The Petidomo Mailing List Manager

Peter Simons <simons@cryp.to>

2019-07-10

Contents

Chapter 1

Installing Petidomo

The installation of the Petidomo Mailing List Manager is simple and straight forward; do not be scared by the length of this chapter. There are many different ways and options how to install it and I have tried my best to cover *all* of them. If you are not interested in every little detail, you will be able to skim over most of the text here.

1.1 Getting it

The Petidomo Mailing List Manager is available or download via Sourceforge from `http://sourceforge.net/projects/petidomo/files/`. Users of the version control system Git can also clone the public repository from `https://github.com/peti/petidomo`.

1.2 Building the Binaries

Untar the source archive of Petidomo in a directory of your choice like /usr/local/src or your home directory. This will create a directory called petidomo-VERSION, where the "VERSION" part is called exactly as in the file name of the tar archive. Change into this directory.

Now you have to run the configure script

```
./configure
```

which will determine the characteristics of your system and create the files required to actually build Petidomo. You may provide several parameters to the script. The interesting ones, including the default values if unspecified, are:

--help Display the complete list of command line options.

--prefix The PREFIX for all following paths. The default is /usr/local.

--exec-prefix Set the EPREFIX for all following paths. This is useful in case you want to install binaries into a different directory hierarchy than normal text files, but usually the EPREFIX is identical to PREFIX. The default is PREFIX.

--bindir Set the directory where the binaries should be installed. The default is EPREFIX/bin.

--libexecdir Set the directory where executables should be installed that will be called by Petidomo but not by the user directly (like posting filters). The default is EPREFIX/libexec.

--datadir Set the directory where read-only architecture-independent data files should be installed (like the help file). The default is PREFIX/share.

--sysconfdir Set the directory where read-only configuration files should be installed. The default is PREFIX/etc.

--localstatedir Set the directory where modifiable data files should be installed (like the approve-queue or the mailing list config files). The default is PREFIX/var.

--mandir Set the directory where man documentation files should be installed. The default is PREFIX/man.

Please note that the directories you specify here are only the default settings that are compiled into Petidomo. You can modify *all* paths at run-time via the command line and through the configuration files. So don't waste to much time figuring out what you want here, you can change anything later without having to recompile Petidomo.

Finally, here is an example output of the configuration script when run without any parameters on a Linux machine:

```
simons@peti:~/projects/petidomo-4.0b1$ ./configure
Configuring OSSP Petidomo, Version 4.0b1 (18-Jan-2001)
creating cache ./config.cache
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
checking whether the C compiler (gcc  ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for ranlib... ranlib
checking for flex... flex
checking for yywrap in -lfl... yes
checking for bison... bison -y
checking size of unsigned short... 2
checking size of unsigned int... 4
checking size of unsigned long... 4
checking how to run the C preprocessor... gcc -E
checking for ANSI C header files... yes
checking for ssize_t... yes
updating cache ./config.cache
creating ./config.status
creating Makefile
```

Often, you may want to pass certain flags to the compiler or the linker to modify the building process. To achieve this, you can set certain environment variables before calling the configure script. These variables are:

CC The name of the C compiler to use.

CPPFLAGS Flags to pass to the preprocesser before compiling a source code file.

CFLAGS Flags to pass to the compiler when compiling a C source code file.

LDFLAGS Flags to pass to the linker when linking the binaries.

I personally find this useful to raise the level of compiler optimization or to add linker flags that tell the linker to strip unnecessary symbols from the binaries. To achive these effects, I call the configure script like this:

```
CFLAGS=-O3 LDFLAGS=-s ./configure
```

Anyway, once the configure script has been run, just call

```
make
```

to start the building process. Petidomo has been tested with various flavours of the make utility and all of them seem to work fine. If in doubt, try GNU Make, which is available from ftp.gnu.org.

Petidomo has also been built using parallel builds. This is useful if you have a multi-processer system. You can do this with most make utilities by adding the flag "-j4" with "4" being the number of processes you want to spawn simultaneously. Please note, though, that some make utilities have problems with the rules that translate the yacc-modules included in Petidomo correctly when building in parallel. If you experience any trouble, just build it conventionally and you should be fine.

1.3 Installing the Binaries

To install the software to your system, all you have to do is execute

```
make install
```

This will copy the Petidomo binary, the posting filters included in the distribution, the sample config files and the manual pages into the directories you chose at configure time earlier. If you're a first-time user, you may also want to execute

```
make install-testlist
```

which will create a sample mailing list called "testlist" for you.

Assuming you used the default paths when running configure, the install process will create the follwing directories, respectively copy the following files to your system:

```
/usr/local/
/usr/local/bin/
/usr/local/bin/petidomo
/usr/local/bin/petidomo-approve
/usr/local/bin/petidomo-kickout
/usr/local/etc/
```

```
/usr/local/etc/petidomo.acl-sample
/usr/local/etc/petidomo.conf-sample
/usr/local/libexec/
/usr/local/libexec/petidomo/
/usr/local/libexec/petidomo/insert-name-in-subject.sh
/usr/local/libexec/petidomo/pgp-decrypt.sh
/usr/local/libexec/petidomo/pgp-encrypt.sh
/usr/local/libexec/petidomo/rfc2369.sh
/usr/local/man/
/usr/local/man/man1/
/usr/local/man/man1/petidomo.1
/usr/local/share/
/usr/local/share/petidomo/
/usr/local/share/petidomo/help
/usr/local/var/
/usr/local/var/petidomo/
/usr/local/var/petidomo/ack-queue/
/usr/local/var/petidomo/index
/usr/local/var/petidomo/lists/
```

If you run the "install-testlist" target, the following directory/files will be created additionally:

```
/usr/local/var/petidomo/lists/testlist/
/usr/local/var/petidomo/lists/testlist/config
/usr/local/var/petidomo/lists/testlist/acl
/usr/local/var/petidomo/lists/testlist/list
```

1.4 Configuring Sendmail

Before you can use Petidomo, you have to configure sendmail so that it knows about Petidomo — I assume that you have sendmail installed already. If you are using an MTA other than sendmail, you are on your own from here on, I am afraid. Any users who have successfully installed Petidomo on a qmail-, vmailer-, or postfix-based system are more than welcome to contribute to this documentation to help other users.

To run Petidomo via sendmail — what is what you want to do —, you have to create apropriate aliases for it. You can do this by adding the folling lines to your aliases file, which usually resides in /etc/aliases or, with newer sendmail versions, in /etc/mail/aliases:

```
petidomo-manager:   postmaster
petidomo:           "|/usr/local/bin/petidomo --mode=listserv"
petidomo-approve:   "|/usr/local/bin/petidomo --mode=approve"
```

In case you installed the Petidomo binary to some other location, you will have to change the paths here apropriately of course. You may also chose that mail for the "petidomo-manager" should go to some different address than "postmaster", if that suits your needs better; the main point is that somebody actually *reads* what arrives there.

If executed the "install-testlist" target earlier and thus have the example mailing list from the distribution installed, you may also want to add the lines:

```
testlist:           "|/usr/local/bin/petidomo --mode=deliver testlist"
testlist-request:   "|/usr/local/bin/petidomo --mode=listserv testlist"
testlist-owner:     petidomo-manager
```

Having done all this, execute the `newaliases(1)` utility to rebuild sendmail's internal database. Your changes will not have any effect unless you do this.

1.5 Configuring the File Permissions

The final step, before Petidomo is successfully installed, is to set the right permissions to the installation directories and installed files. Unfortunately, the installation process cannot do this automatically; you have to chose what permissions are "right" yourself. If works like this: Petidomo will be called from sendmail, thanks to the aliases you just created. That means, that sendmail determines under what user to start Petidomo. In 99% of all configurations I have ever seen, that user is "daemon", but it *may* be something else, so we better figure it out for sure.

Add the line

```
foobar:             "/tmp/foobar-mail"
```

to your aliases file and execute newaliases(1). Then send an e-mail to the address "foobar". The contents of this mail will be stored in the file /tmp/foobar-mail then and we are interested in the user who owns this file:

```
root@peti:/# sendmail -v foobar </dev/null
foobar... aliased to "/tmp/foobar-mail"
"/tmp/foobar-mail"... Sent
root@peti:/# ls -l /tmp/foobar-mail
-rw-------   1 daemon   daemon        269 Feb 12 17:57 /tmp/foobar-mail
```

See? On my system it is "daemon" indeed. On your system it may be someone else. Now that we know, you may remove the foobar-line from the aliases file again.

OK, sendmail starts Petidomo under user id "daemon". This means that "daemon" must have read access to virtually any file in the Petidomo installation. This is the default, because all files are installed with read permisson for everybody. Also, all directories allow access to anybody by default. But "daemon" also needs write access to the "localstatedir" — /usr/local/var/petidomo per default. You can ensure this by executing the command:

```
chown -R daemon /usr/local/var/petidomo
```

This is a rather simplistic solution to the permisson problem; you *can* use much more fine-grained settings if you like to. But I figured that if you are the kind of person who wants to do things like this, you won't need an explanation how to do it anyway. Just that much information for you: Petidomo does not actually write to the "localstatdir", but only to the subdirectory ack-queue located in it.

Of course, you do not necessarily need to have the ack-queue directory owned by "daemon", you can also set the group permissions apropriately. Furthermore, Petidomo will usually want to write to the lists directory located in the "local-statedir", because most list administrators tend to place the mailing list archives there, but you can enable write access according to the list's configuration once you know how you're mailing lists are configured. In case something does not work as expected, check out the syslog messages for the LOG_MAIL facility — this is where Petidomo logs its error messages.

1.6 Configuring Petidomo

The last step before we can test our installation is to configure Petidomo. This is really simple. List the contents of the "sysconfdir" you chose. If you did not change the default paths, this is /usr/local/etc. There you will find two files: petidomo.conf-sample and petidomo.acl-sample. Just rename them to petidomo.conf and petidomo.acl respectively and fire up your favorite text editor to edit the file petidomo.conf.

Uncomment the options "Hostname", "AdminPassword", and "MTA" and set the values correctly. "Hostname" should be the fully qualified domain name of the machine running Petidomo. It is essential that this name can receive e-mail, that is, that is has an MX record. (Talk to the person administrating the domain name service of your organization if this doesn't make any sense to you.) As "AdminPassword", you can chose pretty much any text you like, just make sure you remember it. The "MTA" setting will usually be alright the way it is. You may want to check whether sendmail does actually live at this path; on some Unixes, it is not installed at /usr/sbin/sendmail, but at /usr/lib/sendmail. Change the setting if this is the case. You can ignore all other settings right now. Come back and configure those once you have read the apropriate sections of this manual. If you're an experienced Unix wizard, the comments in the config file will probably be enough for you to guess what these options do, though.

Once you have done this, your installation is ready to be tested.

1.7 Testing the Installation

Asserting you followed all steps described above, you have a working Petidomo installation now. Occasionally, some minor permisson problem may still remain to be fixed, or you may want to customize some texts. To figure out what is left to do (or to realize that there is nothing left do to), send an e-mail to the "petidomo" user on your machine and put the word "help" into the mail body — without the quotes of course.

On my system, this looks like this:

```
simons@peti:~/projects/petidomo$ echo help | sendmail -v petidomo
petidomo... aliased to "|/usr/local/bin/petidomo --mode=listserv"
"|/usr/local/bin/petidomo --mode=listserv"... Connecting to prog...
"|/usr/local/bin/petidomo --mode=listserv"... Sent
```

Once you sent the e-mail, sendmail will start up Petidomo and feed the mail text into it for processing. If you take a look at the syslogfile containing the

LOG_MAIL facility now — this is usally /var/log/messages or /var/log/maillog —, you will find that Petidomo logged entries there that look pretty much like the following ones. The backslash ("\") characters at the end of some of these lines denote that the line has been wrapped for readability. In reality, this would be one single large line.

```
sendmail[8705]: f1CIHWJ08705: from=simons, size=5, class=0, \
        nrcpts=1, msgid=<200102121817.f1CIHWJ08705@peti.cryp.to>, \
        relay=simons@localhost
petidomo[8706]: Petidomo 4.0b1 (18-Jan-2001) starting up; \
        mode=listserv, listname=<none>, \
        masterconf=/usr/local/etc/petidomo.conf, \
        approved=false, ruid=2, euid=2, gid=2, egid=2
petidomo[8706]: simons@peti.cryp.to: help
sendmail[8707]: f1CIHX508707: from=petidomo-manager@peti.cryp.to, \
        size=2091, class=-100, nrcpts=1, \
        msgid=<200102121817.f1CIHX508707@peti.cryp.to>, \
        relay=daemon@localhost
sendmail[8705]: f1CIHWJ08705: \
        to="|/usr/local/bin/petidomo --mode=listserv", \
        ctladdr=petidomo (2/0), delay=00:00:01, xdelay=00:00:01, \
        mailer=prog, pri=30005, dsn=2.0.0, stat=Sent
sendmail[8709]: f1CIHX508707: to=simons@peti.cryp.to, delay=00:00:00, \
        xdelay=00:00:00, mailer=local, pri=212091, dsn=2.0.0, stat=Sent
```

As you can see, Petidomo logged how it was started, where it is expecting its master config file and under which user- and group id it is running. Then it logs that it has received a HELP request. This request will be answered by sending the file /usr/local/share/petidomo/help back to the person who requested help, and if everthing worked, you will now find that mail in your mail box.

If something went wrong, Petidomo will tell you what went wrong. So, please fix the problem and try again. In 99% of all cases, the error will say something like "opening file XYZ failed: permission denied". Then all you have to do is to make sure that the user under which Petidomo has been started (you have the numeric id in the logfile) has read access to that file. If the user has but Petidomo keeps complaining, check, whether that user has access to the directory at all!

Those of you who executed the "install-testlist" target earlier in the "Building the Binaries" chapter may also want to test whether this mailing list is working. To do so, send another mail to Petidomo and put the command "subscribe YOUR-ADDRESS testlist" in the mail body — without the quotes! "YOUR-ADDRESS" naturally means that you should insert your e-mail address here. This command will subscribe your e-mail address to the "testlist" mailing list; you should receive a confirmation about that via e-mail within moments. Once that is accomplished, send another e-mail to the "testlist" address on your system. The e-mail may look like whatever you want.

Within seconds, you should get the mail back from the mailing list server — and so will all other addresses that are subscribed to the list. My personal test mail looked like this:

```
From testlist-owner@peti.cryp.to  Mon Feb 12 19:43:56 2001
```

```
Received: (from daemon@localhost)
        by peti.cryp.to id f1CIhuA08872 for simons@peti.cryp.to;
        Mon, 12 Feb 2001 19:43:56 +0100
Received: (from simons@localhost)
        by peti.cryp.to id f1CIhJY08853 for testlist;
        Mon, 12 Feb 2001 19:43:19 +0100
Date: Mon, 12 Feb 2001 19:43:19 +0100
From: Peter Simons <simons@peti.cryp.to>
Message-Id: <200102121843.f1CIhJY08853@peti.cryp.to>
Subject: Petidomo absolutely rules the known earth!
Reply-To: testlist@peti.cryp.to
Sender: testlist-owner@peti.cryp.to
Precedence: list

It does ...
```

If this all worked for you, you have a your Petidomo installation up and running. Congratulations!

Chapter 2

Configuring Petidomo

2.1 Configuration File Syntax

All configuration files in the Petidomo-package, have the following format:

```
keyword         parameter
```

The "keyword"-part must start at the first column of the line and is followed by one or several blanks or tabs. The first non-blank character then is interpreted as the parameter for this keyword. The following line, for example:

```
Hostname        petidomo.is.great
```

will tell Petidomo that the name of the machine it is running on is called "petidomo.is.great". If the parameter contains any blanks, what is not very likely for a hostname, but may happen with other settings, you should enclose it in double quotes, like this:

```
AdminPassword   "open sesame"
```

Quoting the parameter is not strictly necessary, though, Petidomo's config file parser will get it right anyway. You only have to quote the parameter, if it contains blanks as first or last character, what is rather unlikely to happen.

Furthermore all empty lines are ignored. So are lines that start with a '#' sign. You can use this for writing comments for the reader into the config file.

2.2 The Master Configuration File

The following keywords are recognized in the master config file.

Hostname "hostname.domain.name"

This entry specifies the fully qualified domain name of the machine, Petidomo is running on. A fully qualified domain name is the hostname of the machine with the domain name appended with a dot. The following, for example:

```
HostName        listserver.foo.bar
```

would be a valid statement. Normally this option has been set by the install script correctly already.

The name you set here is not necessarily the name, Petidomo will use when delivering mailing list-postings to the subscribers, or when answering requests, because you can specify a different fully qualified domain name for every mailing list you host. This is known as *virtual hosting*.

This option is *required*. Petidomo will abort with an error, if the master config file doesn't set it.

AdminPassword "password"

This tag sets the master password, which authenticiates the Petidmo administrator. Here is an example:

```
AdminPassword   "open sesame"
```

Normally this option has been set by the install script already.

Please chose this password carefully. Knowledge of the master password will enable you to access *all* mailing lists running on this system.

Passwords are compared case-insensitively. That means, that the passwords "Open SESAME", "open sesame" and "OPEN seSAme" are all the same.

This option is *required*. Petidomo will abort with an error, if the master config file doesn't set it.

MTA "/path/to/executable"

The MTA tag tells Petidomo which mail transport agent should be used to deliver outgoing emails. Normally this option has been set by the install script already, so you don't need to worry about this anymore.

An example setting is:

```
MTA     "/usr/sbin/sendmail"
```

but Petidomo will run fine with other mail transport agents, too. So far, the system has been tested with the Allman sendmail, SMail and qmail without any problems.

This option is *required*. Petidomo will abort with an error, if the master config file doesn't set it.

MTAOptions "string"

This tag is a bit tricky and in ninety-nine out of hundred cases you should simply leave this option undefined as it is rarely required anyway.

This entry sets the options which will be handed over to the MTA when it is called. The following example

```
MTAOptions "-odq -i -f%s"
```

will yield a call "<MTA> -odq -i -f<envelope>". The '%s' is replaced with the envelope the mail should be sent under.

Adding options to the execution call of the mail transport agent can be useful to enable or disable certain features for mailing lists only, while leaving them on for all other mail. The '-odq' setting is a fine example. This parameter will tell the Allmann sendmail to queue all mail, instead of trying to deliver it immediately.

ListDirectory "/path/to/directory"

Here you can tell Petidomo the path to the directory where the mailing list config file reside. The compiled-in default is "LOCALSTATEDIR/petidomo/lists" — in most cases that resolves to /usr/local/var/petidomo/lists. When Petidomo tries to find the configuration of list, say, "foobar", it will look for any of the following files in this directory: foobar.conf, foobar.config, foobar/conf, or foobar/config.

AckQueueDirectory "/path/to/directory"

This tag will tell Petidomo where to store files that need to be queued for later processing — for example subscribe requests that need to be acknowledged by the user before they'll be carried out. The default location is "LOCALSTATEDIR/petidomo/ack-queue"; unless you changed the default path at compile time, this will resolve to /usr/local/var/petidomo/ack-queue. Please note that Petidomo will need permission to write to that directory in order for things to work.

HelpFile "/path/to/file"

This tag will tell Petidomo where to find the text file that will be sent back to a user sending in a "help" command. The default location is "DATADIR/petidomo/help"", what is usually the file /usr/local/share/petidomo/help.

ACLFile "/path/to/file"

This tag tells Petidomo the path to the system-wide ACL file. The default location is "SYSCONFDIR/petidomo.acl", what usually resolves to /usr/local/etc/petidomo.acl.

IndexFile "/path/to/file"

Similarly to "HelpFile", this tag will tell Petidomo where to find the text file that will be sent back to a user requesting the server's "index". The default location is "LOCALSTATEDIR/petidomo/index"; unless you changed the default path at compile time, this will resolve to /usr/local/var/petidomo/index.

2.3 List Configuration Files

While the master config file sets options which are relevant for the Petidomo package as a whole, the list config file sets options which are valid only locally for the mailing list. The supported keywords are as follows.

ListType "open", "closed", "moderated", "acknowledged", or "acknowledged-once"

This setting tells Petidomo who is allowed to post to the mailing list. On an "open" mailing list, everybody is allowed to post, whether he's subscribed or not. On a "closed" mailing list, only subscribers are allowed to post. That means that only mails coming from an address found on the list's address database will go through. All other mails will be sent back to the person trying to post with the comment that he should subscribe first.

Please note that a closed list may not do exactly what you want, because when a person is subscribed to a list as "example@address.net", but tries to post from a different account than that one, Petidomo will not let him post. It tries to recognize this case as far as possible: For example, it doesn't matter whether you are posting from "address@host1.address.net" or "address@host2.address.net", Petidomo will handle that. But if the article comes from "example@private.account", it will be rejected, even though the sender might be a valid subscriber. It depends on the subscribers of the mailing list, whether this is a problem or not.

A better way to keep spam off your lists may be the "acknowledged" or "acknowledged-once" type of list. The former list mode means that every time someone tries to post, he will get a short reply back which contains some cryptographic cookie. Furthermore the mail will tell him to please reply to that mail and send the cookie back to Petidomo. Once that is accomplished, Petidomo will let the posting pass.

This means that only people will be able to post that have a valid envelope or from address — something spammers usually do not. Since the request for confirmation never reaches them, their postings will not go through. Everybody else can acknoledge the posting by sending the cookie back and thus get by that hurdle.

The difference between the "acknowledged" and "acknowledged-once" mode finally is that in the latter mode, people have to send that confirmation back only *once*, while in the first mode, they have to send it in every time they're trying to post. Petidomo will keep a database of addresses that have been confirmed by that mechanism and these addresses are allowed to post from now on without having to confirm their posting again. Please note that this list of addresses is not related to the list of subscribers!

Last but not least, there is a mode called "moderated"; in this mode, nobody is allowed to post unless he can provide the correct posting- or administrator password for the list.

If this option is unset, the default is to run an open list.

SubscriptionType "public", "admin", or "acknowledged"

This option specifies who may subscribe to a mailing list. On an "open"-subscription list, anybody may subscribe any address to the list. If the subscription type is "admin", only the person knowing the admin password of the mailing list is allowed to subscribe or unsubscribe addresses. All other requests that aren't properly authenticated will be forwarded to the list owner for approval.

If the subscription type is set to "acknowledged", anybody can issue a subscribe command, but the address will not be added to the list (or removed from the list in case of an "unsubscribe") unless the request has been confirmed: Petidomo will send a random cookie to the address in questing and ask to send the cookie back to acknowledge the request. Once the cookie has been send back, the request is carried out.

This mode is useful to prevent people from adding random addresses to the list or to prevent them from removing other people from the list without their consesus.

If this option is unset, the default to allow public subscription.

AllowMembersCommand "yes" or "no"

Petidomo knows a command "members" or "who", which can be sent to the server and it will reply with the complete list of subscribed addresses for the mailing list. This may be useful for list administrators, but it can be abused easily by spammers, to collect addresses where to send their unsolicted commercial e-mail to.

Furthermore, with certain mailing lists it may be undesirable that one can see "who else" is subscribed to that list. That's why this option has been added. If you set it to "no", the "members"-command will be diabled for this list. (This is also the default if the option is not specified in the config file.)

If you set it to "yes", the "members"-command will work.

Hostname "hostname.domainname"

This options tells Petidomo to use this hostname for the mailing list, instead of the one configured in the master configuration file. This feature is useful to do virtual hosting.

Virtual hosting is required when several mailing lists run on the same server, but they have to look like they would coming from different machines. Let's use an example: The internet service provider "Inter.Net" offers its customers to host mailing lists for them. A small software house of the name "Petiware" wants to provide a mailing list for all its customers, but they don't have a dedicated Internet line.

So they use the service provided by Inter.Net and let them host the mailing list on their machine. The mailing list server at Inter.Net has the fully qualified domain name "mail.inter.net". Petiware, though, wants the list to run under the name "customers@petiware.com" and *not* "customers@inter.net" — what would a be misleading.

So all the mailing list guru from Inter.Net has to do is to set the entry

```
Hostname        petiware.com
```

in the config file of the "customers" mailing list. Petidomo will now use the hostname "petiware.com" in all mails that are posted to that list, instead of "mail.inter.net".

You can specify a different hostname for every mailing list, using this feature. *That* is "virtual hosting". Further details on virtual hosting can be found in section 5.3 of the user manual.

If this entry is unset, the name configured in the master config file will be used as hostname for this mailing list.

AdminPassword "string"

This tag sets the master password, which authenticiates the administrator of this mailing list. The administrator has special priviledes, such as deleting other users, overriding access control restrictions or un-/subscribing users to closed mailing lists. This is described briefly in section 4 of the user manual.

Please note that passwords are always case-insensitive. It is also worth noting that the master password is always valid as administrator password for the list, also.

Leave this entry blank, if you don't want to enable remote administration of the mailing list.

PostingPassword "string"

This tag sets the "posting password". The posting password allows to post an article to a moderated mailing list, but it does not allow any administration of the list itself. On lists that are of a different type than moderated, setting a posting password does usually not make any sense and you can leave this entry unset.

ReplyTo "email@address.net" or "none"

This tag controls the 'Reply-To:' field, which Petidomo adds to posted articles before it is delivered to the recipients. Using this option, you can force Petidomo to insert a 'Reply-To:' which points to a certain address. On a moderated list, for example, you can set this as follows:

```
ReplyTo          moderator@address.net
```

to direct all replies to the posting to the moderator again, regardless of what address is noted in the 'From:' line of the mail.

If you set "none", Petidomo will not add a 'Reply-To:' header at all.

If this option is unset, Petidomo will to insert a 'Reply-To:' header that directs replies back to the mailing list, so that subscribers can conveniently post simply by hitting the 'reply' function in their mail reader.

PostingFilter "bourne shell command"

If you specify a posting filter, this program or script will be started by Petidomo before it sends a posting out to the subscribers. The programm will receive the article, as it has been prepared by Petidomo, on standard input and is expected to write the final version of the mail to standard output. The posting filter can be used to manipulate the headers for special purposes.

An example for a postin filter that wouldn't modify the mail at all is the following:

```
PostingFilter   /bin/cat
```

A detailed discussion of posting filters can be found in section 5.1 of the manual.

If the filter program exits with a returncode not equal to 0 (zero), Petidomo will not post the article and terminate.

Archive "/path/of/archive"

If this option is set, Petidomo will archive all articles that have been posted on that mailing list. The parameter for this tag may either be the name and path of a file or of a directory. The path may either be absolute (/var/archive/list) or relative (archive). For relative paths, the directory where the list's config file resides will be used as starting point. If the "Archive"-tag points to a file, Petidomo will append every posted article to that file. If points to a directory, each posting will be stored in that directory in a separate file.

If this option is unset, posted articles will not be archived at all.

For further information an creating mailing list archives, please refer to section 5.4 of the user manual.

IntroductionFile "/path/to/file"

This tag specifies the path to the so called "introduction" file. Every time an address is added to the mailing list, Petidomo will send the contents of this file to the new subscriber. This is meant to be used to inform the new subscriber about the list's topic, habits he should know, etc. If the file does not exist, no mail is sent out.

If the path specified here is relative — not starting with a "/" character that is —, it is interpreted to be relative to the directory where the list's config file has been found. The default path is introduction.

DescriptionFile "/path/to/file"

This tag specifies the path to the so called "description" file. This file is supposed to contain a short description of the mailing list's topic and purpose. It's contents will be sent back if a user requests the command "help listname".

If the path specified here is relative — not starting with a "/" character that is —, it is interpreted to be relative to the directory where the list's config file has been found. The default path is description.

ACLFile "/path/to/file"

This tag specifies the path to the list-specific ACL file. Please refer to section 4.5 for more information about the access control language of Petidomo.

If the path specified here is relative — not starting with a "/" character that is —, it is interpreted to be relative to the directory where the list's config file has been found. The default path is acl.

HeaderFile "/path/to/file"

The contents of this file this tag points to will be added to the header of *every* posting on this list. This may be used to add custom headers like:

```
X-List-Archive-is-at: http://list-archive.example.org/
```

Please note that the contents of this file will be added *verbatim*! So don't include any empty lines in here as empty lines mark the end of the mail headers! Generally, please use this feature with care; most mailing list administrators tend to overestimate the importance of custom headers on their mailing list.

If the path specified here is relative — not starting with a "/" character that is —, it is interpreted to be relative to the directory where the list's config file has been found. The default path is header.

SignatureFile "/path/to/file"

The contents of this file this tag points to will be appended to *every* posting on this list. This may be used to add a list-specific signature, like:

```
--
 Useful comment here.
```

If the path specified here is relative — not starting with a "/" character that is —, it is interpreted to be relative to the directory where the list's config file has been found. The default path is signature.

AddressFile "/path/to/file"

This tag specifies the path to the path of the file Petidomo uses to store the list of subscribed addresses. If the path specified here is relative — not starting with a "/" character that is —, it is interpreted to be relative to the directory where the list's config file has been found. The default path is list.

AcknowledgementFile "/path/to/file"

This tag specifies the path to the path of the file Petidomo uses to store the list of addresses that have been verified by the acknowledgement mechanism and may furtheron post without having to acknowledge their posting again. This is only used if the list is set to "acknowledge-once" mode.

If the path specified here is relative — not starting with a "/" character that is —, it is interpreted to be relative to the directory where the list's config file has been found. The default path is ack.

2.4 Command Line Syntax

Petidomo understans several command line parameters. Here is the complete list:

--mode=mode "listserv", "deliver", "approve", or "dump"

The mode parameter is the only mandatory parameter and it will determine what mode Petidomo runs in. Anyway, if Petidomo is started in "listserv" mode, it will expect to read an e-mail from standard input, which contains commands like "subscribe", "index" or "help". These commands will be carried out and notifications be sent back to the mail's originator if apropriate.

In “deliver” mode, Petidomo will read the incoming e-mail from standard input and post it to the mailing list, which’s name has been provided via the “listname” option. When running in “approve” mode, Petidomo will read the incoming mail from standard input and search for any cookies that mail might contain. If it does, it checks the ack-queue for a mail that has been deferred until confirmation that matches that cookie and processes the mail.

In “dump” mode, Petidomo will expect the name of a mailing list on the command line — the “listname” option — and dump the list of subscribed addresses on that list to standart output.

--listname=list name This parameter may contain any valid mailing list name. Depending on the mode, it this list name will be used as follows. In “listserv” mode, that list will be used as default list name for any command where no list name has been specified. A “subscribe”, for example” will subscribe the user to the list given here; a “subscribe some-other-name” will still subscribe the user to that other list, though.

When running in “deliver” mode, this is the name of the list the posting is supposed to be posted to. In “dump” mode, this is the name of the list, which’s subscriber list should be dumped. In “approve” mode, this parameter is ignored.

--masterconf=/path/to/petidomo.conf Using this parameter you can tell Petidomo to use a different location for the master config file than the one that has been compiled in.

--approved This flag is for internal purposes and should not be specified by the administrator. It will tell Petidomo that, whatever it is supposed to do now, is the result of a received confirmation cookie. This will effectively tell the checks for posting (in ListType “acknowledged” and “acknowledged-once” mode) and un-/subscription (in SubscriptionType “acknowledged” mode) that everything is fine and that the request should be carried out.

2.5 Aliases

The Petidomo binary will usually not be called manually from the shell, but by the mail transport agent. This works as follows: You create an e-mail account, which serves the purpose of accepting the incoming e-mail and piping it into the Petidomo binary.

This is archieved with the “alias”-function of your mail transport agent. Most MTAs, like sendmail, have a file where a list of special account names is given together with the instructions what to do with any mail received for that account. This file is usually located at /etc/aliases or /etc/mail/aliases.

One thing, aliases can do is pipe the mail into a program for processing. This is the mechanism Petidomo uses. Petidomo requires you to add the following aliases to your system:

```
#
# Mailing List Stuff
#
```

```
petidomo-manager:   postmaster
petidomo:           "|/usr/local/bin/petidomo --mode=listserv"
petidomo-approve:   "|/usr/local/bin/petidomo --mode=approve"
```

The lines starting with the '#' character are only comments and are ignored by the mail transport agent. The fourth line, though, is the first command. It tells the MTA to accept mail for an user of the name "petidomo-manager" and to re-direct the e-mail to the address "postmaster" — the mail system administrator.

Petidomo will send error notifications and things like that to the address "petidomo-manager". By setting this alias to a certain user name, you can control who will receive those mails.

The next line tells the MTA to pipe any incoming mail for the user "petidomo" into the "petidomo" program, instead of delivering it into a mailbox. "petidomo" (in listserv mode) will then parse the mail for commands and react accordingly. Hence, the address people can send their subscription requests to is "petidomo@your.host.name".

Similarly, the address "petidomo-approve" will be used to receive any acknowledges user send back after Petidomo requested them. Only now, Petidomo is started in "approve" mode.

Furthermore, each mailing list on your server *requires* three aliases, as shown in the example below, which is written for the "testlist" mailing list that comes with the distribution:

```
testlist:           "|/usr/local/bin/petidomo --mode=deliver testlist"
testlist-request:   "|/usr/local/bin/petidomo --mode=listserv testlist"
testlist-owner:     petidomo-manager
```

The first alias, "testlist" is the address to which people can send their mail in order to post to the mailing list. Any incoming mail for that account will be piped into the "petidomo" binary in "deliver" mode, which will process the mail and then re-send it to all subscribers of the mailing list. In order to let Petidomo know, for which mailing list the posting was meant, the parameter "testlist" has to be specified on the command line. If the name of the mailing list was "foobar", the line would look like this:

```
foobar:             "|/usr/local/bin/petidomo --mode=deliver foobar"
```

The second alias is a special request address, to which users can send their commands. The difference between this address and the "petidomo" alias described above is that here Petidomo is being given a default listname on the command line. The difference is this: If Petidomo receives a mail, which has the command "subscribe" in it, without any further parameters, it will reject the command with an error, because it doesn't know to which list the sender wants to be added.

If the command "subscribe" is sent to the "testlist-request" address, though, it will assume that the user wants to be subscribed to the "testlist" mailing list, as this is the default list for this address.

The name of this alias should always be the name of the mailing list with the string "-request" appended. Theoretically you could choose a different name, but this unwritten standard has been widely accepted in the Internet for several years now.

The last alias is the name of the mailing list with the string "-owner" appended. This alias points to the person who is responsible for managing the "testlist" mailing list. Petidomo will send all e-mail concerning the administration of the mailing list to the address "listname-owner". Usually this will ultimately be the same person as the "petidomo-manager", but you are free to direct mail for this account to somebody else, or to several persons.

Chapter 3

Petidomo for Mailing List Users

In this chapter, we will describe the commands, that are understood in "listserv" mode. This is the interface for the users of the mailing lists, where they can send their requests to in order to be subscribed to a mailing list, be unsubscribed again and similar things. The text here is mostly identical with the default help text that is sent to the user whenever he or she issues a command that is syntactically incorrect.

User commands always have to be sent to the request address of the mailing list — *not* to the mailing list itself!

Alternatively, commands can always be sent to the address "petidomo@your.address", but the "-request"-address is preferable, for the fact that the Petidomo will have a default listname for this address and thus understand a simpler command syntax.

3.1 SUBSCRIBE

The "subscribe" command will add the address of the user to a mailing list. When using the "-request"-address, only the word "subscribe" is required for the request to suceed. If the command is sent to the "petidomo" address, the user will have to specify an additional parameter: The name of the mailing list he or she wants to be added to, like in the following example:

```
subscribe politics
```

If the user wants to add an address that is not equal to the one he or she is sending the e-mail from, the e-mail address will have to be specified, too:

```
subscribe politics joe@foo.bar
```

The order in which the e-mail address and the mailing list name are provided does not matter. Please note that the administrator can configure Petidomo to disallow un-/subscring other addresses than the one, the request is sent from, using the "AllowAlienSubscription" option in the list's config file.

The command "add" is synonymous to "subscribe".

3.2 UNSUBSCRIBE

The syntax and usage of the “unsubscribe“ command are the same as the “subscribe” command. The difference is, though, the user’s address is removed from the mailing list rather than added to it.

“delete” and “remove” can be used synonymously to “unsubscribe”.

3.3 INDEX

The “index” command does not need any parameters. Sending it to the server will return a list of available mailing lists on this server. This is useful in case you want to subscribe to a list but can’t remember the exact name anymore.

The commands “lists” and “longindex” are synonyms to “index”.

3.4 HELP

If the server receives the command “help”, it will send the help file back. If “help” has a parameter, Petidomo will check whether this is a valid name of an existing mailing list, and if it is, it will return the description file for this mailing list, rather than the help file.

3.5 MEMBERS

The “members” command will return the addresses of all subscribers of the mailing list, if the administrator chose to allow this command. When “members’ is sent to the “-request”-address, the default list will be used by Petidomo. Otherwise, the name of the mailing list which’s subscribers should be listed, has to be specified as an option like in the following example:

```
members politics
```

The command “who” can be used synonymously to “members”.

Chapter 4

Petidomo for Administrators

On the "other side" of Petidomo, from the user's perspective, is the administrator of the mailing list — also called the *mailing list owner*). Each mailing list has an alias "listname-owner" (see section 2.5), where the mail address of the person who is responsible for this mailing list should be specified. Per default, this is the user who is known as "petidomo-manager". But you are free to direct mail for this accoun to any other person — or several persons.

The list owner will receive administrative e-mail from Petidomo in the following cases:

- When a new user subscribes, or a subscriber removes himself from the list, a carbon copy of the recipt will be sent to the owner. By looking at these mails, the owner can check whether a "subscribe" or "unsubscribe" command looks bogus. He or she can also keep track of who is on the list and who is not.
- If a "members" command is received for a mailing list where this command has been disabled, this will also be forwarded to the owner.

These mails are merely for information purposes and do not necessarily require an action from the admin. There are cases, where the list owner will receive mails from Petidomo, though, that require some kind of reaction.

4.1 Bounces

While maintaining mailing list with a larger number of subscribers, it happens regularly that subscribed addresses become invalid or are temporarily not reachable. In this case postings will *bounce*. You will then receive a mail from a mail server telling you, that the delivery of the mail failed.

Often, addresses become unreachable due to a misconfiguration of a machine, so it is not always necessary to remove that address from the list immediately, but when an addresses bounces for several days in a row, it is a good idea to delete that address from the mailing list. You should do that by sending

an “unsubscribe” command for that address to the “-request”-address of the mailing list.

If you have configured Petidomo to disallow the unsubscription of addresses not equal to the address the mail is sent from, you will have to specify your admin password in the mail, to override the barrier. How this is done is described in section 4.3 later.

4.2 Closed and moderated lists

If you have configured a mailing list to reject postings under certain circumstances, such as a closed or moderated mailing list, these rejected articles will be forwarded to you for approval. When you receive such a rejected article, you can either silently discard it, contact the author or post it to the mailing list with your approval.

You can approve an article with the master password for Petidomo, the admin password of the mailing list in question or the posting password (see section 2.3 of that list.

4.3 Approving requests

To approve an article, you have several ways of specifying the appropriate password. They are all the same for Petidomo and it is only a matter of taste, which scheme you use.

When sending a command to Petidomo in “listserv” mode through the “-request” or “petidomo”-address, it is easy: Just preface your commands with a “password” command, like in the following example.

```
To: testlist-request@foo.bar
Subject:

password open sesame
subscribe some@one.else
subscribe someone@even.elser
```

One “password” command sets your password for all the commands to follow. If you want to use one mail to send requests for several mailing lists with different passwords, just give a “password” command again:

```
To: petidomo@foo.bar
Subject:

password open sesame
subscribe user@inter.net testlist1
password let me in
subscribe user@inter.net testlist2
```

Instead of “password”, you can also use the commands “passwd”, or “approve”, they are all synonymous.

4.4 Approving postings

If you want to approve a posting for a mailing list, just send the article to the mailing list and specify your password either in the header or in the body of the mail.

If you choose to approve the mail in the body, add line with the command "approve" to the mail as first line of the body. Petidomo will strip that line before actually posting the article then. You can also use the synonyms "approved", "password" or "passwd" instead. Here is an example:

```
From: peter@example.org (Peter Simons)
Subject: Cats are the most beautiful animals in the world.

approve let me post
It's not that I wouldn't like animals like dogs, birds
or fishes, but for me, a cat is *the* animal to have.
[...]
```

The line "approve let me post" will be stripped by Petidomo and then the article will be sent out.

If you want to specify the password in the headers, just add an header of the name "Approved" or "Approve" to the headers of the mail. (Unfortunately, many mail readers do not allow you to modify the headers of outgoing mail. That is why the body-approval has been added.) Here is the same example as above now using the headers:

```
From: peter@example.org (Peter Simons)
Subject: Cats are the most beautiful animals in the world.
Approve: let me post

It's not that I wouldn't like animals like dogs, birds
or fishes, but for me, a cat is *the* animal to have.
[...]
```

Please note that you have to add a colon to the keyword to make a valid RFC mail-header.

4.5 The Access Control Language

Unfortunately, we live in a world where some people are trying to abuse services like mailing lists for their entertainment or for commercial purposes. It is also not uncommon that among thousands of mailing list subscribers, there is one particular moron who simply can't behave. That is why access control is a useful feature, even though it contradicts the idea of a mailing list: To be a media for communication.

Writing and understanding ACL files is, to be honest, not very easy and the novice mailing list administrator should better be careful when using them, because a wrong access control rule might cause more trouble than it is worth, but the experienced administrator will certainly appreciate their power. Understanding how ACL files work will also require you to know a bit about the

syntax of an RFC format e-mail. A good place to start is to take a look at RFC822 and its sons.

In Petidomo, two places exist to control who is allowed to do what: The global acl file and the acl file that is local to the mailing list. While the latter is valid only for the list in which's home directory it is stored, the globl acl file will be parsed for *all* your mailing lists. ACL files are only relevant for mailing list postings, Petidomo does not use them in "listserv" mode.

The syntax of an ACL file is similar to the C programming language, as you can see in the following example:

```
if (envelope matches "mailer-daemon@") then
        forward "petidomo-manager";
```

This is a simple version of the default ACL file which comes with the Petidomo distribution. It tells Petidomo to forward all postings to a mailing list, where the envelope of the mail matches the regular expression "mailer-daemon@". This rule is included in the default distribution to make sure that bounces of articles will not be posted to the list again, thus causing an infinite mail loop. The syntax of an ACL statement is shown in figure 4.1.

IF	(	from	match	"regexp"	)	THEN	pass	;
		subject	matches				drop	
		envelope	==	"string"			reject	
		header	=				rejectwith	"file"
		body					redirect	"address"
							forward	"address"
							filter	"script"
							approve	
IF	(		"filter"		)	THEN		;

Figure 4.1: Access Control Language syntax

Admittedly, the figure is rather impossible to understand without further explaination, don't worry if things are still a bit unclear after looking at it. There is also an EBNF grammar of the ACL to be found in figure 4.2, which might help those who can read BNF much more than the other figure.

Every ACL statement looks like this: "IF condition THEN action ;". The condition may or may not be enclosed in brackets. Several conditions can be combined with the keywords "OR" and "AND". Furthermore every condition can prefaced with a "NOT", which will reverse the outcome of the condition.

Let's explain this all at a concrete example: You want to reject all postings which come from the addresses "moron@moron.net" and "spam@spam.net", because these people have constantly been abusing your mailing list service. This can be done with the following two statements:

```
IF from == "moron@moron.net" THEN reject;
IF from == "spam@spam.net" THEN reject;
```

Using the "OR" statement you can combine this into one statement:

```
IF from == "moron@moron.net" OR
```

```
   from == "spam@spam.net" THEN
      reject;
```

And now we include brackets for readability:

```
IF (from == "moron@moron.net") OR
   (from == "spam@spam.net") THEN
      reject;
```

The keyword "from" stands for the address, noted in the "From:" header line of the mail and, the "== "address"" means that the condition if this address is equal to the one written in quotes thereafter. (You can also use a single '=' character, if you prefer that over two equal-characters.) This is a verbatim match. If we'd use the "match" or "matches" keyword instead of the "==", the parameter would be interpreted as an extended regular expression and the condition would be true if the addresses matched this pattern. (Regular expressions are described in the re_format(7) man page, or in the manpages of sed(1), grep(1) or egrep(1).)

Other keywords than "from" for the first part of the conditional are "subject" (the contents of the "Subject:" header), "envelope" (the envelope of the mail), header and body. The latter two represent the whole header or body of the mail and should be used only for regular expression matches and not for verbatim matches.

A short comment on the difference between "redirect" and "forward": The "redirect" action will send the mail to the specified address without changing anythin in the mail. All the headers are left untouched and thus the mail will look as if it has been sent by the person to that address right away. This is useful for redirecting mails to daemons or programs, but it will usually confuse a human recipient

The "forward" action, though, will send a mail to the specified address with a new set of headers, which identify the Petidomo Mailing List Manager as originator and then it will quote the mail that has been forwarded in the mail body.

Valid actions are "pass" (post the mail immediately), "drop" (discard the mail without further notice), "reject" (send a mail to the poster, telling him his posting was rejected), "rejectwith" (sending mail to the poster, too, but with the contents of a specified file), "redirect" (redirect the mail to a specified address), "forward" (like "redirect" but preface the mail with a note telling why the mail was re-sent) or "filter" (pipe the mail into the specified filter script and post the mail as the filter writes it to the standard output). Furthermore, there is the "approve" action that allows you to approve the posting, thus bypassing all other checks.

Here are a few more examples in the hope that they make this all easier to understand: Let's assume you would like to catch all postings to your mailing lists, that contain the words "MAKE MONEY FAST" in the subject. Then one way of doing this is the following statement:

```
IF (subejct matches "make money fast") THEN
      rejectwith "/usr/local/share/petidomo/make-money-fast.txt";
```

The file make-money-fast.txt could, for example, contain the following text:

```
Dear poster,

your mail has been rejected. Please note that chain letters
like the "make money fast" text you tried to post are
illegal throughout the world and your are likely to get in
trouble if you continue to spread them.
```

If someone tried to post the chain letter to your mailing lists now, he would receive a mail like that:

```
Date: Sat, 28 Jun 1997 19:59:18 +0200 (MET DST)
From: testlist-owner@example.org (Petidomo Mailing List Server)
To: simons@example.org
Cc: testlist-owner@example.org
Subject: Your posting to list "testlist" was rejected
Precedence: junk
Sender: testlist-owner@example.org

Dear poster,

your mail has been rejected. Please note that chain letters
like the ‘‘make money fast’’ text you tried to post are
illegal throughout the world and your are likely to get in
trouble if you continue to spread them.

>From simons  Sat Jun 28 19:59:17 1997
Received: from [[UNIX: localhost]]
        by example.org (8.8.5/8.8.4) id TAA16959
Date: Sat, 28 Jun 1997 19:59:17 +0200 (MET DST)
Message-Id: <199706281759.TAA16959@example.org>
From: Peter Simons <simons@example.org>
To: testlist
Subject: MAKE MONEY FAST
Mime-Version: 1.0 (generated by tm-edit 7.92)
Content-Type: text/plain; charset=US-ASCII

Hi, my name is David Rodes...
```

A few more words about how the ACL files are parsed:

- All comparisons are done case insensitive. “MAKE MONEY FAST” matches “make money fast” in both the verbatim and the regular expression match just fine.
- Any whitespace in the ACL file is ignored. The statements

  ```
  if (envelope matches "mailer-daemon@") then drop;
  ```

  and

  ```
  if
      (envelope matches
  "mailer-daemon@")
  ```

```
then
       drop
;
```

are the same to Petidomo.

- The argument after the "==" or "matches" keyword *has* to be included in quotes. An ACL statement like this:

  ```
  if from == peter@example.org then drop;
  ```

  will cause Petidomo to abort with an error, because it can't parse this.

- If you use an action that requires a parameter, like "rejectwith" or "forward", this parameter has to be enclosed in quotes, too. A statement like this can also not be parsed by Petidomo:

  ```
  if from == "peter@example.org" then
          forward postmaster@example.org;
  ```

- Petidomo stops parsing the ACL file after the first statement has matched. If you want to reject all mails from an address that matches "simons@.*de", but you want mails from the address "simons@rhein.de" to pass nonetheless, the following two statements will not work as expected:

  ```
  if from matches "simons@.*\.de" then reject;
  if from == "simons@rhein.de" then pass;
  ```

  Instead you should use

  ```
  if from == "simons@rhein.de" then pass;
  if from matches "simons@.*\.de" then reject;
  ```

  or

  ```
  if (from matches "simons@.*\.de") and
     (not (from == "simons@rhein.de")) then
           reject;
  ```

- Currently you can't match for the double quote character ("), we're afraid. The escape sequence `\"` is not supported yet.

One last example and then we'll come to the filters. The following statement rejectes a mail based on a match in the headers. This is very useful for rejecting mail from known spam domains. You usually can't rely on the spammer to use a valid "From:" header and hence the "from"-match is useless to catch them. But the following statement will usually get them nonetheless:

```
if (header matches "^Received:.*from spam.domain") then
     forward "petidomo-manager";
```

If you thought, the Access Control Language is powerful so far, take a look at the things you can do using filters. Rather than the examples described above, you could use the following statement:

```
if ("/usr/local/libexec/petidomo/CheckPosting") then reject;
```

This is a special form of the usual ACL statements and it means the following: The mail in question is piped into the "CheckPosting" script. The script or program can perform various tests and when it exists, the action part is executed depending on the return code the script exited with. A return code of zero (0) means "true" and the action will be executed. A return code of one (1) "false" and the action will not be executed.

Any other return code will cause Petidomo to abort with an error and to save the mail. By using this mechanism, you can program even the most sophisticated tests and hook them into the access control mechanism.

Another feature that hasn't been described yet is the action "filter". The filter-action is pretty much the same as the posting filter, but it allows you to re-write the posting depending on who posted it or other criteria. Please note that this filter is executed additionally to a regular posting filter you might have configured.

A nice example for what this feature can be used is the following:

```
if (address == "peter@example.org") then
       filter "/usr/local/libexec/petidomo/simons.filter";
```

The script simons.filter would then look like this:

```
#! /bin/sh

cat
echo
echo "-- "
echo " Hold your breath -- this is *the* Peter Simons!"
```

We resisted the temptation of adding this ACL statement into the default configuration of Petidomo.

```
input:     /* empty */
         | input statmt
         ;

statmt:    ';'
         | 'if' exp 'then' action ';'
         ;

exp:       qualifier '=' string
         | qualifier 'match' string
         | string
         | exp 'or' exp
         | exp 'and' exp
         | '!' exp
         | '(' exp ')'
         ;

qualifier: 'from'
         | 'subject'
         | 'envelope'
         | 'header'
         | 'body'
         ;

action:    'pass'
         | 'drop'
         | 'approve'
         | 'reject'
         | 'rejectwith' string
         | 'redirect'   string
         | 'forward'    string
         | 'filter'     string
         ;

string:    '"' [^"]* '"'
```

Figure 4.2: EBNF of the Access Control Language

Chapter 5

Miscellaneous Topics

5.1 Using posting filters

The posting filter functionality of Petidomo is a very useful mechanism for you, if you run mailing lists where you want to guarantee certain formal criteria, because you can hook a script or program of your into the posting process and use it to re-format or re-write the article that is going to be posted.

We have included one script into the distribution, InsertNameInSubject.sh, which adds a string into the subject line of every posting. The script is pretty short and used sed(1) to perform its function.

To use it, just add the line

```
PostingFilter     ~petidomo/bin/InsertNameInSubject.sh listname
```

with "listname" being the name of the mailing list.

If the mailing list name was "testlist", for example, then this posting filter would re-write the subject line

```
Subject: Hi everbody
```

to

```
Subject: [testlist] Hi everbody
```

It is recommended to take a look at the script itself to understand how this works. You will need a bit of knowledge of the Unix scripting language and tools like sed(1) to program more complex posting filter, we're afraid.

As the last point it should be made clear, that the string you specify as a filter is interpreted by the bourne shell for execution. It is thus absolutely possible, to use a posting filter like that

```
PostingFilter "/bin/cat | /bin/cat ; echo ; echo testing"
```

even though one might argue whether this particular example is a useful thing. Anyway, you know what we wanted to demonstrate.

5.2 PGP-encrypted mailing lists

Another very useful feature of the posting filter and the access control languange is the ability to maintain *encrypted mailing lists*. The idea is very simple: You create a PGP key pair for your mailing list and spread the public key among the subscribers of your mailing list. In turn you collect their public keys and store them on the mailing list server.

Whenever a subscriber wants to post an article to the mailing list, he will encrypt it with the public key of the list server before transferring it through the Internet. Petidomo will then receive the mail, decrypt and process it and encrypt it again, with the public keys of the subscribers. Once encrypted again, the mail is distributed to the readers.

Please note that at no time the mail was sent through the Internet in clear text. Hence this mode is well-suited for maintaining internal discussion lists for, say, software development among a few people who know each other but live spread throughout the world. Included in the distribution are two scripts, pgp-encrypt.sh and pgp-decrypt.sh, which realize this. The setup needs a bit of work, but once you understand the principle, it is rather easy. Just follow the steps described below.

1. Get the PGP software package from 'http://www.pgpi.com/', you will need the PGP 2.6.2 version or later — 5.x won't work, as far as I know, maybe someone wants to adapt the PGP-mechanism to PGP 5.x, any volunteers are welcome, and install it.

2. Log in as user "petidomo".

3. Create a directory .pgp in the home directory of the users Petidomo runs under and set the `$PGPPATH` variable to it.

4. Create a PGP key pair by calling 'pgp -kg". As user-id enter the address of the mailing list itself, for example: "The secret mailing list <secretlist@example.org>".

5. Create a config.txt file for PGP in the .pgp directory and insert the appropriate user id there.

6. Distribute the newly created PGP key of the mailing list among the subscribers.

7. Add the PGP keys of the subscribers to Petidomo's keyring.

8. Edit the following definitions in pgp-encrypt.sh:

```
#
# Please customize these things for your system.
#
PGP=/usr/local/bin/pgp
PASSWORD="DecryptMe"
PGPPATH=$PDHOME/.pgp
```

   You will need to change the location of the PGP binary and insert the password you chose for the secret key. For security reasons, the script

itself should be owned by "petidomo" as user- and group id, and it should have the permission "110", so that only Petidomo can execute it.

9. Edit the equivalent definitions in pgp-encrypt.sh.

10. Now create the mailing list in question. In our example that would be "secretlist". Naturally the mailing list should not be open for public subscription.

11. Edit the ACL file of the "secretlist" to contain the following line:

```
if (body matches "^-----BEGIN PGP MESSAGE-----$") then
        filter "~petidomo/bin/pgp-decrypt.sh";
```

12. Edit the config file to have the following posting filter:

```
PostingFilter   "~petidomo/bin/pgp-encrypt.sh secretlist"
```

    Please note that you must provide the name of the mailing list on the command line as parameter to pgp-encrypt.sh, so that it know which list file it should take the subscriber addresses from.

13. Do a test posting and that's it.

There are a few things you should take care of: First of all, you must make sure that you have the PGP public keys of all subscribers in the keyring belonging to the "petidomo" user, or some of them won't be able to decipher the mail posted via the list. You must also take care that the addresses these people are subscribed under, are actually listed in their public key, or PGP won't be able to recognize who is who, when being called by pgp-encrypt.sh.

Finally, make sure that you do this only with the correct versions of the software. Petidomo needs to be version 2.1 or later, earlier versions won't work. The PGP binary needs to understand the `-@` operator on the command line, which has been added in PGP 2.6i at some time.

One last hint: If PGP-encryption or decryption doesn't work, it will usually help to remove the `$LOGFILE` parameter from the `trap` command in the scripts:

```
trap 'rm -f $TMPFILE $HEADER $BODY $NEWBODY $LOGFILE; exit'...
                                                ^^^^^^^^
```

As a result, the script won't delete the output PGP issued when called after exiting. Thus you will find the file still lying in /tmp and can easily investigate what's wrong.

5.3 Virtual hosting and sendmail

A very useful things is Petidomo's virtual hosting feature. Unfortunately, there are a few traps into which you might run when you are trying to use it together with the Allmann sendmail. But we'll start at the beginning.

If you host mailing lists for domains other than your own, you can tell Petidomo to use this name instead of the local one for certain mailing lists in the list config file by setting the appropriate "HostName". Now all mail posted

to this list, or sent to the "-request" address of the list, will appear to be coming from the domain name you configured, rather than your own.

When you are using sendmail v8, you will have to write these names to the w class in your sendmail.cf file, or the corresponfing M4 config. This is done by adding the line

```
Cwdomain.name1 domain.name2 ...
```

to the file.

This will tell sendmail that these names are to be accepted and delivered locally rather than to the MX of these entries.

Doing this might deliver a surprise, though, if you are using sendmail's masquerading abilities to hide the various hostname of your domain. Per default, sendmail masquerades not only the domain names you told him with the MASQUERADE_DOMAIN() command, it automatically masquerades all domain names of the w class, too.

The result is that Petidomo's fine virtual hosting is gone immediately, because sendmail will re-write the name to your own domain name. The fix for this is rather easy: Add the command "FEATURE(limited_masquerade)" to your M4 file and sendmail won't touch the names that are stated only in the w class.

5.4 Mailing list archives

If your are hosting a public mailing list, you might want to offer a mailing list archive, that is accessible through the WWW and has all the postings available for immediate access. We were in the midst of developing a tool that does this for you when we came accross a brilliant tool named "MHonArc". We installed it, tested it, and deleted all attempts of writing something like that ourselves immediately.

We strongly recommend looking at MHonArc, if you want to offer a WWW archive of your mailing lists. You can find more information about MHonArc at the following location: 'http://www.oac.uci.edu/indiv/ehood/mhonarc.html'

The installation of the tool itself is very easy. Once you have MHonArc running, just enable the archiving feature in Petidomo and feed the archives into MHonArc. That's it.

5.5 Verifying the address lists

Petidomo tries its best to make sure that only syntactically correct addresses are subscribed to mailing lists, and if you stick to the correct mail interface, there's very little chance, an incorrect address will make it into the list file.

Sometimes, it is necessary to edit these files manually, though, or Petidomo's address validation algorithm fails. Once you have an incorrect address in your list file, sendmail will abort with an error, without trying to deliver the mail at all.

To clarify, this does not happen when an address is not reachable, this happens only when you subscribe something like `hey@this@is@wrong....`. Once you suspect that your address list has been corrupted, there's an easy way to

find out, which addresses are wrong. Simply use sendmail's address verification mode like this:

```
$ xargs <list sendmail -bv | sed -e '/deliverable/d'
> @bogus.address.here... user address required
```

This call will find all incorrect address and notify you. The 'sed' call will filter out all correct addresses for your convenience.