command lines can be indented, but that's not nessecary. You can also embed semicolon comments in the zone file.

Using zone files

Basically the same as using the "monitor" module, except that instead of monitoring, this method uses a polling strategy (that is, it checks every n seconds for changes in the zone files, where n is the value of the *update_ttl* configuration setting). New zone files won't be detected in this way though. It is mainly there for platforms on which the FAM server, on which this depends, is not available, or if you don't want to use FAM.

The *Configdir* setting determines where the zone files are found; use *"Allzonefiles yes"* to load all zone files in that directory, or use "*Zonefile <file1>, <file2>, ...*" to load specific zone files.

Using the "zone" statement

This is the traditional method other DNS servers use as well. The advantage of this is that you have all the configuration in one file, so it is easily transferable or publishable. For primary zones, you will always need a master file so in that case this isn't really true.

Either way, the "zone" statement in your Posadisrc looks like this:

```
zone <type> <domain>
  command1 value11, value12
  command2 value2
  ...
```

Notice that the command lines are indented.

PRIMARY ZONE TYPE

The "primary" zone type is used by Posadis for normal primary authoritative DNS operation. A primary DNS server for a zone is a DNS server that has the zone data on disk itself.

Normally, you use primary zones implicitly by loading a master file and embedding the settings for the zone in it. In the Posadis Master File Editor, you can edit the settings for the zone by opening the "Zone configuration"

dialog. This will be discussed in greater detail in the next section.

If you want to define a primary zone directly in the Posadisrc, you'll need the *File* setting, that determines the master file (relative to the *Configdir*) for the zone, for example:

```
zone primary acdam.net
  file db.acdam.net
```

is the same as

```
zonefile db.acdam.net
```

Zone transfers

Zone transfers are used by secondary nameservers to get a complete copy of the zone. You can also use a query program like posask to show the complete contents of a zone by doing something like *poshost @server acdam.net axfr*.

By default, Posadis will provide a zone transfer to slaves for the zone only. You can either decide to allow zone transfers to anybody (which might take up significant resources) or tell Posadis who your slaves are (if you have any, anyway):

slaves - List of slaves (secondaries that will use you as the primary source for their zone data) for the domain name.

allow_xfr_to - List of additional clients to always allow zone transfers to. This is an address match list, so you can for example say "192.168.1.*".

allow_xfr - Can be either "any" (the default value) to allow zone transfers to anybody, "slaves" to allow only to slaves and those listed in allow_xfr_to, or "none" to only allow those on the allow_xfr_to list.

Note that clients in the slaves list differ from those in the allow_xfr_to list in that Posadis will send so-called "DNS Notify" messages to the slaves to notify them when the zone changes.

MASTER FILES AND RR TYPES

This section will describe the syntax and contents of master files.

Embedding configuration settings into master files

To specify configuration settings (for example the list of slaves; see the "Primary zone type" section), you can place them in your master files. Using the Master File Editor, edit the "Zone configuration" button, and enter, for example

```
slaves 212.142.29.65, 212.142.29.69
allow_xfr_to 127.0.0.1
```

or, if you want to edit your master file manually, use the ";set " prefix:

```
;set slaves 212.142.29.65, 212.142.29.69
;set allow_xfr_to 127.0.0.1
```

Resource Records

A master file is basically just a list of Resource Records (RRs). Each RR has a so-called *Time-To-Live*, which is the time the record may be stored in cache by resolvers. For example, if you surf to a domain name with an address record with a TTL of "1h" (1 hour), this means that if you visit the domain name again within one hour, you don't need to ask the server about the address again: you can consider your old data recent enough. If your records change often, you'll want a low TTL, whereas if your records are relatively static, you should make the TTL value higher.

The first RR of the zone should always be a SOA (Start of Authority) record, which contains basic zone information. Other RRs may follow. For a discussion on what records you need in your zone, I refer to the Posadis manual.

Resource Records supported by Posadis

Next comes a list of all Resource Records supported by Posadis. Key for the "syntax" column: "d" for a domain name, "i" for an IPv4 number, "s" for a 16-bit number, "l" for a 32-bit number, "c" for a character string, and "6" for an IPv6 address, and "m" for an e-mail address:

Record type - Syntax

Explanation

SOA - dmllllll

The SOA record, which is always the first record of your zone, contains general zone information used mainly by secondaries to determine their update policies. As you can see, the SOA record has six elements, making it the most difficult RR out there. Here are the entries:

MNAME - The domain name of the primary DNS server for the domain. Note that secondaries don't use this, so it has no practical use.

RNAME - E-mail address of the maintainer of the zone.

SERIAL - Serial number of the zone. You should increase this every time you change the zone. One common numbering scheme is to use the date for this, for example "2003042301" for the first change on 23 May 2003.

REFRESH - The time (in seconds, or in hours or weeks or something with a postfix: "2h" is two hours), after which a secondary will try to re-load the zone.

RETRY - If a zone transfer fails for some reason (e.g. the master was down), the secondary will retry after this interval.

EXPIRE - If, after this time, a secondary still hasn't succeeded in doing a new zone transfer, it will stop answering queries for the zone.

NTTL - TTL for negative caching: for example, the information that "wwww.acdam.net" doesn't exists, may be kept for this long.

An example might be something like:

```
@ 2h SOA ( ns1 meilof@acdam.net 2002052301
        2h 1h 1d 2h )
```

A - i

The A record is used to specify an IP address for a domain name. This is the traditional way of looking up an address for a specific service. An example could be as simple as:

```
www.acdam.net.      A       192.168.1.1
```

NS - d

This record lists DNS servers for your zone. Each DNS server should be listed by its domain name by means of a NS record. For example:

```
acdam.net.          NS      ns1.acdam.net.
```

CNAME - d

The CNAME record can be used to create an alias in the domain name system: the domain name the record is attached to is defined as an alias to the given target. For example, if we want www.acdam.net. to be an alias to athlon.acdam.net., we'd do something like:

```
www.acdam.net.      CNAME   athlon.acdam.net.
```

If for example an address query is received for www.acdam.net., the client will now receive the addresses for athlon.acdam.net. instead. Note that this also implies that no other records may be available for www.acdam.net., because those might be ambiguous with the data from athlon.acdam.net..

WKS - <other>

Well-known services. Rarely used anymore today, this RR gives information about the services running on a specific server. First, give the IP address the WKS RR applies to (this is nessecary because a domain name can have more than one IP number). After that, specify the protocol, usually "TCP" or "UDP", and after that, any services you're running on it, for example "http", "ftp", "smtp", and so on. Alternatively, you can give the port on which is being listened. For example:

```
www.acdam.net. WKS 192.168.1.1 tcp dns http ftp 8080
```

PTR - d

Used for reverse-mapping. Specifies the domain name this domain points to. See the reverse-mapping chapters in the reference section for more information. An example could be something like:

```
1.0.0.127.in-addr.arpa. PTR localhost.
```

HINFO - cc

This record is rarely used nowadays, but it can be used to give information about the system running for a given domain name. If you maintain a network, this might come in handy to keep track of your running operating systems. The record consists of two pieces of free-form text: at first one describing your hardware, followed by one describing your operating

system. If the description contains spaces, you'll need to enquote it. For example:

```
athlon.acdam.net.   HINFO     "AMD Athlon XP" "Debian GNU/Linux 3.0
"
```

MX - sd

This record is used to specify a mail server for a domain name. For example, if you send an email message to meilof@acdam.net, your mail delivery software will send an MX query for acdam.net to find out where to deliver mail. Apart from the domain name of the mail server, this record also contains a priority value. If there are multiple mail servers for a domain, the server(s) with the lowest priority will be tried first, and if they all fail, server with a higher priority will be tried. This can be used to specify backup mail servers:

```
acdam.net. MX 10 mail.acdam.net.     ; primary
           MX 10 mail2.acdam.net.    ; primary
           MX 50 fallback.acdam.net. ; fallback
```

Note that MX records are not used to lookup the pop3 or smtp servers of your ISP: normal A-type queries are used for these.

TXT - c[c...]

Can be used to put free-form text into your zone. Do with it whatever you like; these records aren't used in normal DNS operation. You can specify an arbitrary number of text strings here. For example:

```
meilof.acdam.net. TXT ( "Meilof Veeningen"
                        "Software developer"
                        "meilof@acdam.net" )
```

RP - md

Using the RP record, you can list someone who is responsible for the domain the record is attached to. It isn't really used often; you should only used it it you think you'll need it yourself some time. It has two arguments, namely the e-mail address of the person in charge, and a domain name which has TXT records attached to it, that can for example contain the telephone number of the person, or something like that. If you don't have such a TXT domain, use "." instead. An example might go thus:

```
www.acdam.net. RP meilof@acdam.net meilof.acdam.net.
```

AFSDB - sd

To be quite honest, I don't really know what this record does, but according to its documentation, it can be used to provide a distributed service under the domain name. This can either be an AFS cell database server or a DCE authenticated name server (whatever these two things might be). Either way, you can give the type of service (1 for the AFS server, and 2 for the DCE server), followed by the domain name of the server running the service. An

example might be:

```
acdam.net.          AFSDB     1 afs.acdam.net.
                    AFSDB     2 dce.acdam.net.
```

PX - sdd

Again, I haven't got a clue what this is, and quite frankly, I don't care much. This record is meant as a "pointer to X.400/RFC 822 mapping information", a name which by itself makes me not want to know more. Either way, Posadis supports it if you need it. The first value is a preference value, like for the MX record, the second is a RFC 822 domain, and the third is a the value of the X.400 data.

AAAA - 6

Similar to the A record, this record is used to specify an IPv6 address for a domain name. IPv6 is the successor of IPv4 which is expected to be the main internet protocol in about ten years. The representation of an IPv6 address is beyond the scope of this manual.

```
www.acdam.net.      AAAA      dead:beef::1
```

It is worth noting that there have been other proposals to specify IPv6 address in the DNS as well, most notably using the A6 RR type, which is not supported by Posadis (yet), and which is currently marked experimental by the Internet Engineering Task Force. As a result, AAAA might be replaced by a newer technology in the future.

SRV - sssd

This also is a relatively new Resource Record. It is used as a generic way to attach services to a domain. Whereas we usually use subdomains to describe the service for a domain name ("www.acdam.net" in practice really means we want the www server for acdam.net), this method isn't really powerful: it doesn't have load distribution, and it doesn't have the ability to redirect the client to a nonstandard port. For these goals, the SRV record was invented, that does have these features. The SRV record has the following syntax:

```
_service._protocol.domain SRV pri load port domain
```

As you can see, the domain name the SRV record is connected to is a bit odd. It does make sense when you think about it though: if you want to know where to locate the smtp service (which uses the tcp protocol) of acdam.net, you'd just do a SRV-type query for "_smtp._tcp.acdam.net", and you'd get back records describing the service. The SRV record has the following fiels

pri - Same as the MX record priority value. If a client wants to connect, it will at first try all servers with the lowest priority, and if they all fail, it will go to a higher level.

load - Determines load distribution within records of the same level. For example, if one server has a value of "20" and another has a value "80", the second server will receive 4 out of 5 queries.

port - The port the client should connect to.

domain - The domain name the client should connect to. The client will need to look up the addresses for this domain before it can connect.

We might thus set up our acdam.net domain like this:

```
_http._tcp.acdam.net.  SRV 0 80  80   www
                       SRV 0 20  80   slow-server
                       SRV 1 100 8080 backup

_smtp._tcp.acdam.net.  SRV 0 100 25   mail

_imap2._tcp.acdam.net. SRV 0 100 143  mail
```

Given how useful this SRV record could be, it is really a pity no-one seems to go into great trouble implementing it. I know no major servers that have SRV records, and I don't think there's much client software using it either, which poses kind of a chicken-and-egg problem. So I'd say: even if no-one uses SRV today, help the world and create SRV records anyway!

NAPTR - sscccdd

This record is used for something called "DNS ENUM" which has to do with mapping domain names to "E. 164" numbers. I don't know or seem to really care what this record does, but if you need it, it's there.

You can edit Resource Records by hand, or, more conveniently, using the Posadis master file editor. This editor has a built-in syntax check that can scan your zone for common configuration errors.

Master file syntax

This section will discuss the syntax of master files, would you want to edit them manually yourself rather than letting Mfedit do the dirty work.

Master files contain Resource Records in the following form:

```
<domainname> <ttl> <rrtype> <rrdata>
```

The entries in the syntax above are separated by one or more spaces or tabs. You can span a RR over multiple lines using a syntax like:

```
<domainname> <ttl> <rrtype> (
  <rrdata>
)
```

Note that you can also place other entries of the syntax on multiple lines; just as long as the first line contains a free-standing "(", and the last line contains a free-standing ")".

The following can for example be used to define an A record:

```
www.acdam.net.  2h  A   192.168.1.1
```

Lines that begin with ";" are ignored, and can thus be used as a comment. Also, text after ";' on a RR line is ignored. For example:

```
; next is the A record for www.acdam.net
www     2h A  192.168.1.1  ; this is it!
```

Shorter syntax

All domain names in master files should contain a trailing dot like in the previous example. If they don't have a trailing dot, they're considered relative. The origin they're considered relative to, is, by default, the zone root, so the domain name "www" in the acdam.net zone would be interpreted as "www.acdam.net.". You can change the origin by using the "$origin" directive, like this:

```
$origin google.nl.
```

The domain name specified in the $origin directive, itself, is always considered relative to the zone root. If you want to refer to the zone root, you can use "@" instead of the zone root name.

If you give multiple records for the same domain after each other, you don't need to give the domain name for each record. If the first character of a line is a space or a tab, the domain name of the previous record is used, for example:

```
www     2h      192.168.1.1
        2h      192.168.1.2
```

If no TTL is specified, the TTL of the first record in the master file is used. The first record should thus always have a TTL. Unless, that is, you use the "$ttl" directive to set the default TTL, for example:

```
$ttl 2h
```

SECONDARY ZONE TYPE

The "secondary" zone type, supported by the Posadis core, is used for zones that retrieve the zone data using a "zone transfer" from a master server. It has the following configuration options:

masters - List of DNS servers the server will attempt to get a zone transfer from. These servers will be tried in order. Also, the zone accepts notify messages, that tell the server to re-load the zone, from these masters.

allow_notify_from - Lists alternate sources (in address match list form) from which we trust NOTIFY messages. If zone data changes on a master, it sends out NOTIFY messages to its slaves to, well eh, notify them that something changed1. The slaves then know they should update the zone. Changing allow_notify_from is useful if the master is a multi-homed host. You can list only one IP number in the "masters" list for the zone, and list the others in the allow_notify_from list, so that the master is not tried twice for zone transfers, but its notify messages are always accepted by the secondary regardless of what IP address of the master they happened to come from.

Because secondary servers can also serve zone transfers themselves (this configuration is referred to as a "slave-master configuration"), the zone transfer-related configuration settings of the primary zone type (see the "Primary zone type" section) also apply to secondary DNS servers.

REVMAP MODULE DOCUMENTATION

The revmap module offers a convenient way to automatically do reverse and forward mapping of domain names for large IP namespaces. It can algorithmically generate answers to "in-addr.arpa" reverse-mapping queries by giving back a domain name generated from the IP number. On the other hand, it also offers a forward-mapping zone companion to convert these domain names back to IP numbers. This is really useful when you maintain a large site: some programs require any connnecting host to have valid reverse-mapping information, which can sometimes be quite a pain.

The "revmap" and "formap" zone types defined in this module support standard queries only. They do NOT support zone transfers, so secondary nameservers for these reverse-mapping domains will either also need to use this module, or be set up using a master file or other static data. Also, this module is for IPv4 only. Quite possibly, A "revmap6" module might be added in the future.

For example, a query for 1.1.168.192.in-addr.arpa might return a pointer to c0a80101.dyn.acdam.net., where a query for c0a80101.dyn.acdam.net. would return the address 192.168.1.1.

The "revmap" zone type

Maps back domain names in the "in-addr.arpa" namespace to domain names. Configuration options:

prefix - The domain name to append to the generated label. In our example, this was "dyn.acdam.net". This domain name should be a zone of the "formap" type to make forward queries work as well.

nslist - List of nameservers for the zone. Note that these nameservers may not be in the reverse-mapping zone itself, since the zone cannot give addresses for the nameservers.

ttl - Time To Live value for the records sent back.

The "formap" zone type

Companion to the "revmap" zone type. Configuration options:

nslist - See the "revmap" zone documentation.

ttl - See the "revmap" zone documentation.

Algorithm

The 8-byte domain label is the hexadecimal representation of the four bytes in the IP number, with all lowercase characters.

Known bugs and limitations

The forward mapping zone will answer any A query as long as it is a subitem of the zone root with exactly 8 bytes, even if it is not a valid hexadecimal value. In such cases, the returned address will be garbage.

The zones have only one, non-customizable algorithm for generating domain names. This might cause interoperability problems with other DNS servers offering similar capabilities but with a different algorithm.

LOCALHOST MODULE DOCUMENTATION

The localhost module automagically sends back the address of the client that sent the query to the DNS server. If you send a query to the server directly, it will thus send back your own IP number (so that you can find out which IP number you use for internet connectivity), but if you send a query to a recursive DNS server which will query for the information, it will find the IP number of the recursive DNS server.

So what's this good for, you ask? Well, I don't know really myself either. If you're in an identity crisis and you want to know who you are, this might help. Or, if you want to know whether your ISP's DNS server forwards it questions instead of answering them itself, this is just the tool. As you can see, this module is actually not quite useful at all. I got the idea at one point and decided I should give it a try, just because I was so excited with the modular design of Posadis. And after that, I just decided not to remove it, but rather to let it serve as an example of how to write a module that disciminates against clients (e.g. gives some clients back other answers than others).

The "localhost" zone type defined in this module supports standard queries only. It does NOT support zone transfers, so secondary nameservers for these "localhost" domains will also need to use this module, or use similar functionality, if any, in other DNS servers. Furthermore, the DNS server will answer queries for the root domain of the zone only. For other items, it will return NXDOMAIN. The module also supports IPv6.

The "localhost" zone type

Sends back the querier's address. Configuration options:

nslist - List of nameservers for the zone. Note that these nameservers may not be in the reverse-mapping zone itself, since the zone cannot give addresses for the nameservers.

ttl - Time To Live value for the records sent back.

MONITOR MODULE

The Monitor module of Posadis can automatically monitor a directory containing DNS master files and/or zone data files. It will automatically load all files in the directory, reload them when they change, and remove the zones when you remove the files. This eliminates the need to do a SIGHUP whenever you change a zone. The fam module is particularly useful if you run a large amount of relatively small zones.

To use the Monitor functionality, add the following to your Posadisrc:

```
LoadModule "fam"
```

For a discussion on how to use the Monitor module in your configuration,

refer to the "Adding zones".

On Unix, the Monitor module depends on the availability of the File Alteration Monitor program by SGI. For more information, visit:

http://oss.sgi.com/projects/fam/

If you install Posadis from binary packages, you might need to install the Monitor module separately.

posadis

Updated: 0.60.5

NAME

Posadis - A modular, portable DNS server

USAGE

posadis [*options*]

-c configfile - Specifies the location of the Posadis configuration file. By default, Posadis will use /etc/posadisrc.

-u user - Specifies the user to switch to after starting up (Unix only).

-g group - Specifies the user to switch to after starting up (Unix only).

-f - If specified, fork after starting up (Unix only).

-l logfile - File to put logging information into.

-p pidfile - File to write PID to.

-r chrootdir - Directory to chroot() to after reading config file (requires root privileges; Unix only).

--help - Print usage information

--version - Print version information

DESCRIPTION

Posadis is an open-source Domain Name Server. It features a modular, extensible structure, standards compliance and good karma.

The Posadis homepage can be found at http://www.posadis.org/. On that website, you'll also find the full Posadis documentation. For more information on the Posadisrc file, consult the *posadisrc(5)* manual page.

Posadis is based on the Poslib library. For more information, goto http://www.posadis.org/poslib/.

SIGNALS

Currently, Posadis only supports these signals (beyond SIGTERM and SIGKILL, that is):

- *SIGUSR1* - Toggles the Posadis query logging feature
- *SIGUSR2* - Dumps the Posadis cache to /tmp/posadis-cachedump

Posadis will probably support SIGHUP to re-load its configuration soon in the future.

You can find out the PID of the Posadis process by running "/etc/init.d/posadis status" (or wherever the Posadis init script is located). You can toggle query logging using this init script as well, by running

/etc/init.d/posadis togglequerylog
(optionally, you can enter "tql" as a short notation for "togglequerylog")

FILES

/etc/posadisrc

poshost

Updated: 0.60.5

NAME

poshost - Simple query program producing human-readable output

SYNTAX

poshost [@server] [domainname] [querytype]

DESCRIPTION

This is a simple console query program, that gives it answers in a simple, human-readable form, quite unlike the raw DNS messages shown by its big brother, Posask. The result is displayed in the same way the graphical "DNS querier" tool shows its answers, e.g.:

```
meilof@linux> poshost @216.239.32.10 google.nl
Querying 216.239.32.10#53 for {google.nl.,A}
Received answer from 216.239.32.10#53:
  Authoritive answer

Answers for google.nl.:
  -> [A] 216.239.39.100
  -> [A] 216.239.37.100
```

OPTIONS

<*server*> - Server to query

<*domainname*> - Domain name to query.

<*querytype*> - Query type (e.g. a, ptr, mx, any)

If no server is given, the DNS server the system resolver uses is used (Unix), or the DNS server at localhost (Windows). If no domain name or type is given, a query for {.,NS} is assumed. If no query type is given, "a" is assumed unless the domain name is in one of the reverse-mapping zones such as in-addr.arpa. In that case, "ptr" is the default query type.

EXAMPLES

In its simplest form, poshost prints out the root nameservers:

```
poshost
```

To find out the addresses of www.posadis.org, do:

```
poshost www.posadis.org
```

Or, to query another nameserver:

```
poshost @192.168.1.102 www.posadis.org
```

If we want to find the posadis.org mail servers:

```
poshost posadis.org mx
```

If we want to find out the domain name of 192.168.1.1, perform one of the following, equivalent queries:

```
poshost 1.1.168.192.in-addr.arpa
poshost .192.168.1.1
```

posask

Updated: 0.60.5

NAME

Posask - Send queries to DNS servers

SYNOPSIS

posask [@server[#port]] [option1=value1 ...] [qname] [qtype]

DESCRIPTION

Posask is a tool to query Domain Name Servers. Similar to dig and askmara, this tool can interactively query any RFC-compliant nameserver. Posask supports all RR types Posadis supports, as well as zone transfers.

The settings on the command line are:

@[server][#port] - Specify the server to query. By default, the system resolver (the resolver that is usually used to look up domain names) is used. If you give no port, port 53 will be used; if you don't specify an IP number (e.g. @3000), the given port at localhost will be used. You can also give an IPv6 number here, if your operating system supports that.

qname - The domain name to query. If you don't specify one, Posask will do a query for {.;NS} instead. If you want to do reverse-mapping, you can use the special dot notation for reverse-mapping domain names, e.g. ".192.168.1.1" is equivalent to "1.1.168.192.in-addr.arpa".

qtype - The query type. Any RR type supported by Posadis (e.g. a, mx, soa), any for all RR types, axfr for a complete zone transfer, or ixfr/<serial> for an incremental zone transfer. If you don't specify a query type, Posask will default to A for normal domain names, and to PTR for reverse-mapping domain names, such as subdomains of in-addr.arpa.

See the *OPTIONS* section for available options. You can specify them on the command line using option=value

OPTIONS

rd - Specifies whether we want the DNS server to do recursion.

use_tcp - Specifies whether to use TCP. By default, Posadis uses TCP for IXFR and AXFR, and UDP for normal queries.

tries - Specifies the number of times to retry the query (3 by default).

timeout - Specifies the time in seconds to wait for an answer (5 seconds by default). Note that retries of queries happen within this timeout range, so this timeout value is actually the maximum time you will wait until you get an answer to your query.

posadis-getroots

Updated: 0.60.5

NAME

posadis-getroots - Retrieve root nameserver list to put in your "posadisrc"

SYNTAX

posadis-getroots [icann|<root-server-provider>]

posadis-getroots <server1> <server2> ...

DESCRIPTION

This program retrieves an up-to-date list of the root nameservers you can put into your Posadisrc. It has a hardcoded list of root nameservers, which it will try to reach in order to find the most recent information.

The program can, at your choice, either query the standard, Icann (www.icann.org) root nameservers, or another root server organisation. To try one, type its name, or, to enumerate all available root nameserver sets, use a nonexistent root server set, for example, *none*. Alternatively, you can specify your own DNS server that should be queried.

The resulting list of root nameservers will be printed to standard output in a form suitable for inclusion in the *posadisrc* file for Posadis. If an error occurs, an error message will be printed and the program will return with a nonzero exit code.

OPTIONS

icann - Query the Icann root nameservers

<other_name> - Query another root nameserver provider

<server> - Query another root nameserver

FILES

/etc/posadisrc