GNU Anubis

An SMTP message submission daemon.
GNU Anubis Version 4.0
18 December 2004

Wojciech Polak and Sergey Poznyakoff

1 Overview

GNU Anubis is an SMTP message submission daemon. Its purpose is to receive the outgoing message, perform some manipulations over its contents, and to forward the altered message to the mail transport agent.

The usual mail sending scheme looks as follows: the user composes his message using *mail user agent* (*MUA* for short). Once the message is composed, the user sends it. When the MUA receives the send command it connects to the *mail transport agent* (*MTA* for short) and passes it the message for delivery. The figure below illustrates this interaction:

```
+-------+                 +-------+
|  MUA  | ---[outmsg]---> |  MTA  | ... [outmsg]
+-------+                 +-------+        |
                                           |
                                           V
                                   +--------------+
                                   |  Recipient's |
                                   |   Mailbox    |
                                   +--------------+
```

As shown in this figure, the outgoing message (*outmsg*), reaches the recipient's mailbox unaltered.

However, there are situations where it may be necessary to modify the outgoing message before it reaches MTA. As the simplest example, the user might wish to sign the outgoing messages with his PGP key, but his MUA does not support this operation or supports it unconditionally.

In such cases, installing GNU Anubis between the MUA and MTA allows the user to perform any additional processing on the sent message. The figure below illustrates this concept:

```
+-------+                 +--------+                 +-------+
|  MUA  | ---[outmsg]---> | Anubis | ---[modmsg]---> |  MTA  |
+-------+                 +--------+                 +-------+
                                                         |
                                                      [modmsg]
                                                         .
                                                         .
                                                         V
                                                 +--------------+
                                                 |  Recipient's |
                                                 |   Mailbox    |
                                                 +--------------+
```

The outgoing message is processed by GNU Anubis, and it is the resulting message (*modmsg*) that reaches the MTA.

GNU Anubis is able to perform on messages a wide set of operations, such as modifying message headers or body, encrypting or signing messages with GPG (GNU Privacy Guard) keys, installing secure tunnels to MTA using TLS/SSL encryption, tunneling messages through SOCKS proxies, etc.

When the set of built-in operations is not enough, the user can define his own operations using Guile, a *GNU's Ubiquitous Intelligent Language for Extensions*.

The message processing is controlled by system-wide and per-user configuration files written in a flexible and easy to use command scripting language, specially designed for this purpose.

2 Glossary of Frequently Used Terms

Authentication
A process whereby Anubis determines the authenticity of the connecting party, its user name and configuration settings.

Protocol Any standard for the exchange of information. A protocol defines the specific wording and control flow for communications between two or more programs, devices, or systems.

SMTP Simple Mail Transport Protocol is a common mechanism for exchanging mail across a network. This protocol is described in the RFC 821 document.

Daemon We use a term *daemon* to define a process that runs in the background, doing automated processing.

Server A server provides information or other services for its clients. Most network protocols are client–server based. This term usually refers to an entire machine, but it can refer (and we're doing that) also to the particular program or process, on that machine, that provides the service.

Proxy We use a term *proxy* to define a program, which goes between the MUA and the MTA (it makes a tunnel). It can be used as a gateway to the outside world, while using a firewall. In this case the host under the firewall sends data to the proxy server, which in turn forwards it to the real server outside, receives the response, and passes it back to the internal host.

Guile GNU's Ubiquitous Intelligent Language for Extensions. It provides a Scheme interpreter conforming to the R4RS language specification. GNU Anubis uses Guile as its extension language. For more information about Guile, See section "Overview" in *The Guile Reference Manual*.

GPG GNU Privacy Guard, a tool compatible with the Pretty Good Privacy.

3 Authentication

When GNU Anubis accepts an incoming connection, it first has to identify the remote party, i.e. determine whether it has the right to use Anubis resources and, if so, what configuration settings should be used during the session. We call this process *authentication*. The exact method of authentication depends on Anubis *operation mode*. Currently there are two modes:

transparent
: This is the default mode. It is compatible with versions of GNU Anubis up to 3.6.2. In this mode, Anubis relies on AUTH service (`identd`) to authenticate users.

auth
: This mode uses SMTP AUTH mechanism to authenticate incoming connections. See Chapter 12 [Pixie-Dixie], page 51, this is the first draft description of this mode.

Both modes have their advantages and deficiencies, which you have to weigh carefully before choosing which one to use. These are discussed below:

Transparent (‘`traditional`’) mode.

Deficiencies:

1. The user must have `identd` installed on his machine.
2. Each user must have a system account on the machine where GNU Anubis runs (though the system administrator may relax this limitation using user name translation, see Section 4.3 [TRANSLATION Section], page 22).

Advantages:

1. Relative simplicity. You don’t have to create your users database.
2. Authentication is performed immediately after the connection.

Auth mode.

Deficiencies:

1. You have to maintain your users database
2. User’s MUA must be able to perform ESMTP AUTH.[1]

Advantages:

1. Better reliability.
2. Users do not have to run `identd` on their machines.
3. Users are not required to have accounts on the machine where Anubis runs.
4. Users can remotely modify their configuration files.

---

[1] It is not a serious restriction, however. The user may install Anubis on his machine for the sole purpose of SMTP authentication, as Pixie-Dixie suggests.

3.1 User Database

GNU Anubis uses *User Database* for keeping *user credentials*, i.e. data used to authenticate and authorize users. The exact way of storing these data does not matter here, it will be addressed further in this manual. In this section we treat user database as an abstraction layer.

The user database consists of *records*. Each record keeps information about a particular *user*. A record consists of four *fields*. A field may contain some value, or be empty, in which case we say that the field has *null* value.

The record fields are:

`SMTP AUTHID`
: SMTP authentication ID of the user.

`AUTH PASSWORD`
: SMTP password.

`ACCOUNT`
: System user name to be used.

`CONFIG`
: Path to the configuration file.

The first two fields are mandatory and must always have non-null values. No two records in the database may have the same value of `SMTP AUTHID` field. When `anubis` is trying to authenticate a user, it first looks up in the database a record with the value of `SMTP AUTHID` field matching AUTHID given by the user. If no such entry is found, authentication fails. Otherwise, `anubis` goes on and compares the password supplied by the user with that from `AUTH PASSWORD` column. If these match, authentication succeeds and `anubis` passes to authorization state.

In this state, it first determines the user ID (UID) to switch to. If the `ACCOUNT` field is not null, its value is used as a login name of the system account to use. If it is null, `anubis` switches to the privilege level of a *default not privileged user*, specified by `user-notprivileged` statement in the global configuration file (see Section 4.2.6 [Security Settings], page 21).

The final step is to parse *user configuration file*. If `CONFIG` field is not null, its value is used as absolute path to the configuration file. Otherwise, `anubis` searches for file '`~/.anubisrc`' (where '`~`' denotes home directory for the system account obtained on the previous step) and if such a file exists, loads it.

3.2 Database URL

Anubis database is identified by its *URL*, or *Universal Resource Locator*. A `URL` consists of following elements (square brackets enclose optional elements):

```
proto://[[user[:password]@]host]/path[params]
```

The detailed description of each `URL` part follows:

proto Specifies a database *protocol*. The protocol describes how the database is to be accessed. In a way, it may be regarded as specifying the database *type*. Currently, GNU Anubis supports following database protocols:

'`text`'	A plain text file, containing users' credentials.
'`gdbm`'	GDBM database
'`mysql`'	MySQL database
'`pgsql`'	PostgreSQL database
'`postgres`'	Alias for '`pgsql`'.

These protocols are described in detail below.

user User name necessary to access the database.

password User password necessary to access the database.

host Domain name or IP address of a machine running the database.

path A *path* to the database. The exact meaning of this element depends on the database protocol. It is described in detail when discussing particular database protocols.

params A list of protocol-dependent parameters. Each parameter is of the form `keyword=name`, parameters are separated by semicolons.

3.2.1 Plain text databases

This is the simplest database possible. It is kept in a plain text file. Each line in this file represents a single *record*, empty lines and lines beginning with '`#`' (*comments*) sign are ignored. Records consist of *fields*, each field being a sequence of characters. Fields are separated by colons ('`:`', ASCII 58). If '`:`' character occurs in a field, it is preceeded by a single backslash character ('`\\`', ASCII 92). A record must contain at least two fields.

1. SMTP '`AUTHID`'.
2. SMTP password.
3. Account name.
4. Path to user configuration file.

URL syntax

The URL syntax for this type of databases is quite simple:

```
text:path
```

where *path* specifies absolute file name of the database file.

3.2.2 Databases in GDBM format

The protocol value '`gdbm`' specifies a *GDBM database*. For the detailed description of GDBM system section "Introduction" in *The GNU DBM Manual*.

URL syntax for GDBM databases is:

```
gdbm:path
```

where *path* specifies absolute file name of the database file.

3.2.3 MySQL and PostgreSQL

This is the most flexible database format. GNU Anubis 4.0 supports MySQL[2] and PostgreSQL[3] interfaces. No matter which of them you use, the implementation details are hidden behind a single consistent Anubis interface.

GNU Anubis supposes that all user data are kept in a single database table. This table must have at least four columns for storing SMTP '`AUTHID`', SMTP password, system account name and path to user configuration file. Among those, only the last two may have NULL values. There is no restriction on the name of the database or the authentication table, nor on its column names. This information may be specified in URL as discussed below.

URL syntax

```
proto://[[user[:password]@host/dbname[params]
```

Proto describes the exact database type to use. Use '`mysql`' for MySQL databases and '`pgsql`' or '`postgres`' for PostgreSQL databases.

Optional *user* and *password* specify authentication credentials used to access the database.

Host sets domain name or IP address of the machine running the database. It may be omitted if the database resides on '`localhost`'.

The database name is specified by *dbname* element.

Finally, further details needed for connecting to the database may be given by URL parameters. All of them have reasonable default values, so you'll have to specify only those parameters that does not match the default values. Known parameters are:

port=number
: Specifies the port number to be used when connecting to the database. If it is not specified, the behavior depends on the value of *socket* parameter: if *socket* is not present, the program will use the default port number for the given protocol (i.e. 3306 for '`mysql`' and 5432 for '`pgsql`'.

socket=string
: Specifies UNIX name of the socket to connect to. This parameter cannot be used together with *port* (see above).

bufsize=number
: Sets the length of the buffer used to create SQL queries. Default is 1024 bytes.

---

[2] See `http://www.mysql.com`.

[3] See `http://www.postgres.org`.

table=string
: Specifies the name of database table keeping where the authentication data are stored. Default is '`users`'.

authid=string
: Specifies the name of a column in *table* which holds '`AUTHID`' value. Default is '`authid`'.

passwd=string
: Specifies the name of a column in *table* which holds user password. Default is '`passwd`'.

account=string
: Specifies the name of a column in *table* which holds the name of system account to be used for this '`AUTHID`'. Default is '`account`'.

rcfile=string
: Specifies the name of a column in *table* which holds path to the user's configuration file. Default is '`rcfile`'.

3.3 Managing the Database

Managing the user database is a complex task, which looks differently from administrator's and user's point of view. The administrator have full rights on the database, it can add new records and delete or modify existing ones. A user, of course, does not have such ample rights. The only thing he is able to do is to maintain his own record in the database, provided that he already has one. If he does not, he should contact the system administrator and arrange for the creation of his record.

3.3.1 Administrators

All administrative tasks are done using `anubisadm` command — a multipurpose tool for Anubis administrator.

The command usage syntax is:

```
anubisadm command [options] database-url
```

where *command* specifies the operation to be performed on the database, *options* give additional operation-specific parameters, and *database-url* specifies the database to operate upon.

All administrative tasks can be subdivided into the following five categories:

- Creating the Database
- Listing Database Records
- Adding New Records
- Removing Existing Records
- Modifying Existing Records

These operations are described in detail in the following subsections.

3.3.1.1 Creating the Database

To create a database use *anubisadm --create* (or *anubisadm -c*) command. `Anubisadm` will read database entries from the standard input and write them to the database. The standard input is supposed to be formatted as `text` database (see Section 3.2.1 [text], page 7).

Thus to create a GDBM database from plain text file '`userlist`', use the following command

```
anubisadm --create gdbm:/etc/anubis.db < userlist
```

Similarly, to create an initially empty database, type

```
anubisadm --create gdbm:/etc/anubis.db < /dev/null
```

Notice, that if you use SQL database format, '`--create`' command does not imply creating the database structure! So, before running

```
anubisadm --create mysql://localhost/dbname < userlist
```

make sure you create the underlying database structure (including granting privileges to the `anubis` user), via the usual procedure. Please refer to corresponding database manual for the detailed instructions on this.

It is sometimes necessary to convert the existing user database from one format (protocol) to another. For example, suppose you have been running GDBM database (`text:/etc/anubis.db`) for some time, but now it has grown considerably and you decided to switch to PostgreSQL database to improve performance. To do so, first create the database using postgres utilities. Then run

```
anubisadm --list text:/etc/anubis.db | \
 anubisadm --create pgsql://localhost/dbname
```

That's all there is to it!

3.3.1.2 Listing Database Records

The command '`--list`' (or '`-l`') lists the existing database. When run without additional options, it will display all records from the database, e.g.:

```
anubisadm --list gdbm:/etc/anubis.db
```

Among its other uses, such invocation is handy for converting user database to another format (see Section 3.3.1.1 [Create], page 10).

If you wish to list only a particular record, specify the `AUTHID` using '`--authid`' ('`-i`') option. For example, to list record of the user with `AUTHID` '`test`', type:

```
example$ anubisadm --list --authid test gdbm:/etc/anubis.db
```

3.3.1.3 Adding New Records

To add a new record use command '`--add`' ('`-a`'). Additional data are specified via the following options:

'-i *string*'
'--authid=*string*'
Specify the user SMTP AUTHID.

'-p *string*'
'--password=*string*'
Specify user password password.

'-u *string*'
'--user=*string*'
Specify system user name corresponding to the given AUTHID.

'-f *string*'
'--rcfile=*string*'
Specify configuration file to be used for this user.

For example, the following command adds a record with SMTP AUTHID 'test', password 'guessme' and maps it to the system account 'gray':

```
anubisadm --add --authid test --password guessme \
          --user gray gdbm:/etc/anubis.db
```

3.3.1.4 Removing Existing Records

Removing a record is quite straightforward: use '--remove' ('-r') command and specify AUTHID using '--authid' option. For example, to remove the record created in the previous subsection, run:

```
anubisadm --remove --authid test gdbm:/etc/anubis.db
```

3.3.1.5 Modifying Existing Records

To modify an existing record use command '--modify' ('-m'). The record is identified via '--authid' option. The fields to be changed are given with the following options:

'-p *string*'
'--password=*string*'
Specify user password password.

'-u *string*'
'--user=*string*'
Specify system user name corresponding to the given AUTHID.

'-f *string*'
'--rcfile=*string*'
Specify configuration file to be used for this user.

For example, the following command sets new configuration file name for the user 'smith':

```
anubisadm --authid smith \
          --rcfile=/var/spool/anubis/common gdbm:/etc/anubis.db
```

3.3.1.6 Summary of All Administrative Commands

Usage

```
anubisadm command [options] database-url
```

Commands:

'-c'
'--create'
  Create the database.

'-l'
'--list' List the contents of an existing database.

'-a'
'--add' Add a new record.

'-m'
'--modify'
  Modify an existing record.

'-r'
'--remove'
  Remove an existing record.

'--version'
  Display program version number and exit.

'--help' Display short usage summary and exit.

Options:

'-i *string*'
'--authid=*string*'
  Specify the authid to operate upon. This option is mandatory for '--add', '--modify' and '--remove' commands. It may also be used with '--list' command.

'-p *string*'
'--password=*string*'
  Specify the password for the authid. This option is mandatory for '--add', '--modify' and '--remove' commands.

'-u *string*'
'--user=*string*'
  Specify the system user name corresponding to the given authid. It may be used with '--add', '--modify', and '--remove' commands.

'-f *string*'
'--rcfile=*string*'
  Specify the rc file to be used for this authid. The option may be used with '--add', '--modify', and '--remove' commands.

3.3.2 Users

Users maintain their database records using `anubisusr` command. Main purpose of this command is to keep the copy of your configuration on GNU Anubis server up to date. . We recommend to invoke `anubisusr` from your '`~/.profile`', which will make sure that your configuration file is up to date when you log in.[4].

Usage

```
anubisusr [options] [smtp-url]
```

where *smtp-url* is a URL of your GNU Anubis server. Notice that if it lacks user name and password, then `anubisusr` will first try to retrieve them from your '`~/.netrc`' file (See *netrc(5)* for more info), and if not found it will prompt you to supply them.

Options

'`-m` *mech*'
'`--mechanism` *mech*'
: Only use SASL mechanism *mech*. Use this option several times to set a list of allowed mechanisms.

'`-v`'
'`--verbose`'
: Verbose output. Multiple options increase the verbosity. Maximum verbosity level is 3.

'`--version`'
: Display program version number and exit.

'`--help`' Display short usage summary and exit.

---

[4] Make sure to run `anubisusr` in background, so it does not slow down your normal login sequence

4 Configuration

The behavior of GNU Anubis is controlled by two configuration files. The *system configuration file*, '/etc/anubisrc', specifies system-wide options that affect all users. This file is usually owned by root. The *user configuration file* specifies what GNU Anubis should do for a particular user. By default it is located in '~/.anubisrc'. This location can be changed in auth mode. To protect your passwords in the configuration files, use the 0600 (u=rw,g=,o=) permissions, otherwise GNU Anubis won't accept them.

Lexical Structure

Both configuration files use simple line-oriented syntax. Each line introduces a single statement. A statement consists of *words*, each word being defined as a contiguous sequence of non-whitespace symbols. A word may be composed of alphanumeric characters and any of the following punctuation symbols: '_', '.', '/', '-'. Any arbitrary sequence of characters enclosed in a pair of double quotes is also recognized as a word.

The familiar shell *here document* syntax may be used to produce a word containing several lines of text. The syntax is:

```
<<[-]delimiter
    text
delimiter
```

If "here document" starts with '<<-', then all leading tab characters are stripped from input lines and the line containing *delimiter*. This allows to indent here-document in a natural fashion.

To summarize all the above, let's consider the example:

```
first-word "second word" <<-EOT
                            Third word
                            containing several
                            lines of text
                            EOT
```

This line contains three words: 'first-word', 'second word' and the third one composed of the three lines between the 'EOT' markers.

If a statement is very long, it may be split among several lines of text. To do so, precede the newline characters with a backslash '\', e.g.:

```
a very long statement\
  occupying several lines\
  of text
```

A '#' in a line starts a *comment*. It and the rest of the line are ignored. Comments may appear on any of the lines in the configuration file, except on a commands and within a "here-document" construction. A line containing just a comment (with perhaps spaces before it) is effectively blank, and is ignored. For example:

```
# This is a comment
if header[Subject] :re "No.*"  # This is also a comment
  guile-process action-name This # is not a comment!!!
fi
```

Logical Structure

The statements within a configuration file are grouped into *sections*. Each section has its name. A section begins with one of the following constructs:

```
BEGIN name
---BEGIN name---
```

and ends with one of the following constructs:

```
END
---END---
```

Notice, that both 'BEGIN' and 'END' must be uppercase. When using the second form, any amount of whitespace is allowed between the three dashes and the word.

The sections cannot be nested.

There are five predefined sections, whose names are uppercase. The user may define his own sections, which may then be referred to from the `RULE` section as subroutines (see Section 5.6.2 [Call Action], page 28).

The predefined section names are:

AUTH Controls authentication mechanisms.

CONTROL
: This section specifies the basic GNU Anubis behavior. Its presence is required in the system configuration file. It may be used in the user configuration file to override the system-wide settings.

TRANSLATION
: This section specifies a translation map for remapping remote or local users. It may be used only in the system-wide configuration file.

GUILE Contains the settings of the Guile interpreter. The section is allowed in both configuration files.

RULE Defines the rules that are used to alter the contents of the messages (conditional and unconditional rules).

4.1 AUTH Section

`AUTH` session controls various aspects of authentication mode.

`smtp-greeting-message` *text* [Option]

Configures the greeting message issued by GNU Anubis upon accepting the connection.

`smtp-help-message` *`help-text`* [Option]
Sets the test of the message issued by Anubis in response to SMTP `HELP` command. *Help-text* is a list of strings. Each string from the list will be displayed at a separate response line.

`sasl-password-db` *`url`* [Option]
Sets the user database URL (see Section 3.1 [User Database], page 6).

`sasl-allowed-mech` *`mech-list`* [Option]
Defines the list of allowed authentication methods.

4.2 CONTROL Section

The '`CONTROL`' section specifies the basic GNU Anubis behavior. Specified in the system configuration file, it applies to all users on the machine, but each user can specify its own '`CONTROL`' section, to customize own settings. Of course, not all options can be set or change by user. Some options can only be set in the system configuration file, and some only in user configuration file. By default, options specified in user configuration file have a **higher** priority that those specified in system configuration file.

All option names are case insensitive, so you can use for instance: `bind` or `BIND` or `BiNd`, and so on.

4.2.1 Basic Settings

`bind` [*`host`*:]*`port`* [Option]
Specify the TCP port on which GNU Anubis listens for connections. The default *host* value is '`INADDR_ANY`', which means that anyone can connect to GNU Anubis. The default *port* number is 24 (private mail system). This option is available only in the system configuration file. If you would like, for instance, to bind GNU Anubis to port 25 (SMTP) and limit its clients only to those from '`localhost`', then set the following in your system configuration file:

```
bind localhost:25
```

`remote-mta` *`host`*[:*`port`*] [Option]
Specify a remote SMTP host name or IP address, which GNU Anubis will connect and forward mail to (after a processing). The default *port* number is 25. This option is available in both configuration files.

`local-mta` *`file-name`* [*`args`*] [Option]
Execute a local SMTP server, which works on standard input and output (inetd-type program). This option excludes the '`remote-mta`' keyword (or '`--remote-mta`' command line option). For example:

```
local-mta /usr/sbin/sendmail -bs
```

`mode` *`mode-name`* [Option]
Selects Anubis operation mode. Allowed values for *mode-name* are:

transparent
auth

See Chapter 3 [Authentication], page 5, for the detailed discussion of GNU Anubis operation modes.

4.2.2 Output Settings

`termlevel` *`level`* [Option]

This is a logging level for `syslogd` or a terminal (if using the '`--foreground`' command line option). *level* can be one of the following:

normal Only errors are logged. This is the default level.

verbose Produce more diagnostic output.

debug Produce debugging output.

silent Do not log anything.

This command may be used only in system configuration file.

`logfile` *`file-name`* [Option]

This command specifies an additional file, where GNU Anubis can log its information, but only those information available for a client. Only in user configuration file. For example:

```
logfile "anubis.log"
```

This will log to the '`~/anubis.log`' file in a client's home directory.

`loglevel` *`level`* [Option]

This option specifies an output level for an additional file ('`logfile`'). It can be used only in user configuration file. *level* is one of the following:

none

fails

all

`tracefile` *`yes-or-no`* [Option]
`tracefile` *`file-name`* [Option]

This option instructs `anubis` to log the execution of tests and actions from the RULE sections. This is useful for debugging the configuration files.

When this option is used in the system-wide configuration file, only its first form is allowed. Using '`tracefile yes`' enables logging of the actions and tests to the default syslog channel. Using '`tracefile no`' disables it.

When used in the user configuration file, a filename is allowed as an argument to this option. This allows you to explicitly specify to which file the tracing output should go. Otherwise, using '`tracefile yes`' enables logging to the same file as '`logfile`' (if possible).

4.2.3 Proxy Settings

`socks-proxy` *host*[*:port*] [Option]
This option enables tunneling the connections through a SOCKS proxy server, specified as an argument *host*. The *port* default value is 1080, which is a common port number for SOCKS proxies.

`socks-v4` *yes-or-no* [Option]
This specifies a SOCKS protocol version 4. By default it is turned off, and a default mode is SOCKS protocol version 5.

`socks-auth` *username:password* [Option]
Specify a user name and a password, if a SOCKS proxy server requires them. A *username* and a *password* are separated with a colon (':').

4.2.4 ESMTP Authentication Settings

The following options set authentication credentials for ESMTP authentication. You may use this option, for example, if your MTA requires such an authentication, but your MUA does not support it.

`esmtp-allowed-mech` *mech-list* [Option]
Defines the list of allowed authentication mechanisms. *Mech-list* is a list of valid authentication mechanism names separated by whitespace.

Anubis selects the authentication method using following algorithm: The MTA presents the list of authentication methods it supports. For each element in *mech-list*, Anubis tests whether it is available in the list presented by MTA. If found, this method is selected. For example, suppose that the MTA supports following mechanisms:

```
PLAIN LOGIN CRAM-MD5 ANONYMOUS
```

and you have following statement in your configuration file

```
esmtp-allowed-mech DIGEST-MD5 CRAM-MD5 LOGIN
```

In this case Anubis will select CRAM-MD5.

`esmtp-require-encryption` *mech-list* [Option]
This statement declares the list of mechanisms that can be used only over a TLS encrypted channel. By default Anubis uses

```
esmtp-require-encryption LOGIN PLAIN
```

This prevents sending user password over an unencrypted connection.

`esmtp-auth-id` *authentication-id* [Option]
Sets authentication ID (user name).

`esmtp-authz-id` *authorization-id* [Option]
Sets authorization ID (user name).

`esmtp-password` *password* [Option]
Sets password to be used in authentication.

`esmtp-auth` *username:password* [Option]
This option sets both authentication and authorization IDs and the password. It is equivalent to

```
esmtp-auth-id username
esmtp-authz-id username
esmtp-password password
```

The following options specify authentication credentials for GSSAPI, DIGEST-MD5 and KERBEROS_V5 authentication mechanisms:

`esmtp-service` *service-name* [Option]
Sets the name of GSSAPI service.

`esmtp-hostname` *hostname* [Option]
Sets hostname of the machine.

`esmtp-generic-service` *servise-name* [Option]
Sets generic service name.

`esmtp-passcode` *passcode* [Option]
Sets passcode.

`esmtp-realm` *realm-name* [Option]
Sets GSSAPI realm.

Following option is useful with ANONYMOUS authentication mechanism:

`esmtp-anonymous-token` *token* [Option]
Sets the token to be used with ANONYMOUS authentication mechanism

4.2.5 Encryption Settings

`ssl` *yes-or-no* [Option]
This option enables the TLS/SSL encryption between the MUA and the MTA. Value '`no`' is the default, but using the TLS/SSL encryption is recommended. You should also specify a private key and a certificate using the '`ssl-key`' and '`ssl-cert`' keywords (defined below). See Chapter 8 [TLS/SSL], page 41, for details.

`ssl-oneway` *yes-or-no* [Option]
This option enables the *ONEWAY* encryption. Use this mode, when you want to use the TLS/SSL, but your MUA doesn't provide a support for ESMTP TLS/SSL. Using this option doesn't require using the '`ssl-key`' and '`ssl-cert`' keywords.

`ssl-cert` *file-name* [Option]
Specify a certificate for the TLS/SSL encryption. Value '`anubis.pem`' is the default.

`ssl-key` *`file-name`* [Option]
Specify a private key for the TLS/SSL encryption. Value '`anubis.pem`' is the default.

`ssl-cafile` *`file-name`* [Option]
Specify a CA certificate file (supported only by GnuTLS).

4.2.6 Security Settings

The following options control various security settings.

`allow-local-mta` *`yes-or-no`* [Option]
For security reasons, this option is set to '`no`', but the '`yes`' value enables the '`local-mta`' keyword (or '`--local-mta`' command line option), so if you want to use a local mail server, which works on standard input and output, a supervisor must set this option to '`yes`'. The option is available only in system configuration file.

`drop-unknown-user` *`yes-or-no`* [Option]
This option drops an unknown user, i.e. a client which has not been verified by IDENT service. Value '`no`' is the default.

`user-notprivileged` *`username`* [Option]
For security reasons, it is recommended to create an unprivileged user, which the server runs as most of the time, when doing unprivileged operations. The option is available only in system configuration file. For example:

```
user-notprivileged "anubis.unprivileged"
```

Caution: Create a user account named '`anubis.unprivileged`' in the '`/etc/passwd`', if necessary. Add this user name also to the '`/etc/anubis.allow`', if using GNU Anubis with PAM support.

`rule-priority` *`value`* [Option]
This statement defines the order of execution of the system and user `RULE` sections (See Chapter 5 [Rule System], page 23, for detailed description). It is available only in system configuration file.

`system` The system section is executed first, then the user section is executed.

`user` The user section is executed first, next the system section is executed.

`system-only`
Only the system `RULE` section is executed.

`user-only`
Only the user `RULE` section is executed.

`control-priority` *value* [Option]

Sets the order of processing the `CONTROL` sections. The option is available only in system configuration file. Its possible values are:

`system`
: The system `CONTROL` section is processed first. Notice, that this means that the user may override the system settings in his configuration file. This is the default setting.

`user`
: The user `CONTROL` section is processed first. Thus, the system-wide settings always override the user private settings.

4.3 TRANSLATION Section

The '`TRANSLATION`' section specifies how to translate remote or local user names, or host names or addresses, to local user names. The '`TRANSLATION`' section is available *only* in the system configuration file. Syntax:

```
---BEGIN TRANSLATION---
translate  [user@]address into  username
...
---END---
```

address means host name or IP address. You can also specify '`0.0.0.0`', and it means any address ('`INADDR_ANY`').

An example:

```
---BEGIN TRANSLATION---
translate jack@somewhere.net into john
---END---
```

The rule above will allow a remote user '`jack`' at '`somewhere.net`' to use the configuration file of the local user '`john`'. Or you can write: '`translate somewhere.net into john`', and this means that *all* users at '`somewhere.net`' are allowed to use the local john's configuration file.

4.4 GUILE Section

`guile-output` *file* [Command]

Specifies the name of the file to bind to the Scheme standard error and output ports. This option has no effect if GNU Anubis is started with either of '`--foreground`' or '`--stdio`' command line options.

`guile-debug` *yes-or-no* [Command]

When set to '`yes`' enables Guile stack traces and debugging output.

`guile-load-path-append` *path* [Command]

Appends the given *path* to the list of Guile load paths (see section "Build Config" in *The Guile Reference Manual*).

`guile-load-program` *file* [Command]

Reads the given Scheme program.

5 The Rule System

The rule system is a core part of GNU Anubis. It can be regarded as a program that is executed for every outgoing message.

Throughout this chapter, when showing syntax definitions, the optional parts of these will be enclosed in a pair of square brackets, e.g.:

```
keyword [optional-part] mandatory-part
```

When the square braces are required symbols, they will be marked as such, e.g.:

```
remove '['key']'
```

The rule system is defined in *RULE* section. The statements within this section are executed sequentially. Each statement is either an *action* or a *conditional statement*.

5.1 Actions

An *action* is a statement defining an operation to be performed over the message. Syntactically, each action is

```
command [=] right-hand-side
```

Where *command* specifies a particular operation and *right-hand-side* specifies the arguments for it. The equal sign is optional.

5.2 Conditional Statements

A *conditional statement* defines the control flow in the section. It allows to execute arbitrary actions depending on whether a certain condition is met. A conditional statement in its simplest form is:

```
if part [pattern-match-flags] cond-expr
  action-list-1
fi
```

The *part* specifies which part of the input should be considered when evaluating the condition. It is either '`command`', meaning the text of an smtp command issued while sending the message, or '`header`', meaning the value of an RFC822 header. Either of the two may be followed by the name of the corresponding command or header enclosed in square brackets. If this part is missing, all command or headers will be searched.

The optional *pattern-match-flags* alter the pattern matching type used in subsequent conditional expression. It will be described in detail in the section Section 5.5 [Regular Expressions], page 26. The *cond-expr* is a *conditional expression*. It consists of a series of *conditions* joined together with boolean operators '`and`' or '`or`' (see Section 5.4 [Boolean Operators], page 25). Each condition is:

= *regexp* Returns true if the requested part of the input matches the given regular expression (*regexp*).

!= *regexp* Returns true if the requested part of the input does not match the given regular expression.

not *condition*
Reverses the sense of *condition*

(*cond-expr*)
Returns the result of the conditional expression in parentheses. This is useful for changing operator precedence.

The simplest example:

```
if header [Subject] "^ *Re:"
  ...
fi
```

The actions represented by ... will be executed only if the 'Subject:' header of the message starts with 'Re:' optionally preceded by any amount of whitespace.

The more elaborate form of a conditional allows you to choose among the two different action sets depending on a given condition. The syntax is:

```
if part [flags] cond-expr
  action-list-1
else
  action-list-2
fi
```

Here, the *action-list-1* is executed if the condition *cond-expr* is met. Otherwise, *action-list-2* is executed.

```
if part [flags] cond-expr
  action-list-1
else
  action-list-2
fi
```

Note also, that in the examples above any of the statements *action-list* may contain conditionals, so that the conditional statements may be nested. This allows to create very sophisticated rule sets. As an example, consider the following statement:

```
if [List-Id] :re ".*<anubis-commit@gnu.org>"
  modify [Subject] "[Anubis Commit Notice] &"
else
  if [List-Id] :re ".*<bug-anubis@gnu.org>"
    modify [Subject] "[Anubis Bug Notice] &"
  else
    add [X-Passed] "Subject checking"
  fi
fi
```

This statement, depending on the value of `List-Id` header, will prepend the `Subject` header with an identification string, or add an `X-Passed` header if no known `List-Id` was found.

5.3 Triggers

Triggers are conditional statements that use the value of the '`Subject`' header to alter the control flow. Syntactically, a trigger is:

```
trigger [flags] pattern
  action-list
done
```

Here, *pattern* is the pattern against which the '`Subject`' header is checked, *flags* are optional flags controlling the type of regular expression used (see Section 5.5 [Regular Expressions], page 26). For backward compatibility, the keyword `rule` may be used instead of `trigger`.

The triggers act as follows: First, the value of the '`Subject`' header is matched against the pattern '`@@`'*pattern*. If it matches, then the matched part is removed from the '`Subject`', and the *action-list* is executed.

Basically, putting aside the possibility to use different flavors of regular expressions, a trigger is equivalent to the following statement:

```
if header[Subject] :posix "(.*)@@pattern"
  modify header [Subject] "\1"
  action-list
fi
```

Thus, adding the '`@@rule-name`' code to the '`Subject`' header of your message, triggers a rule named *rule-name*, specified in a user configuration file. For example:

```
---BEGIN RULE---
trigger :basic "^gpg-encrypt-john"
   gpg-encrypt "john's_gpg_key"
done
---END---
```

Now you can simply send an email with the following subject: '`hello John!@@gpg-encrypt-john`' to process an outgoing message with the rule specified above—encrypt message with a John's public key. Moreover, the trigger will remove the '`@@`', so John will only receive a message with a subject '`hello John!`'.

Another example shows an even more dynamic trigger, that is using a substitution and back-references:

```
---BEGIN RULE---
trigger :extended "^gpg-encrypt:(.*)"
   gpg-encrypt "\1"
   add [X-GPG-Comment] "Encrypted for \1"
done
---END---
```

To encrypt a message to user e.g. '`John`', simply send an email with a subject '`hello John!@@gpg-encrypt:john's_gpg_key`'. This way, you decide at a run time which public key should be used, without creating separate rules for each user; thanks to back-references, those 3—4 lines are enough.

5.4 Boolean Operators

The following table lists the three boolean operators that can be used in Anubis conditional expressions in the order of increasing binding strength:

- 'OR'
- 'AND'
- 'NOT'

As an example, let's consider the following statement:

```
if header[X-Mailer] "mutt" or header[X-Mailer] "mail" \
   and not header[Content-Type] "^multipart/mixed;.*"
   action
fi
```

In this case the *action* will be executed if the `X-Mailer` header contains the word '`mutt`'. The same *action* will also be executed if the `X-Mailer` header contains the word '`mail`' *and* the value of the `Content-Type` header does not begin with the string '`multipart/mixed`'.

Now, if we wished to execute the *action* for any message sent using `mail` or `mutt` whose `Content-Type` header does not begin with the string '`multipart/mixed`', we would write the following:

```
if (header[X-Mailer] "mutt" or header[X-Mailer] "mail") \
   and not header[Content-Type] "^multipart/mixed;.*"
   action
fi
```

Notice the use of parentheses to change the binding strength of the boolean operators.

5.5 Regular Expressions

GNU Anubis supports two types of regular expressions: POSIX (both basic and extended), and Perl-style regular expressions. Among this, the former are always supported, whereas the support for the latter depends on the configuration settings at compile time. The default type of regular expressions is POSIX Extended.

A number of modifiers is provided to change the type of regular expressions. These are described in the following table.

`:regex`
`:re`
: Indicates that the following pattern should be considered a regular expression. The default type for this expression is assumed.

`:perl`
`:perlre`
: The regular expression is a Perl-style one.

`:exact`
`:ex`
: Disables regular expression matching, all patterns will be matched as exact strings.

`:scase` Enables case-sensitive comparison.

`:icase` Enables case-insensitive comparison.

`:basic` Switches to the POSIX Basic regular expression matching.

`:extended`
Switches to the POSIX Extended regular expression matching.

The special statement `regex` allows you to alter the default regular expression type. For example, the following statement

```
regex :perl :scase
```

sets the default regular expression types to Perl-style, case-sensitive. The settings of `regex` statement regard only those patterns that appear after it in the configuration file and have force until the next occurrence of the `regex` statement.

A couple of examples:

```
if header[Subject] :perlre "(?<=(?<!foo)bar)baz"
  ...
fi
```

This will match any `Subject` header whose value matches an occurrence of '`baz`' that is preceded by '`bar`' which in turn is not preceded by '`foo`'.

```
if header[Subject] :scase "^Re"
```

will match a `Subject` header whose value starts with '`Re`', but will not match it if it starts with '`RE`' or '`re`'.

When using POSIX regular expressions, the extended syntax is enabled by default. If you wish to use a basic regular expression, precede it with the `:basic` flag.

For the detailed description of POSIX regular expressions, See section "Regular Expression Library" in *Regular Expression Library*. For information about Perl-style regular expressions, refer to the Perl documentation.

5.6 Action List

An *action list* is a list of action commands, which control processing of an outgoing messages. All action command names are case insensitive, so you can use for instance: '`add`' or '`ADD`' or '`AdD`', and so on.

5.6.1 Stop Action

The `stop` command stops immediately the processing of the section. It may be used in the main `RULE` section as well as in any user-defined section. For example:

```
if not header[Content-Type] "text/plain; .*"
  stop;
fi
```

5.6.2 Call Action

The `call` command allows to invoke a user-defined section much in the same manner as a subroutine in a programming language. The invoked section continues to execute until its end or the `stop` statement is encountered, whichever the first.

```
BEGIN myproc
if header[Subject] "Re: .*"
  stop;
fi
trigger "pgp"
  gpg-encrypt "my_gpg_key"
done
END

BEGIN RULE
call myproc
END
```

5.6.3 Adding Headers or Text

The `add` command allows you to add arbitrary headers or text to the message. To add a header, use the following syntax:

`add` *header* '['*name*']' *string* [Command]
`add` '['*name*']' *string* [Command]

For example:

```
add header[X-Comment-1] "GNU's Not Unix!"
add [X-Comment-2] "Support FSF!"
```

`add` *body* *text* [Command]

Adds the *text* to the message body. Use of this command with '`here document`' syntax allows to append multi-line text to the message, e.g.:

```
add body <<-EOT
    Regards,
    Hostmaster
    EOT
```

5.6.4 Removing Headers

The command `remove` removes the specified header from the message. The syntax is:

`remove` [*flags*] *header* '['*string*']' [Command]
`remove` [*flags*] '['*string*']' [Command]

The name of the header to delete is given by *string* parameter. By default only those headers are removed whose names match it exactly. Optional *flags* allow to change this behavior. See Section 5.5 [Regular Expressions], page 26, for the detailed description of these.

An example:

```
remove ["X-Mailer"]
remove :regex ["^X-.*"]
```

The first example will remove the '`X-Mailer:`' header from an outgoing message, and the second one will remove all "X-*" headers.

5.6.5 Modifying Messages

The action command `modify` allows to alter the headers or the body of the message.

`modify` [*flags*] *header* '[*key*]' '[*new-key*]' [Command]
`modify` [*flags*] '[*key*]' '[*new-key*]' [Command]

For each header whose name matches *key*, replaces its name with *new-key*. If *key* is a regular expressions, *new-key* may contain back references. For example, the following statement will select all headers whose names start with '`X-`' and change their names to begin with '`X-Old-`':

```
modify header :re ["X-\(.*\)"] ["X-Old-\1"]
```

`modify` [*flags*] *header* '[*key*]' *value* [Command]
`modify` [*flags*] '[*key*]' *value* [Command]

For each header whose name matches *key*, changes its value to *value*. For example:

```
modify [Subject] "New subject"
```

This statement sets the new value to the `Subject` header.

Every occurrence of unescaped '`&`' in the new value will be replaced by the old header value. To enter the '`&`' character itself, escape it with two backslash characters ('\\'). For example, the following statement

```
modify [Subject] "[Anubis \\& others] &"
```

prepends the `Subject` header with the string '`[Anubis & others]`'. Thus, the header line

```
Subject: Test subject
```

after having been processed by Anubis, will contain:

```
Subject: [Anubis & others] Test subject
```

`modify` [*flags*] *header* '[*key*]' '[*new-key*]' *value* [Command]
`modify` [*flags*] '[*key*]' '[*new-key*]' *value* [Command]

Combines the previous two cases, i.e. changes both the header name and its value, as shown in the following example:

```
modify header [X-Mailer] [X-X-Mailer] "GNU Anubis"
```

`modify` [*flags*] *body* '[*key*]' [Command]

Removes all occurrences of *key* from the message body. For example, this statement will remove every occurrence of the word '`old`':

```
modify body ["old"]
```

`modify` [*flags*] *body* '[*key*]' *string* [Command]

Replaces all occurrences of *key* with *string*. For example:

```
modify body :extended ["the old \([[:alnum:]]+\)"] "the new \1"
```

5.6.6 Inserting Files

`signature-file-append` *yes-or-no* [Command]

This action command adds at the end of a message body the '`-- `' line, and includes a client's '`~/.signature`' file. Value '`no`' is the default.

`body-append` *file-name* [Command]

This action command includes at the end of a message body the contents of the given file. If '*file-name*' does not start with a '`/`' character, it is taken relative to the current user home directory

`body-clear` [Command]

Removes the body of the message

`body-clear-append` *file-name* [Command]

Replaces the message body with the contents of the specified file. The action is equivalent to the following command sequence:

```
body-clear
body-append file-name
```

5.6.7 Mail Encryption

`gpg-passphrase` *passphrase* [Command]

Specifies your private key's pass phrase for signing an outgoing message using the GNU Privacy Guard (a tool compatible with the Pretty Good Privacy). Of course, to protect your passwords in the configuration file use the 0600 (u=rw,g=,o=) permissions, otherwise GNU Anubis won't accept them. We recommend setting the '`gpg-passphrase`' once in your configuration file, e.g. at the start of `RULE` section.

GNU Anubis supports the GNU Privacy Guard via the *GnuPG Made Easy* library, available at `http://www.gnupg.org/gpgme.html`.

`gpg-encrypt` *gpg-keys* [Command]

This command enables encrypting your outgoing message with the GNU Privacy Guard (Pretty Good Privacy) public key(s). *gpg-keys* is a comma separated list of keys (with no space between commas and keys).

```
gpg-encrypt "John's public key"
```

`gpg-sign` *gpg-signer-key* [Command]
`gpg-sign` '`yes-or-default`' [Command]

This command signs the outgoing message with your GNU Privacy Guard private key. Specify a *passphrase* with `gpg-passphrase`. Value '`default`' means your default private key, but you can change it if you have more than one private key.

For example:

```
gpg-sign default
```

or

```
gpg-passphrase "my office key passphrase"
gpg-sign office@example.key
```

`gpg-sign-encrypt` *gpg-keys[:gpg-signer-key]* [Command]
`gpg-se` *gpg-keys[:gpg-signer-key]* [Command]

This command simultaneously signs and encrypts your outgoing message. It has the same effect as `gpg` command line switch '`-se`'. The argument before the colon is a comma-separated list of PGP keys to encrypt the message with. This argument is mandatory. The second argument is optional and is separated from the first one by a colon ('`:`'). This argument specifies the signer key. In the absence of the second argument your default private key is used.

For example:

```
gpg-sign-encrypt John@example.key
```

or

```
gpg-se John@example.key:office@example.key
```

5.6.8 Using an External Processor

`external-body-processor` *program [args]* [Command]

Pipes the message body through *program*. *program* should be a filter program, that reads the text from the standard input and prints the transformed text on the standard output. The output from the *program* replaces the body of the message. *args* are any additional arguments the program may require.

5.6.9 Quick Example

Here is a quick example of using an action list:

```
---BEGIN RULE---
if header [X-Mailer] :re ".*"
   remove [X-Mailer]
   add [X-Comment] "GNU's Not Unix!"
   gpg-sign "my password"
   signature-file-append yes
fi
---END---
```

The example above will remove (on-the-fly) the '`X-Mailer:`' line from an outgoing message, add an extra header line ('`X-Comment:`'), sign your message with your private key, and add a simple signature file from your home directory.

5.7 Using Guile Actions

The name Guile stands for *GNU's Ubiquitous Intelligent Language for Extensions*. It provides a Scheme interpreter conforming to the R4RS language specification. GNU Anubis uses Guile as its extension language.

This section describes how to write GNU Anubis actions in Scheme. It assumes that the reader is sufficiently familiar with the Scheme language. For information about the language, refer to section "Top" in *Revised(4) Report on the Algorithmic Language Scheme*. For more information about Guile, See section "Overview" in *The Guile Reference Manual*.

5.7.1 Defining Guile Actions

A Guile action is defined as follows:

```
(define (function-name header body . rest)
 ...)
```

Its arguments are:

header List of message headers. Each list element is a cons

```
(name . value)
```

where *name* is the name of the header field, and *value* is its value with final CRLF stripped off. Both *name* and *value* are strings.

body A string containing the message body.

rest Any additional arguments passed to the function from the configuration file (see Section 5.7.2 [Invoking Guile Actions], page 33). This argument may be absent if the function is not expected to take optional arguments.

The function must return a cons whose car contains the new message headers, and cdr contains the new message body. If the car is `#t`, it means that no headers are changed. If the cdr is `#t`, it means that the body has not changed. If the cdr is `#f`, Anubis will delete the entire message body.

As the first example, let's consider a *no-operation* action, i.e. an action that does not alter the message in any way. It can be written in two ways:

```
(define (noop-1 header body)
  (cons header body))

(define (noop-2 header body)
  (cons #t #t))
```

The following example is a function that deletes the message body and adds an additional header:

```
(define (proc header body)
  (cons (append header
                (cons "X-Body-Deleted" "yes"))
        #f))
```

Let's consider a more constructive example. The following function checks if the `Subject` header starts with string '`ODP:`' (a Polish equivalent to '`Re:`'), and if it does, the function replaces it with '`Re:`'. It always adds to the message the header

```
X-Processed-By: GNU Anubis
```

Additionally, if the optional argument is given, it is appended to the body of the message.

```
(define (fix-subject hdr body . rest)
  "If the Subject: field starts with characters \"ODP:\", replace
them with \"Re:\".
If REST is not empty, append its car to BODY"
  (cons (append
 (map (lambda (x)
(if (and (string-ci=? (car x) "subject")
 (string-ci=? (substring (cdr x) 0 4) "ODP:"))
    (cons (car x)
  (string-append "Re:"
 (substring (cdr x) 4)))
    x))
      hdr)
 (list (cons "X-Processed-By" "GNU Anubis")))
(if (null? rest)
    #t
    (string-append body "\n" (car rest)))))
```

5.7.2 Invoking Guile Actions

The Guile actions are invoked from the `RULE` section using the `guile-process` command. Its syntax is:

function args [Scheme Function]

Arguments:

function The name of the Guile function to be invoked.

args Additional arguments. These are passed to the *function* as its third argument (*rest*).

To pass keyword arguments to the function, use the usual Scheme notation: '`#:key`'.

As an example, let's consider the invocation of the `fix-subject` function, defined in the previous subsection:

```
guile-process fix-subject <<-EOT
                                ----------
                                Kind regards,
                                Antonius Block
                          EOT
```

In this example, the additional argument (a string of three lines) is passed to the function, which will add it to the message of the body.

5.7.3 Support for ROT-13

The ROT-13 transformation is a simple form of encryption where the letters A-M are transposed with the letters L-Z. It is often used in Usenet postings/mailing lists to prevent people from accidentally reading a disturbing message.

GNU Anubis supports ROT-13 via a loadable Guile function. To enable this support, you will have to add the following to your `GUILE` section:

```
guile-load-program rot-13.scm
```

Then, in your `RULE` section use:

`rot-13` *keyword-arguments* [Scheme Function]

The command accepts the following *keyword-arguments*:

`#:body` Encrypt the entire body of the message

`#:subject`
Encrypt the '`Subject`' header.

For example:

```
trigger "rot-13.*body"
 guile-process rot-13 #:body
done

trigger "rot-13.*subj"
 guile-process rot-13 #:subject
done
```

5.7.4 Remailers Type-I

GNU Anubis supports remailers of type I. The support is written entirely in Scheme. To enable it you need to specify the following in the `GUILE` section of your configuration file:

```
guile-load-program remailer.scm
```

To send the message via a remailer, use the following command in the `RULE` section:

`remailer-I` *keyword-arguments* [Scheme Function]

The *keyword-arguments* specify the various parameters for the remailer. These are:

`#:rrt` *string*
This is the only required keyword argument. It sets the value for the *Request Remailing To* line. *string* should be your actual recipient's email address.

`#:post` *news-group*
Adds the '`Anon-Post-To:` *news-group*' line, and prepares the message for sending it to the Usenet via a remailer. Note, that this is only possible with remailers that support '`Anon-Post-To:`' header.

`#:latent` *time*
Adds the '`Latent-Time:`' line, that causes a remailer to keep your message for specified *time* before forwarding it.

`#:random` Adds random suffix to the latent time.

`#:header` *string*
Adds an extra header line to the remailed message.

Example:

```
trigger "remail:(.*)/(.*)"
 guile-process remailer-I \
             #:rrt antonius_block@helsingor.net \
             #:post \1 \
             #:latent \2 \
             #:header "X-Processed-By: GNU Anubis & Remailer-I"
done
```

Some remailers require the message to be GPG encrypted or signed. You can achieve this by placing `gpg-encrypt` or `gpg-sign` statement right after the invocation of `remailer-I`, for example:

```
trigger "remail:(.*)/(.*)"
 guile-process remailer-I \
             #:rrt antonius_block@helsingor.net \
             #:post \1 \
             #:latent \2 \
             #:header "X-Processed-By: GNU Anubis & Remailer-I"
 gpg-sign mykey
done
```

See Section 5.6.7 [Mail Encryption], page 30, for more information on mail encryption in GNU Anubis.

5.7.5 Entire Message Filters

There may be some cases when you need to use an external filter that processes the entire message (including headers). You cannot use `external-body-processor`, since it feeds only the message body to the program. To overcome this difficulty, GNU Anubis is shipped with '`entire-msg.scm`' module. This module provides Scheme function `entire-msg-filter`, which is to be used in such cases.

`entire-msg-filter` *program* [*args*] [Scheme Function]

Feeds entire message to the given program. The output from the program replaces message headers and body.

progname Full pathname of the program to be executed.

args Any additional arguments it may require.

Suppose you have a program `/usr/libexec/myfilter`, that accepts entire message as its output and produces on standard output a modified version of this message. The program takes as its argument he name of a directory for temporary files. The following example illustrates how to invoke this program:

```
BEGIN GUILE
guile-load-program entire-msg.scm
END

SECTION RULE
guile-process entire-msg-filter /usr/libexec/myfilter /tmp
```

```
END
```

Another function defined in this module is `openssl-filter`:

`openssl-filter` *program* [*args*] [Scheme Function]

This function is provided for use with `openssl` program. `Openssl` binary attempts to rewind its input and fails if the latter is a pipe, so `openssl` cannot be used with `entire-msg-filter`. Instead, you should use `openssl-filter`. Its arguments are:

program Path to `openssl` binary.

args Its arguments

See Chapter 9 [S/MIME], page 43, for an example of use of this function.

6 Invoking GNU Anubis

The `anubis` executable acts like a daemon. The behavior of program is controlled by two configuration files, which have a **higher** priority than command line options. See Chapter 4 [Configuration], page 15, for details.

GNU `anubis` supports the following command line options:

'`--altrc` *file*'
: Specify alternate system configuration file.

'`--bind [`*host*`:]`*port*'
'`-b`'
: Specify the TCP port on which GNU Anubis listens for connections. The default *host* value is '`INADDR_ANY`', and default *port* number is 24 (private mail system).

'`--check-config[=`*level*`]`'
'`-c[`*level*`]`'
: Run the configuration file syntax checker. Optional *level* specifies the verbosity level. The following levels are allowed:

    0 Display only errors. This is the default.

    1 Print the syntax tree after parsing the file.

    2 As '`1`', but also prints the parser traces.

    3 As '`2`', but also prints the lexical analyzer traces.

'`--debug`'
'`-D`'
: Debug mode.

'`--foreground`'
'`-f`'
: Foreground mode.

'`--help`'
: Print short usage summary and exit.

'`--local-mta` *file*'
'`-l`'
: Execute a local SMTP server, which works on standard input and output (inetd-type program). This option excludes the '`--remote-mta`' option.

'`--mode` *mode-name*'
'`-m` *mode-name*'
: Selects Anubis operation mode. Allowed values for *mode-name* are '`transparent`' (default) and '`auth`'. See Chapter 3 [Authentication], page 5, for the detailed discussion of Anubis operation modes.

'`--norc`'
: Ignore system configuration file.

'`--relax-perm-check`'
: Do not check a user config file permissions.

'--remote-mta *host*[:*port*]'
'-r' Specify a remote SMTP host name or IP address, which GNU Anubis will connect and forward mail to (after a processing). The default *port* number is 25.

'--silent'
'-s' Work silently.

'--show-config-options'
Print a list of configuration options used to build GNU Anubis.

'--stdio'
'-i' Use the SMTP protocol (OMP/Tunnel) as described in RFC 821 on standard input and output.

'--verbose'
'-v' Work noisily.

'--version'
Print version number and copyright.

Examples:

```
$ anubis --remote-mta smtp-host:25
```

Run GNU Anubis on port number 24 (private mail system). Note that you must have root privileges to use port number lower than 1024. Make the tunnel between your localhost:24 and *smtp-host*:25.

```
$ anubis -f --remote-mta smtp-host:25
```

Same as above, but run GNU Anubis in a foreground mode.

```
$ anubis -f --local-mta /usr/sbin/sendmail -- sendmail -bs
```

Similar to above, but create a tunnel between localhost:24 and a local program (local MTA). In this example local program is `sendmail` with '-bs' command line option. The '-bs' option forces `sendmail` to work on standard input and output.

```
$ anubis --norc --remote-mta smtp-host:25
```

Do not read the system configuration file, make the tunnel between localhost:24 and *smtp-host*:25.

```
$ anubis --bind localhost:1111 --remote-mta smtp-host:25
```

Create the tunnel between localhost:1111 and *smtp-host*:25.

```
$ anubis -i
```

Use the SMTP protocol (OMP/Tunnel) as described in RFC 821 on standard input and output.

7 Sample Beginning

By default, GNU Anubis binds to port number 24 (private mail system), so there shouldn't be any conflict with your local MTA (Mail Transport Agent). You just have to reconfigure your MUA (Mail User Agent) to make it talk to GNU Anubis directly on port number 24. All MUAs are normally set up to talk directly to the MTA, so you must change their settings and specify GNU Anubis' port number as their target. This makes GNU Anubis to work as an outgoing mail processor between your MUA and the MTA. Read your MUA's documentation for more information.

Now you must choose whether you want to connect GNU Anubis with a remote or local SMTP host via TCP/IP or a local SMTP program, which works on standard input and output. In the first case, specify the following option:

```
REMOTE-MTA smtp-host:25
```

In the second case (local SMTP program), specify this:

```
LOCAL-MTA /path/to/your/mta/mta-executable -bs
```

Please note that the '`-bs`' command line option is a common way to run MTAs on standard input and output, but it is not a rule. Read your local MTA's documentation, how to get it working on standard input and output.

If you would like to run GNU Anubis on port number 25 (which is a default value for the SMTP) or any other port number, then you must specify the '`bind`' keyword. For instance, the following code will bind GNU Anubis to '`localhost:25`':

```
BIND localhost:25
```

This can make a conflict between GNU Anubis and your local MTA, which usually listens on port number 25. To solve this problem, you can for instance disable the MTA and specify the '`local-mta`' keyword, or run MTA on port number different than GNU Anubis' port number (e.g. 1111). Please read your local MTA's documentation about this topic. For example:

```
BIND localhost:25
REMOTE-MTA localhost:1111
```

Caution: Make sure that your local machine doesn't accept any incoming mail (i.e. it is *not* a POP or IMAP server), otherwise you cannot disable your MTA or change its port number!

All Mutt users, who would like to set up GNU Anubis between their MUA and MTA, should consider using the '`msg2smtp.pl`' Perl script from the '`contrib`' directory (part of the distribution).

8 Using the TLS/SSL Encryption

According to the RFC 2246 document, the TLS (Transport Layer Security) protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. The primary goal of the TLS Protocol is to provide privacy and data integrity between two communicating applications. The TLS protocol itself is based on the SSL 3.0 (Secure Socket Layer) protocol specification.

GNU Anubis supports the TLS/SSL (via the GnuTLS, a Transport Layer Security Library available at `http://www.gnutls.org/`, or OpenSSL, a cryptographic package available at `http://www.openssl.org/`), but your MTA must provide the STARTTLS command first. This can be checked by:

```
$ telnet your-smtp-host 25
  ehlo your-domain-name
```

The server will response with all its available commands. If you see the STARTTLS, then you can use the TLS/SSL encryption. If your MUA doesn't support the TLS/SSL encryption, but your MTA does, then you should use the '`oneway-ssl`' keyword in your configuration file. Before using the TLS/SSL encryption, you must generate a proper private key and a certificate. You can do it simply with:

```
$ cd anubis-directory
$ ./build/keygen.sh
```

This will create the '`anubis.pem`' file. For example copy this file to '`/usr/share/ssl/certs/`'. Next, edit your configuration file by adding:

```
ssl yes
ssl-key path-to-the-private-key
ssl-cert path-to-the-certificate
```

For example:

```
ssl-key /usr/share/ssl/certs/anubis.pem
ssl-cert /usr/share/ssl/certs/anubis.pem
```

Caution: Each client can specify its own private key and a certificate by adding the '`ssl-key`' and '`ssl-cert`' keywords in its own user configuration file.

See Section 4.2.5 [Encryption Settings], page 20, for details.

9 Using S/MIME Signatures

Anubis version 4.0 does not yet provide built-in support for S/MIME encryption or signing. To encrypt or sign messages using S/MIME, you will have to use external programs. Usually such programs require the whole message as their input, so simply using `external-body-processor` will not work. GNU Anubis distribution includes a special Guile program, '`entire-msg.scm`', designed for use with such programs. For its detailed description, please refer to Section 5.7.5 [Entire Message Filters], page 35. This chapter addresses a special case of using it with `openssl` to sign outgoing messages.

To use `openssl` for S/MIME signing, invoke it using `openssl-filter` function defined in '`entire-msg.scm`'. You will have to supply at least `-sign` and `-signer` arguments to the program. Notice, that you should not specify any input or output files.

The following example illustrates this approach:

```
BEGIN GUILE
guile-load-program entire-msg.scm
END

BEGIN RULE
guile-process openssl-filter /usr/local/ssl/bin/openssl \
              smime -sign -signer FILE
END
```

10 Using Mutt with Anubis

At the time of this writing `mutt`[1]is not able to send mail via SMTP channell, instead it invokes local mailer program to transmit the message. There are at least three possible ways to overcome this difficulty:

1. Using `mail.remote` from GNU mailutils
2. Using `msg2smtp.pl` script provided with Anubis
3. Using a patch by Steven Engelhardt (`patch-`*`version`*`.sde.libesmtp.3`) that enables `mutt` to use SMTP.

The following sections discuss each method in detail.

10.1 Using GNU mailutils as an interface to mutt

GNU Mailutils is a collection of utilities for handling electronic mail. It includes lots of programs necessary for dealing with e-mail messages. One of them is `mail.remote`, which is designed as a drop-in replacement for `sendmail` to forward all mail directly to an SMTP gateway. Its interface is compatible with `sendmail` which makes the program especially useful as an interface between `mutt` and `anubis`. The package can be downloaded from `ftp://ftp.gnu.org/gnu/mailutils` or any of the mirrors (See `http://www.gnu.org/order/ftp.html` for a complete list of these. Please, select the mirror closest too you). The complete information about the package is available from its home page at `http://www.gnu.org/software/mailutils/`

To use `mail.remote`, first download and install GNU mailutils (as usual the package is shipped with files '`README`' and '`INSTALL`' which provide the necessary guidelines). Then add to your '`.muttrc`' file the following line:

```
set sendmail="mail.remote smtp://hostname[:port]"
```

where *mail.remote* stands for the full file name of `mail.remote` utility, *hostname* and optional *port* specify the host name (or IP address) of the machine running `anubis` and the port it listens on. Notice, that default port value for `mail.remote` is 25, which means that in most cases you will have to specify it explicitely.

For example, suppose you run `anubis` on machine '`anubis.domain.org`' and that it listens on port 24. Let's also assume you have installed mailutils in the default location, so that full file name of `mail.remote` is '`/usr/local/libexec/mail.remote`'. Then, your '`.muttrc`' will contain:

```
set sendmail="/usr/local/libexec/mail.remote \
              smtp://anubis.domain.org:24"
```

(the line being split for readability).

[1] versions 1.4.1 and 1.5.3

10.2 Using msg2smtp.pl as an interface to mutt

GNU Anubis is shipped with `msg2smtp.pl` — a perl script designed as an interface between it and `mutt`. The script is kindly contributed by Michael de Beer.

The script is located in the subdirectory '`contrib`' of GNU Anubis distribution. To use it:

1. Make sure its first line correctly refers to the full file name of the `perl` interpreter on your system. By default the first line reads

   ```
   #!/usr/bin/perl
   ```

   If the file name after '`!`' differs from the actual file name of the `perl` interpreter, update it. For example, if `perl` is installed in '`/usr/local/bin/perl`', the first line of `msg2smtp.pl` should read

   ```
   #!/usr/local/bin/perl
   ```

2. Copy the script to any convenient location. Simply running `cp` will do, e.g.

   ```
   cp anubis-4.0/contrib/msg2smtp.pl /usr/local/libexec
   ```

3. Add to your '`.muttrc`' the following line:

   ```
   set sendmail="/usr/local/libexec/msg2smtp.pl -h hostname -p port"
   ```

   where *hostname* and *port* specify the host name (or IP address) of the machine running `anubis` and the port it listens on, respectively.

Complete description of `msg2smtp.pl` and a discussion of its command line switches can be found in file '`contrib/msg2smtp.txt`'.

10.3 Patching mutt

Steven Engelhardt modified `mutt` so that it is able to use SMTP to transfer messages. For the time being the patch is not accepted by the mainline `mutt` distribution, but one of the authors of GNU Anubis[2], has tested it extensively and has found it to be quite adequate for interfacing between `anubis` and `mutt`. The patch is described in detail at `http://www.deez.info/sengelha/projects/mutt/libesmtp/` and is available for `mutt` versions 1.4.x and 1.5.3.

To use it, follow the instructions on the page mentioned above. Once you compile the patched `mutt` you will be able to use the following new keywords in its configuration file:

`set smtp_host = `*`hostname`*
: Sets the hostname or IP address of the remote SMTP host.

`set smtp_port = `*`port`*
: Sets the port number to use.

`set smtp_auth_username = `*`user-name`*
: Sets the username to use with SMTP AUTH command (optional).

[2] Sergey Poznyakoff, blame it on him:^)

So, assuming you run `anubis` on machine '`anubis.domain.org`' and it is listening on port 24, you will add to your '`.muttrc`' the following two lines:

```
set smtp_host = anubis.domain.org
set smtp_port = 24
```

10.4 Comparison of the Three Interface Methods

The following short discussion summarizes the advantages and deficiencies of the three interface methods described in the previous sections. It could serve you as a guideline on which interface method to choose.

Using mail.remote

Advanatages:

1. Does not require modifying `mutt`.
2. Is compatible with any version of `mutt`.
3. Runs faster than `msg2smtp.pl`

Deficiences:

1. Running an external program to transmit the message is not the best idea. However, it is mutt default, anyway...
2. Runs slower than directly connecting to `anubis` using SMTP

Using msg2smtp.pl

Advantages:

1. Does not require modifying `mutt`.
2. Is compatible with any version of `mutt`.

Deficiences:

1. See [extprog], page 47.
2. Runs slower than the other two methods (sending each message requires loading `perl` interpreter, which is rather expensive).

Using patch.sde.libesmtp.3

Advantages:

1. Is the fastest of the three methods.
2. Does not require any intermediate programs.

Deficiences:

1. Requires patching `mutt`, which is not always possible or acceptable.
2. May not work for versions of mutt newer than 1.5.3 (but then, again, not necessarily so).

11 Reporting Bugs

Please send any bug reports, improvements, comments, suggestions, or questions to `bug-anubis@gnu.org`.

Before reporting a bug, make sure you have actually found a real bug. Carefully reread the documentation and see if it really says you can do what you are trying to do. If it is not clear whether you should be able to do something or not, report that too; it's a bug in the documentation!

12 Pixie & Dixie

- Introduction

  This document describes a new scheme for client authentication and authorization in GNU Anubis 4.x.
- Task Description

  So far the only authentication method used by Anubis was based on the AUTH protocol (RFC 1413) (`ftp://ftp.rfc-editor.org/in-notes/rfc1413.txt`), and thus required client party to use a popular daemon `identd`, which listens on TCP port 113 for authentication requests. As its primary advantage, this method allows to quickly identify whom the server had to deal with, i.e. to obtain user name or his UID. Actually, the authentication process finishes before the client sends over his first byte. Besides, this method allows to process the entire SMTP envelope. It has, however, several drawbacks, first of them being the requirement to run `identd` on the client machine, which is not always possible (e.g. on mobile devices), and may be considered harmful for the system security (due to sending user ID over the wire).
- The Proposed Solution

  Proposed are two operation modes:
  1. *Traditional* or *transparent* (also known as *Pixie* ;-)
  2. *Authentication first* (also known as *Dixie* ;-)

  A short description of each mode follows:
  - '`Pixie`' mode
    - Server requires the remote party to authenticate itself using SMTP AUTH (RFC 2554) (`ftp://ftp.rfc-editor.org/in-notes/rfc2554.txt`).
    - Early processing of SMTP envelope is possible.
    - Connections between MUA and MTA are tunneled "on the fly"
  - '`Dixie`' mode In this mode GNU Anubis runs its own user database, additionally translating logins (see [login translation], page 52). It also is able to keep users' configuration files (an additional option and an advantage — see [anubis database], page 52).

  Users are authenticated using ESMTP AUTH protocol. Early processing of SMTP envelope is not possible in this mode , instead it becomes possible only after the authentication is finished successfully. This mode also delays connecting to the MTA, since Anubis first has to perform ESMTP AUTH, and only after finishing authentication, does it read and process the user's configuration file and connects to the selected MTA. Of course, the client is not able to begin sending messages until he is authenticated and accepted by Anubis.

- Details

  There is a great difference between the two modes. To begin with, 'Pixie' mode provides a tunnel (or proxy), in the sense that Anubis connects user's MUA to the remote MTA without requiring any special actions from the user.

  Let's consider a simple interaction between 'Machine-A', which runs Anubis 4, and 'Machine-B', where MUA is run.

```
A: 220 Machine-A (GNU Anubis vX.X [Dixie]) ESMTP time; send your identity!
B: EHLO Machine-B
A: 250-Machine-A Hello ID
250-STARTTLS
250-AUTH DIGEST-MD5 CRAM-MD5 LOGIN
250-XDATABASE
250 HELP
B: STARTTLS
A: 220 2.0.0 Ready to start TLS
<TLS>
B: AUTH <METHOD>
[method-specific authentication interchange follows]
```

  Now, the Anubis server has authenticated the client using data from Anubis database! I'd like this database to contain, beside the user name and password, the name and password of this user on Machine-A.

  Confusing? Let's suppose that the database contains following record:

```
JohnSmith encrypted-pass-1  John
```

  The user has authenticated himself as 'JohnSmith' with password 'encrypted-pass-1', using ESMTP AUTH, and the given credentials matched those from the Anubis database. Now, Anubis, which has been running with super-user privileges, switches to UID of the user 'John'.

  Such solution will allow for a very flexible database, that would ease the administration tasks, since users will be able to update their corresponding records (of course, if the system administrator grants them such privileges). For instance, ODBC, SQL?

  Let's return to our sample session. After successful authentication and switching to the user's privileges, Anubis parses file '~/.anubisrc'. Then, based on user's configuration settings, it connects to the MTA and from then on operates as SMTP tunnel and mail processor :-). It sends the following response to 'Machine-B':

```
A: 220 OK, Welcome. Continue sending your mail!
```

- Further details

  The above description shows that it is impossible to use both 'Pixie' and 'Dixie' simultaneously. It is the responsibility of the system administrator to decide which operation mode to use. We could probably provide for a smooth switching between the two modes, without requiring to restart the daemon... However, it is not critical. Restarting the daemon in order to switch to another operation mode is also a feasible solution.

Now, let me describe for what kind of users each mode is intended.

The traditional (‘`Pixie`’) mode is intended for those users who use Anubis on a single machine or within a local network that allows to use `identd`. In short, ‘`Pixie`’ is useful when the use of `identd` is possible and safe.

In contrast, the new mode ‘`Dixie`’ is intended for more complex setups, where a single machine running GNU Anubis serves a number of clients connecting from different machines and networks. It is supposed that no client machine is running `identd`. The only recommendation for this mode is that each user have a system account on the machine running Anubis. But then, even this is not required!

That’s a feature I haven’t described yet :^) As described above, Anubis database must contain second login name in order for Anubis to be able to switch to the user’s privileges and parse his ‘`~/.anubisrc`’ file. Now, I supposed that the database is able to keep user configuration files as well. So, each database record must contain an additional flag informing Anubis whether it should read the local file ‘`~/.anubisrc`’, or read the configuration file stored in the database. Sure enough, GNU Anubis still will have to switch to the user’s privileges, for security reasons, but this can be done using usual `user-notprivileged` configuration (see Section 4.2.6 [Security Settings], page 21).

Surely you have noticed that in its response to EHLO command Dixie returned `250-XDATABASE` capability. Yes, this is exactly that command that I’d like to be used for remote management of the database records (after having successfully passed ESMTP AUTH).

Available operations are: `ADD`, `MODIFY`, `REMOVE`, meaning addition, modification and removal of a user record, and `UPLOAD`, providing a way to upload the user’s configuration file ‘`~/.anubisrc`’.

This solution will free the users from the obligation to have ‘`~/.anubisrc`’ on the server machine, so they, for the first time since early Anubis versions, will be able to have their *own* configuration files. Current versions () require that the user configuration file be stored on the server machine before the user is able to use the service. This approach requires a certain attention from the system administrator. Should the user wish to change something in his configuration file, he would have to install the modified file on ‘`Machine-A`’ (that’s how it works now, and that’s how it will continue to work for ‘`Pixie`’ mode). The new ‘`Dixie`’ mode solves this and frees the user from necessity to contact the system administrator of ‘`Machine-A`’. The Anubis database engine is supposed to check the correctness of the uploaded configuration file and inform the client about the result. It also should compute MD5 hash of the file and compare it to the one sent by the user... What for?

- A program sending user's configuration file

  Well, we're almost finished. The user will have a small program, `config-sender`, written in whatever language (C, Java, C#), whose main purpose is to send user's configuration file to the database. Such a program could even be installed on a mobile device! Notice also, that this program is optional, the user is not required to use it. I envision a situation where:

  1. A user logs in to his account on '`Machine-B`'
  2. His '`~/.profile`' invokes `config-sender` program. This program, in turn, computes MD5 sum of the local '`~/.anubisrc`' file and sends it to Anubis. There it will be compared to the sum kept in the Anubis database, and if the two sums differ, the `config-sender` will upload the contents of '`~/.anubisrc`'...[1]
  3. The `config-sender` program will, of course, connect to the Anubis database using ESMTP (TLS/AUTH) and `XDATABASE`.

  Such a program will be an additional advantage, since no existing MUA is, of course, able to use `XDATABASE` command to manage Anubis database. Notice however, that GNU Hydrant (`http://savannah.gnu.org/projects/hydrant`) will probably support `XDATABASE` in the future...

- The End.

  Thus, the user will simply use his MUA, no identd, no hassle :)

  Actually, the only requirement for the MUA is that it support ESMTP AUTH. Unfortunately, some MUA, even on UNIX-like systems, are still not able to use ESMTP AUTH. But in this case, the user can install Anubis on his machine and use it to perform authentication ;-)))

  And the last detail: what to do if the remote MTA also requires ESMTP AUTH? The answer is quite simple: GNU Anubis is already able to handle this (see Section 4.2.1 [Basic Settings], page 17).

- Summary ('`Dixie`' mode)
  - a little slower than '`Pixie`', in the sense that the actual connection to the MTA is established only after successful authentication
  - does not require `identd`!
  - allows the user full control over his configuration settings

[1] The scheme implemented currently is a bit different. First, the `config-sender` program issues an `EXAMINE` command that fetches the contents of the user configuration file from the server. Then, it compares it with the local copy kept on the client machine. If the copies differ, `config-sender` issues `UPLOAD` and thus updates the configuration on the server.

  - delays processing of SMTP envelope until after successful authentication.

- PS: A couple of words about storing configuration files in the database...

  These can be stored in a special directory as usual files, then each database record will have an additional field with the name of the configuration file for the given user.

— THE END —

Appendix A GNU Free Documentation License

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

   B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

   C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

   D. Preserve all the copyright notices of the Document.

   E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

   F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

   G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

   H. Include an unaltered copy of this License.

   I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warrany Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.0.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Concept Index

L

M

O

P

R

S

T

U

Short Contents

Table of Contents