Sympa, mailing-lists software

March 2002

Serge Aumont & Olivier Salaün
CRU

Vincent Mathieu
University of Nancy 2

Translation : Mathieu Jan (INSA Rennes)

Sympa is an outcome of our investments to meet the needs of a national service for French universities in the field of mailing-lists. This activity started in 1992 to prepare the migration of the *bitnet* lists. Today, Sympa is a product under the GPL license whose developments continue under the control of the CRU and with many contributions from the users community. More than 3000 sites of any origins load regularly new versions of SYMPA , which in French stands for **SY**stème de **M**ulti-**P**ostage **A**utomatique (**SYMPA)**.

It would be useless to rewrite the reference handbook of Sympa, but setting up Sympa is often considered to be complex by list masters. It is of course due to its great number of functionalities. The goal of this course is to help you to use all the devices of Sympa aiming to simplify its administration while offering improved services.

This course was given in March 2002, it is based on version 3.3.4 of Sympa.

1 Overview of Sympa functionalities

The point is not to enumerate all the specificities of this mailing-lists server, at most we can give an outline of its original points.
One of the characteristics of Sympa is to offer two integrated interface (e-mail and web). The web interface is unique and delivers to each one a personalized sight of the lists service. Thus, users, owners of lists and list master*(s)* use the same interface.

1.1 Services to users

Sympa proposes to mailing-lists users all the panoply of casual services in this field : subscription, subscription withdrawal, subscription options to receive messages under various formats, indexed archives, presented by date or by thread, web space enabling deposit of documents reserved to list subscribers , and so on.
It should be stressed that the service is accessible by two interfaces :

- The e-mail interface which recognizes the command sets of most used robots and adapts without difficulties to messages formatted with all the MIME possibilities (e-mail recognition of commands placed in multipart/alternative messages) ;
- The web interface, presented under the shape of a mailing-lists gateway of the site. It proposes to each one a personalized view of the mailing-lists service as soon as the person identifies herself through a device of passwords assignement/re-assignement. The user can then see the whole sets of his subscriptions, manage his preferences (choice of a language for interface for example), remove his message from archives and so on.

Each subscriber can choose for each list, subscription options like receiving periodic and grouped articles, replacement in messages of attachments by URLs, exclusive reception of ASCII text version or HTML version when both are contained in a message.

1.2 Services to owners and moderators of lists

Sympa distinguishes moderators (who authorize or refuse broadcasting messages in the lists) from owners (who configure the list and manage the subscriptions). Authenticated by his password, the list owner has on the web interface a specific view of his lists. For each one he can :

- Write service messages (welcome message, subscription recall, withdrawal message …) and the list greeting page ;
- Access and set parameters of his list (those the list master has made editable) in particular define the people authorized to subscribe to his list, to distribute message and son on ;
- Validate or refuse (by e-mail or web) the subscription requests and the messages, both subjected to validation ;

1.3 Services to list masters

One or more list masters are defined for each lists robot (Sympa enables to manage several lists robot relating to different internet domains). The list master validates list creation requests on the web interface when they are subjected to authorization. He sets up the default behaviors of Sympa as well as the service messages nonspecific to a list. He amongst other things, defines the people authorized to create mailing lists and so on.

1.4 Other features of Sympa

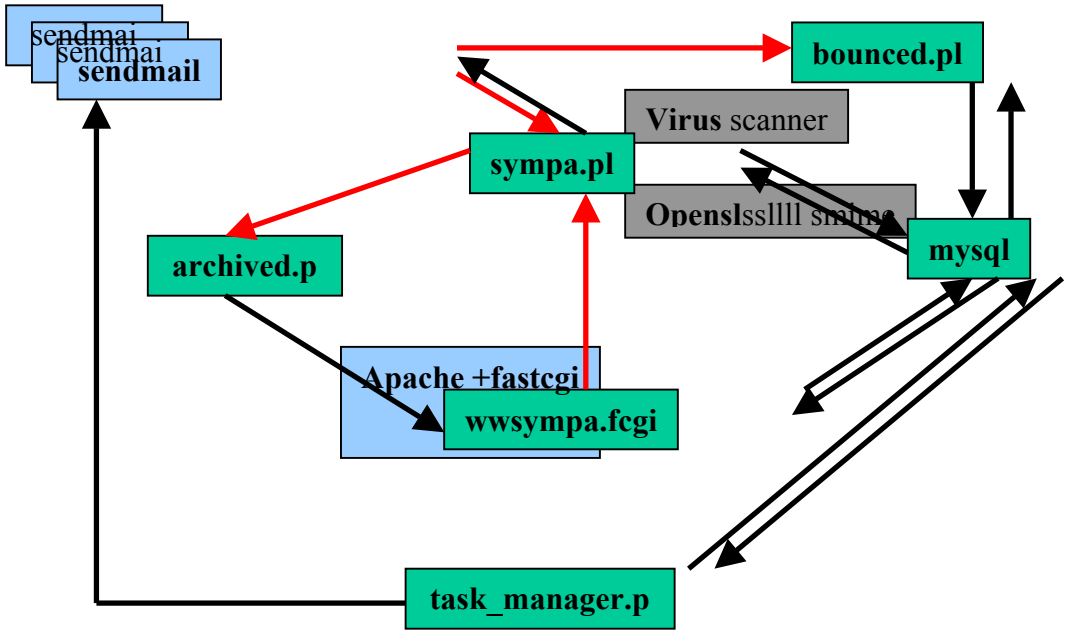
The privileges management system called *scenario* allows to define the operation according to all the elements of the context (user categories, presence of enclosures in the message, use from a computer inside a local network and so on). Moreover, Sympa integrates an authentication system by password; which can be common to other softwares in particular by using LDAP authentication.

Sympa authorizes two definition modes of the subscriber lists population : classical lists built by subscription/withdrawal and dynamic lists directly extracted from an LDAP directory or from database servers.

In the field of X509 certificates, Sympa can recognize S/MIME signature but also encrypt messages to subscriber. HTTPS authentication is also possible on the web interface.
Note that all the broadcasted messages are treated by antivirus.

2 Overview of Sympa organization

A quick watch on the whole organization of Sympa is needed to understand how it works and to see the panel of possibilities.



- *sympa.pl* : it's the main daemon, it receives and diffuses messages via sendmail (postfix, qmail or exim). It consults and modifies the subscriber lists in the database. It feeds *archived.pl* after the diffusion of a message. It calls an external antivirus for all the messages to be relayed towards a moderator or distributed ;
- *wwsympa.fcgi* receives requests from HTTP server (Apache or Roxen). Thanks to the module fastcgi, this CGI is resident : it preserves all the initializations done in memory. In particular, it maintains its connection with the SQL server as well as a memory image of list objects. *wwsympa.fcgi* makese updates in the database. It delegates the messages diffusion to Sympa via a spool ;
- *bounced.pl* is the daemon in charge of processing the non-delivery reports, received from a SMTP engine and then it updates the database ;
- *archived*.pl is the daemon in charge of archiving. It treats messages of the « outgoing » spool and converts them into HTML template used by wwsympa.fcgi ;
- *task_manager.pl* : currently this daemon is only used to transmit subscriptions recall messages and to remove the non-significant bounces. Many tasks will be reserved to it in new version.

3 Data organization

The data model of Sympa is composed of :
- The Sympa database ;
- A set of files ;
- Databases (LDAP or SQL) for constituting dynamic lists, according to the configuration of external data sources ;
- LDAP directories for authentication.

Do not confuse the Sympa database (in general MySQL) and the external databases of Sympa used for includes. In the second case, Sympa only read these external data.

3.1 Database structure

Although, for « historical » reasons, the setting up of a RDBMS for the management of subscriber tables is still mentioned as optional, it is imperative to set up a database if the web interface is installed. Indeed, only the database ensures the management of writing conflicts between the fastcgi processes and the various daemons. Therefore, the RDBMS is not only a way to improve performances of the software. Several RDBMS are supported : MySQL, Postgresql, Sybase and Oracle. The access mode to the base is configured in the file `sympa.conf`. The base consists of two tables :

1. the table user_table which enables to store information about users (e-mail, password and preferred language) ;
2. the table subscriber_table : information about subscriptions.

3.2 File structure

Refer to section « organization » of the reference handbook of Sympa : http://listes.cru.fr/sympa/doc/sympa/node3.html#SECTION00310000000000000000 and to the examples of this document. We can distinguishe 3 kind of files :

- `sympa.conf` and `wwsympa.conf` : the global configuration of the software ;

- Configuration files of the virtual robot : robot.conf, edit_list.conf, templates, scenario…
- Files relating to a list : config, stats, archives, bounces,
- Spools.

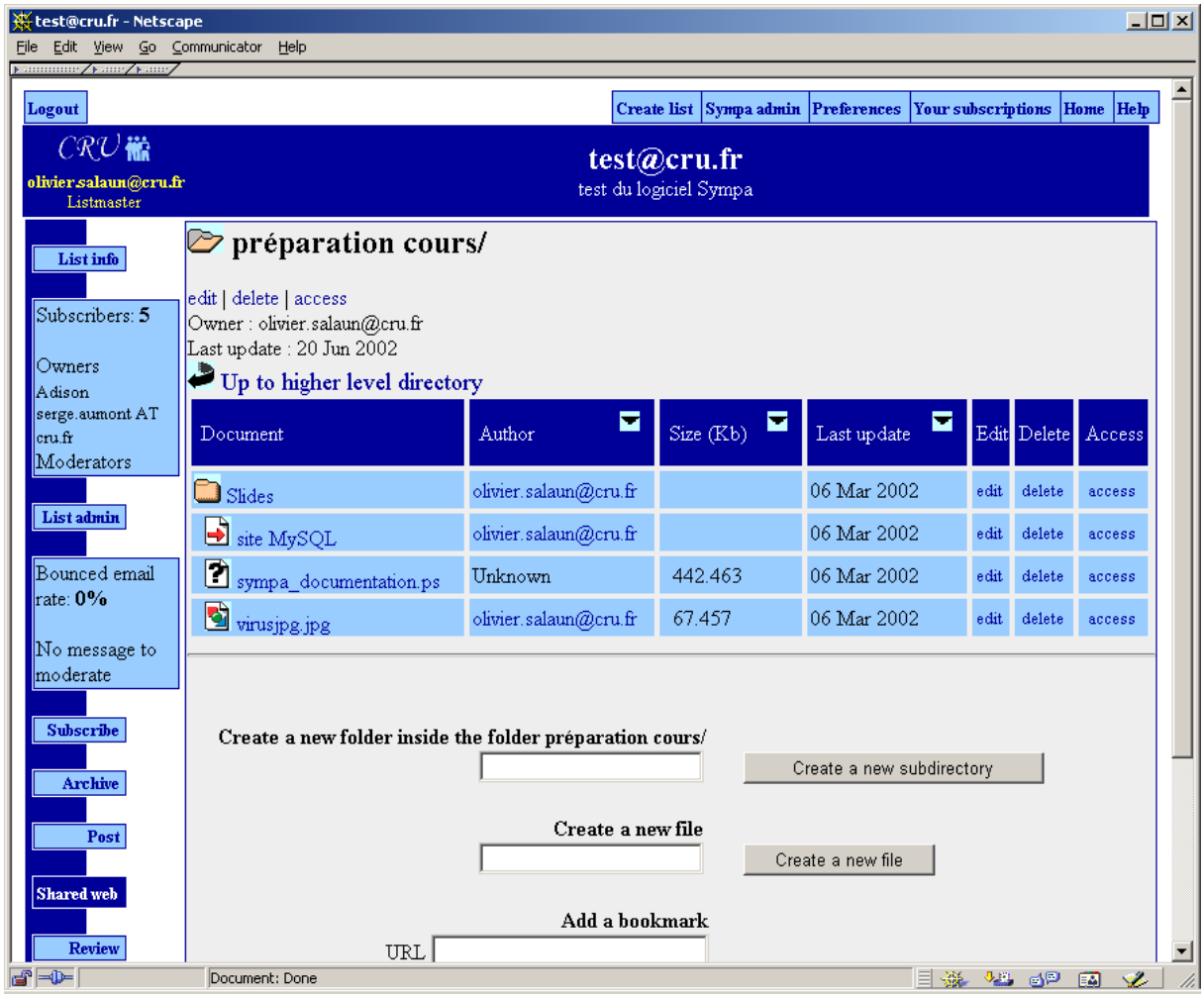
4 The web repository

A web repository for files is available for each list ; it is accessible from the web page of the list. The ambitions of this documentary space are modest, the idea wasn't to propose a management tool of web site (see Rearsite) but only a space to trade documents and bookmarks among subscribers of the list.

4.1 Functionalities

The main functionality is to upload files. It is also possible to edit, rename, and remove uploaded files. A description can be associated to each file. The web repository also allows folders and bookmarks (URLs) management.

By default, the web repository of a list is idle ; activation is made by the list owner, via the web administration interface.



It is possible to deposit a file `index.html` in a folder, it will replace the index file while reading the folder.

Reading access to documents in the web repository is completely controlled by WWSympa which uses its authentication system (internal or LDAP) to identify the client.

4.2 Definition and heritage of privileges

The web repository allows to define 3 types of privileges for each element :

- Reading : access right to a document, a folder ;
- Edition : right to edit and rename/remove a file. Also defines the right to deposit in a folder ;
- Control : allows to define rights in reading/editing of an element of the web repository. This privilege also allows to change the owner of a file/folder.

Each new entry (file / folder / bookmark) inherits the privileges of the folder to which it belongs. The privileges associated with the root folder are defined by the list parameter `shared_doc` (scenarized), except for the control privilege, assimilated as the right to edit the parameter `shared_doc` (by default restricted to the list owner).

Extract of list configuration :

```
shared_doc
    d_read public
    d_edit private
```

The privileges on a file depend of course on its attributes, but also on those of its ascent ; it allows to keep control on a hierarchy of the deposit space. For instance, for the file whose absolute path is D1/D2/F1, the privileges follow the next diagram :

- reading and editing : priv(F1) = priv(D1) AND priv(D2) AND priv(F1) ;
- control : priv(F1) = priv(D1) OR priv(D2) OR priv(F1)

5 Subscription options

The configuration of each list allows to define the available options for messages reception (parameter *available_user_options*) and the default subscription option (parameter *default_user_options*). Each subscriber can thus choose a subscription option. Let's quote the options :

- **`nomail`** : no useful reception in order to suspend the messages reception while keeping the privileges of a subscriber ;
- **`digest`** : grouped and periodic reception of messages (Sympa uses the MIME multipart/digest format, the periodicity can be changed) ;
- **`summary`** : idem digest but reception limited to the messages list ;
- **`urlize`** : messages are delivered normally but attachments are replaced by links ;
- **`text`** and **`html`** : allows to specify the preferred version, when the distributed message contains the two forms (multipart/alternative).

6 Antivirus

Sympa is designed to use the services of antivirus to analyze incoming messages ; the following antiviruses are recognized :

- uvscan (McAfee)
- fsav (Fsecure)
- sophos
- AVP
- viruswall (Trend Micro)

You need to set `antivirus_path` and `antivirus_args` parameters in `sympa.conf` to activate the viruses detection (in the incoming messages) in `sympa.pl`. Contaminated messages are put apart (`bad/` subdirectory) and the author of the message is warned (see the template `your_infected_msg.tpl`).

Viruses found in one week at the CRU (uvscan use) :

```
W32/Magistr.b@MM              51
W32/SirCam@MM          37
W32/Hybris.gen@MM      15
W32/Klez.e@MM          8
VBS/Tam@M              6
W32/BadTrans@MM        4
W32/Magistr.b.dam1            4
W32/Gibe@MM            3
W32/Nimda.gen@MM              1

TOTAL                         129
```

The Sympa server of the CRU intercepts more than 100 contaminated messages a week.

7 Processing of bounces (non-delivery reports)

Casually, the non-delivery reports for distributed messages are transmitted to list owners. Their mailbox is then flooded with these non human-readable reports. To avoid this problem, Sympa treats bounces automatically, leaving (by now) the final operation of withdrawal to list owners.

7.1 Analysis of bounces, bounced.pl

Bounces are received at the address `mylist-owner` then stored in a spool by the program `bouncequeue`.

Example of alias :

```
mylist-owner: "|/home/sympa/bin/bouncequeue mylist"
```

The `bounced.pl` daemon treats the non-delivery reports in the spool in order to find out the email address of the concerned subscriber and the error type. The `bounce_subscriber` field (table `subscriber_table`) is updated ; the last non-delivery report of the subscriber is archived (archive folder defined by the `bounce_path` parameter in `wwsympa.conf`).

The bounce_subscriber field (subscriber_table table) :

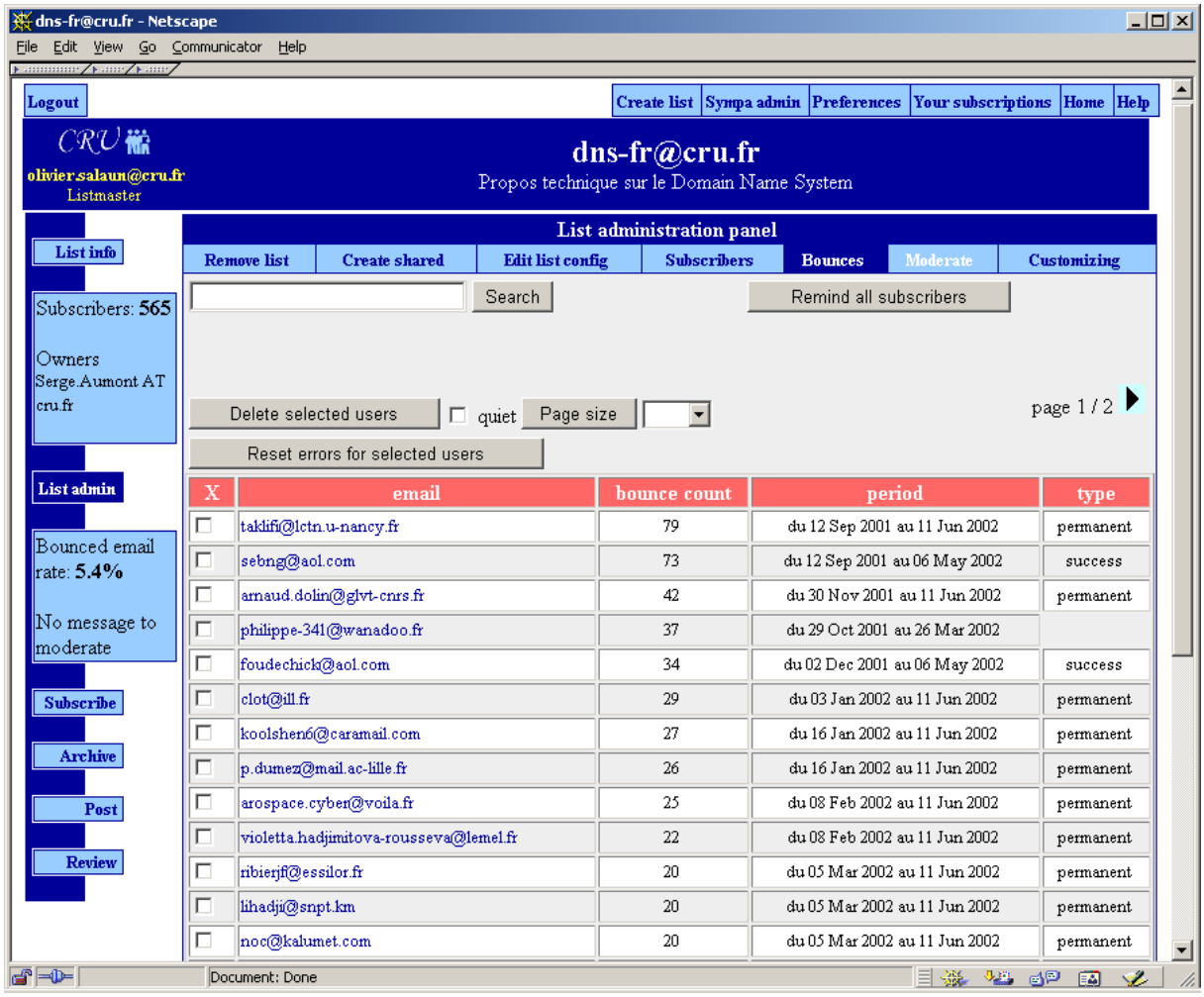
```
1014647647 1016081883 97 5.2.2

   1014647647 : date of the first error (25 February 2002)
   1016081883 : date of the last error (14 March 2002)
   97 : number of bounces received
   5.2.2 : error type (defined by the RFC 1891)
               5 => permanent error
               2.2 => mailbox full
```

If the rate of "bouncing" subscribers in a list exceeds a certain level (defined by the parameter bounce_warn_rate), the owner will receive an alarm for each message sent in the list. This message is currently defined in the NLS (8,28).

7.2 The bounces management interface

The web interface in charge of the administration of a list allows to manage "bouncing" subscribers. The list of "bouncing" email addresses allows to monitor the frequency of errors and their type. The list owner can then choose to cancel errors or unsubscribe the address. The subscriber will then receive a withdrawal message defined in the `bye.tpl` template (except if the box « without preventing » is checked).



Web bounces management interface

7.3 Expiration of bounces

When an email address no longer generates an error, the received errors must be forgotten by the system. This task is entrusted to the Sympa task_manager, via the `expire_bounce` parameter of sympa.conf. The default scenario (and the only one currently) expires bounces of subscribers who no longer generate bounces(ten days after the last diffusion in the list).

The task expire_bounce.daily :

```
title.fr expiration de bounces 10 jours antérieurs au dernier mail diffusé
title.us expire of bounces older than 10 days before message distribution

/ACTION
expire_bounce (10)
next ([execution_date] + 1d, ACTION)
```

7.4 Projects of development

Since release 3.3.3, Sympa carries out a correlation between bounces and distribution of messages (via the `msg_count` file). This piece of information should enable, in future releases, to automate a certain amount of withdrawal. However, in some cases the decision will remain to the list owner.

8 The task manager

Sympa needs to do planned operations ; it uses the task manager for some of these operations. The task manager is to Sympa what *crontab* is to unix. The `task_manager.pl` daemon plans tasks on the basis of some list parameters (expire_task, remind_task) and from global parameters (crl_update, expire_bounce). Tasks can be complex, the daemon stores each task in a spool.

Eventually, some repetitive operations currently treated by sympa.pl will be done by the task manager (digests sending, cleaning up in spools …).

Example of planned task (expire.yearly.task) :

```
title.fr procedure d'expiration : envoi de 2 messages d'avertissement avant
suppression
title.us expiration routine : sending of 2 warning mails before deletion
title.hu törlés menete: a végleges törlés elõtt 2 figyelmeztetõ levelet küld ki

/STEP1
@selection = select_subs (older ([creation_date]-1y))
send_msg (@selection, expire_warning1)
next ([execution_date]+3w, STEP2)

/STEP2
@selection = select_subs (older ([creation_date]-1y))
send_msg (@selection, expire_warning2)
next ([execution_date]+1w, STEP3)

/STEP3
@selection = select_subs (older ([creation_date]-1y))
@deleted = delete (@selection)
```

```
send msg (@deleted, expire deletion)
stop ()
```

The tasks models are customizable (like scenarios) in folders `list_task_models` and `global_task_models`.

9 Virtual robots

The idea of virtual robot is to Sympa what «virtual hosts» are to Apache: the way to manage several mailing-lists services with a single Sympa server and an easier administration. In user view (subscriber, list owners, moderators), services turn out to be completely independent. Example :

https://listes.cru.fr/wws ,
https://listes.renater.fr/wws ,
http://listes-dgcid.diplomatie.gouv.fr/wws, …

9.1 Internal organization of virtual robots

To each virtual robot tallies a robot address `sympa@domaine`. The robot lists are managed in the virtual robot domain. The mailing lists are visible only via the robot of the same domain.
To succeed in setting up a virtual robot, it must be understood how Sympa and WWSympa choose the virtual robot context to apply to each message or HTTP request.

1. sympa.pl treats messages which were put in the message spool via the **`queue`** program and alias of the electronic mail engine. Example :

   ```
   cours-sympa-20-03-2002@cru.fr: "| /home/sympa/bin/queue sympa-fr@cru.fr "
   ```

   sympa.pl retrieves the argument from the `queue` program to identify the list and the corresponding robot (a header field `x-sympa-to: cours-sympa-20-03-2002@cru.fr` is added to the message).

2. `wwsympa.fcgi` : each HTTP request is qualified by the SERVER_NAME environment variable (the host called by the client thus the one used for the definition of the Apache virtual host). An internal table in Sympa, built from these robot configuration files, allows to match this host name and the virtual robot domain. For example `listes.cru.fr` serves the virtual robot `cru.fr`.

9.2 Creating a virtual robot

9.2.1 DNS

You must define a **`MX record`** in the DNS for the virtual domain and a **`CNAME`** or a host (*A record*) for the associated web server.

In the case of a HTTP**S** server, you will have to define a new IP address and not a CNAME (constraint due to the stacked architecture of the HTTP protocol under the SSL layer).

9.2.2 Configure your SMTP engine

Your MTA must recognize the new domain.

- With sendmail and the macro m4 see http://www.sendmail.org/virtual-hosting.html
- With Postfix see http://www.postfix.org/virtual.5.html
- With sendmail and the French "kit Jussieu" see http://www-crc.u-strasbg.fr/docs/kit-jussieu/support/node115.html

9.2.3 Creating aliases

```
listmaster@virtual.fr: "|~sympa/bin/queue listmaster@virtual.fr"
sympa@virtual.fr: "|~sympa/bin/queue sympa@virtual.fr"
sympa-request@virtual.fr: "|~sympa/bin/queue listmaster@virtual.fr"
bounce+*@virtual.fr : "| ~sympa/bin/bouncequeue sympa "
```

9.2.4 Apache

Configure a virtual host on Apache. It's not necessary to define one/many FastCgi server(s) for each virtual robot since each FastCgi server is able to serve any virtual robot. Therefore, we find the FastCgi servers definition in the common part of the Apache configuration and the software path in each virtual host :

```
AddHandler .fcgi
FastCgiServer /bin/sympa/bin/wwsympa.fcgi -processes 3 -idle-timeout 120
<VirtualHost 195.220.94.165:80>
ServerName listes.virtual.fr
<Location /wws>
     SetHandler fastcgi-script
</Location>
ScriptAlias /wws /home/sympa/bin/wwsympa.fcgi
</VirtualHost>
```

The remaining will be treated sooner or later via a creation form of virtual robot. While waiting these refinements, you should create « manually » each environment. In the following examples, the virtual robot serves the domain *virtual.fr*.

9.2.5 robot.conf

The virtual.fr robot is defined by the robot.conf file located in the `~sympa/etc/virtual.fr/` folder. This configuration file allows to changer parameters of sympa.conf and wwsympa.conf (for example the page colors). But `robot.conf` must define the parameter `http_host` which will be used to match the name of the HTTP server referenced and the corresponding virtual robot.

Description of robot parameters :

- `email` : the local part of the robot address. Example : `sympa` ;
- `title` : the welcome page title;

- `default_home` : the default welcome page (home : classification by *topics* or *lists* : alphabetical order) ;
- `create_list` : the name of the scenario controlling the access to the routine which creates lists ;
- `lang` : the default language of the robot ;
- `log_smtp` : activation of the log calls to sendmail ;
- `log_level` : level of log ;
- `listmaster` : e-mail of list masters separated by « , » ;
- `max_size` : maximum size of messages (redefinition allowed for each list) ;
- `dark_color, light_color, text_color, bg_color, error_color, selected_color, shaded_color.`

Example :

```
# http_host must be the same as the server_name defined in Apache Virtual Host
http_host  demo.sympa.org

wwsympa_url https://demo.sympa.org/wws

title A virtual robot dedicated to Sympa demo

default_home  lists

log_level 3

dark_color #00aa00

light_color #ddffdd

selected_color #0099cc

bg_color #dddddd
```

9.2.6 The working directory

Create a folder which will contain the virtual robot lists : `~sympa/expl/virtual.fr`. Each list created in this folder is assigned to this robot.

9.2.7 Restart Apache, Sendmail and Sympa's daemon

The new virtual robots detection is not dynamical for these three software.

9.2.8 Other configuration

As each list can have a specific set of scenarios and templates, these configuration elements can be defined in a virtual robot.

When evaluating a scenario, like the **intranet** scenario that controls the **send** operation in the **foo** list of the **virtual.fr** robot, Sympa searches the **send.intranet** file in the following folders :

- ~sympa/expl/virtual.fr/foo/scenarii/ (only for the operations relating to a list) ;
- ~sympa/etc/virtual.fr/scenarii/ ;
- ~sympa/etc/scenarii/ ;
- ~sympa/bin/etc/scenarii/.

The same strategy is applied for the templates (models of messages and HTML pages) :

- ~sympa/expl/virtual.fr/foo/ (warning : no subfolder templates) ;
- ~sympa/etc/virtual.fr/template/ ;
- ~sympa/etc/template/ ;
- ~sympa/bin/etc/template/.

This organization allows to completely personalize by list, by robot or for the web site appearance both rights for each page of the web interface and the mailing robot.

10 Writing your own scenarios

The scenario system is a language enabling to specify the rights associated to each operation. Almost all the operations in Sympa are under the control of scenarios : lists creation, messages broadcasting, access to a list deposit space and so on. The visibility of a list is controlled by a scenario which allows to hide lists from other categories of users. Therefore, it is possible to create specific views for many categories of users in order to create intranets.

See : http://listes.cru.fr/sympa/doc/sympa/node11.html#SECTION00118000000000000000

Sympa is distributed with a default set of scenarios which are installed in ~sympa/bin/etc/scenari,
http://listes.cru.fr/sympa/distribution/current/src/etc/scenari/

> Never modify these files, next update of Sympa would crush your work. If you wish to modify the standard scenarios, copy and edit them in the ~sympa/etc/scenarii folder.

10.1 Principle of scenario operations

A scenario is a list of rules sequentially called. Each rule is made of :

- a condition relating to the calling context of the scenario ;
- the authentication method used (Header field From:, password or authentication certificate) ;
- an action which will be called if the rule apply to the calling context.

Example :

```
title.fr limité aux abonnés authentifiés
title.cz pouze èlenové
title.hu listatagok
```

```
is_subscriber([listname],[sender])              smtp         -> request_auth
is_subscriber([listname],[sender])              smime,md5    -> do_it
```

This example, applied to the control of messages distribution, enables to limit the sending of messages to identified subscribers. We can notice preset conditions like `is_subscriber` and variables ([`listname`] and [`sender`]) which are instantiated while calling, depending on the current context. Three authentication methods are recognized by the scenarios :

- smtp : indicates that we trust the `From:` header field of a message ;
- md5 : password authentication (passwords given by Sympa are built by the MD5 hash algorithm) ;
- smime : strong authentication based on an X509 certificate (S/MIME signature or HTTPS session with client certificate).

Actions in right hand of scenarios rules are rudimentary indicators : request_auth (return message with a request of authentication), do_it (do the service).

If you write scenarios, think about adding a title (`title.<lang>`), this one will appear in the pop-down menus of the list configuration form.

Each scenario ends up by an implicit rejecting rule :

```
true()                                  smtp,smime,md5    -> reject
```

10.1.1 Available conditions in a scenario :

- true ()
- equal (<value>, <value>)
- match (<var>, /perl_regexp/)
- is_subscriber (<listname>, <value>)
- is_owner (<listname>, <value>)
- is_editor (<listname>, <value>)
- is_listmaster (<value>)
- older (<date>, <date>)
- newer (<date>, <date>) # true if first date is posterior to the second date

To reverse a condition : prefix it by a **!**

10.1.2 Available variables in a scenario :

- [sender] : the *sender* of the current message, by extension, the caller of the current request ;
- [email] : in commands which accept e-mail in argument, used only in an unsubscription context ;
- [subscriber-><subscriber_key_word>] : subscriber attributes : email | gecos | bounce | reception | visibility | date | update_date |<additional_subscriber_fields> .

It is also possible to add owner fields in the subscriber table and to reach them in scenarios and templates (See http://listes.cru.fr/sympa/doc/sympa/node5.html#db-additional-subscriber-fields).

- [listname]
- [list-><list_key_word>] : all entries of the list configuration file
- [conf-><conf_key_word>] : all elements of Sympa configuration (sympa.conf)
- [msg_header-><smtp_key_word>] : all message header fields (in `send` scenarios only)

To moderate messages containing attachments proceed like this:

```
match([header->Content-Type],/multipart/)     smtp,smime,md5  -> editorkey
```

- [msg_body] : body of the message

To block French messages in a English-speaking list :

```
match([msg_body],/[àâçéèêüùû]/)               smtp,md5,smime -> reject
match([msg_body],/bonjour/)                   smtp,md5,smime -> reject
```

- [msg_part->type]
- [msg_part->body] Any test applied to these variables is a logical **OR** applied to each part of the message.
- [is_bcc] Indicate that a target list is neither in the `To:` header field nor in the `Cc:` header field

An anti-spam trick used at the CRU : a common rule to each scenario :

```
equal([is_bcc],'1')      smtp -> request_auth
```

This rule is defined in the file : `/home/sympa/etc/scenarii/include.send.header`
(files include.<action>.header are included as first rules of all scenarios <action>.*).

- [remote_host] : the `REMOTE_HOST` variable set by the HTTP server. It assumes that Apache has been configured to evaluate this variable, but it enables to create a kind of intranet, in particular by using this variable in the `visibility` scenario, allowing to control "who has the rights to know the being of a given list".

11 Templates

Reports of commands sent by Sympa are described in «templates» (except for some old commands) ; this separation of code and messages makes it easier to translate and customize mesages. Templates are model messages that can contain programming elements (variables, conditions, loops...) which will be evaluated while sending the message.

11.1 Organization

Default templates are provided in the Sympa distribution, translated in 10 languages. They are customizable at different level, in the following order of priorities :

1. the list (~sympa/expl/mylist/)
2. the virtual robot (~sympa/etc/virtual.fr/etc/templates/)
3. the site (~sympa/etc/templates/)
4. the distribution (~sympa/bin/etc/templates/)

Warning : if you customize templates, do it in ~sympa/etc/templates/ (the folder ~sympa/bin/etc/templates is overwriten at each new installation).

Files have the `.tpl` extension. If the template relates to a particular language, it will have the `.<lang>.tpl` extension.

Templates can include SMTP header fields (useful to define multipart messages), in which case Sympa does not add SMTP header fields to the message. A template is regarded as MIME if it starts with a From: (see further example).

11.2 Format

Syntactic elements of templates are defined between brackets '[]' ; their interpretation is not case sensitive.

11.2.1 Variables

Examples :

```
[url]
[is_owner]
[list->name]
[user->lang]
```

Operator `->` allows to refer an entry of a `HASH`.

List of usable variables for each template is provided in the documentation : see template site and template list).

11.2.2 Conditions

A control structure [IF...][ELSE][ENDIF] allows to adapt the page according to the context linked to its use :

Example:

```
[IF  user->lang=fr]
Bienvenue dans la liste [list->name]
[ELSIF user->lang=es]
Bienvenida en la lista [list->name]
[ELSE]
Welcome in list [list->name]
[ENDIF]
```

11.2.3 Loops

Loops allow to look over a list of elements (represented internally by an array or an associative array).

Example :

```
Les listes publiques :

[FOREACH l IN lists]
   [l->NAME]
   [l->subject]
[END]
```

`[elt->NAME]` is a particular variable containing the current key in the associative array (in the example above, the name list). In the case of an array loop, the `[elt->INDEX]` variable contains the current index.

11.2.4 Enclosing of files

Enclosed files can be « passive » (INCLUDE) or the included file can be interpreted as a template (PARSE). The path of the file to be included will be provided directly or via a variable.

Example of textual files included :

```
[INCLUDE 'archives/last_message']

[INCLUDE file_path]
```

Example of template inclusions (themselves parsed) :

```
[PARSE 'welcome.tpl']

[PARSE file_path]
```

11.2.5 Escapement

It is possible to interrupt the interpretation of a template (to use [] for instance). Directives `[STOPPARSE]` and `[STARTPARSE]` allow (respectively) to stop and resume the interpretation of the template.

Example of an escapement sequence in a sensitive JavaScript function :

```
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!-- for other browsers
  function toggle_selection(myfield) {
    for (i = 0; i < myfield.length; i++) {
    [STOPPARSE]
       if (myfield[i].checked) {
            myfield[i].checked = false;
       }else {
            myfield[i].checked = true;
       }
    [STARTPARSE]
    }
  }
// end browsers -->
</SCRIPT>
</HEAD>
```

11.3 List of templates

In a general manner, a template is a model of command report. In a near future, all commands reports will be defined in templates (currently some use NLS).

Template	Associated operation	Description of the message
bye.tpl	SIGNOFF	Withdrawal message
global_remind.tpl	REMIND *	Recall of the whole subscriptions
helpfile.tpl	HELP	Help on Sympa
info_report.tpl	INFO	Information on a list
invite.tpl	INVITE	Invitation message
list_created.tp		Notification list creation
list_unknown.tpl		Non-delivery Report (unknown list)
listmaster_notification.tpl		Notifications for listmaster(s)
lists.tpl	LISTS	Catalog of lists
moderate.tpl		Notification for a message to be moderated
modindex.tpl	MODINDEX	List of messages on moderation standby
reject.tpl	REJECT	Notification for message rejection
remind.tpl	REMIND	Subscription remind
removed.tpl	DEL	Removal message
review.tpl	REVIEW	Subscribers list
sendpasswd.tpl		Password remind
stats_report.tpl	STATS	Statistics on the list
summary.tpl		Message in “summary” mode
welcome.tpl	SUBSCRIBE / ADD	Welcome message
x509-user-cert-missing.tpl		Subscriber notification : no certificate for encryption
your_infected_msg.tpl		Subscriber notification : reject of infected message

11.4 Examples

Welcome message (welcome.fr.tpl) including the last diffused message :

```
From: abc-request@cru.fr
Subject: Welcome in list ABC
Content-type: multipart/mixed; boundary="myboundary"

--myboundary
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding:  8bit

Welcome in the list ....

You will find bellow the latest distributed :

--myboundary
Content-Type: message/rfc822

[INCLUDE '/home/sympa/expl/abc/archives/last_message']

--myboundary--
```

12 LDAP support in SYMPA

In its release 3.3.3, Sympa proposes an LDAP support for the three following functionalities : web authentication, dynamical lists setup and LDAP filters in scenarios.
This course's section is a brief and practical introduction to LDAP, and then a tutorial for setting up LDAP in Sympa for the three cases quoted above.
The whole set of configuration possibilities are not described ; for that, refer to Sympa handbook : http://listes.cru.fr/sympa/doc/sympa/ .

12.1 Short introduction to LDAP

This paragraph describes in a very coarse and practical manner LDAP technology.
LDAP is a 'standard' protocol allowing an easy and fast access to information of a directory in a broad sense.
LDAP servers are optimized to quickly deliver data in reading ; writing data via the LDAP protocol in general takes up a lot of time. Therefore, information proposed by an LDAP server is relatively stable, such as information on employees of a company, its servers, its communication object proposed by these servers …
LDAP information is in general structured in a hierarchical way.

12.1.1 Structure of an LDAP base

The structure of an LDAP server is called **DIT** (Directory Information Tree).
An LDAP server has no knowledge of the type of the objects that it publishes ; the different objects are defined by a diagram specific to an LDAP server.
The root of an LDAP server is called **`root DN`**. An example of root DN could be : *DC=MyInstitution, DC=us*. This example assumes a match between the naming of the root of LDAP and DNS ; it's in no way obligatory (but desirable for unicity reasons ?).
An LDAP **`object`** is identified by its Distinguish Name (DN), which is unique in an LDAP base, and describes its address in the tree.
Thus, the DN of an LDAP object located in the 'container' *OU=People* of the preceding root might write as follows :

```
DN : uid=tartempion,OU=People,DC=MonEtabliss,DC=fr
```

The leftmost part of the DN (here, *uid=tartempion*) is called the **RDN** : Relative Distinguish Name. It also appears as an attribute of the object.

12.1.2 Composition of an LDAP object

An LDAP entry is addressed by its DN. It is made up of various attributes, which have no significance for the server (except *objectClass* and *userpassword* attributes).
These attributes can have multiple values.

- The **objectClass** attribute establishes the type of the object handled ; in fact, the diagram of an LDAP base describes the set of objectClass supported by this base, and for each objectClass, possible and required attributes ;
- The **userPassword** attribute: it allows to assign a password to an LDAP entry ; this password can either be in clear or encrypted form. If the password is encrypted, the type of encryption is defined by putting the encryption type in front of the password, enclosed by the characters '{' and '}' ; for instance, *{sha}xxxxxxx* for a sha type password.

An LDAP object necessarily has a multiple value *objectCLass* attributed to it. The RDN is part of attributes of an LDAP entry. The unicity of the RDN is not ensured in an LDAP base. The syntax and the use of each attribute are described in the diagram : numerical, 'case sensitive', single or multiple valued, obligatory or not…

12.1.3 LDAP groups

Two 'standard' classes of LDAP define objects of type group : groupOfUniqueNames and GroupOfNames.
These objects have various standardized attributes, and a multiple value one which contains the DN of LDAP objects belonging to the group ; this standardized attribute is named respectively *uniqueMember* or *Member*, the first being more common.
Many LDAP authentication interfaces support these groups (and particularly the first, *groupOfUniqueNames*).

12.1.4 Level of protocol

The 'standardized' definition of LDAP protocol is currently in version 3 - not to be confused with the various versions of LDAP servers (Iplanet : 5.xx, OpenLDAP : 2.xx, …).

12.1.5 Accentuated characters

Since version 3 of the LDAP protocol, accentuated characters are coded in **UTF8**.

12.1.6 Exchange format

An exchange format of LDAP objects has been standardized : format **LDIF** (???). Non pure ASCII characters are encoded using the 'encode64' format.

12.1.7 Access rights

Access rights are defined (or not) on the LDAP server level in order to give access in reading, writing and creating …, according to the server authentication.

12.1.8 Unfolding of LDAP request

An LDAP request usually unfolds the following way :

- Establishing a TCP/IP connection to the LDAP server (IP address and TCP port) ;
- Establishing of a logical connection (bind) to the LDAP server ; this logical connection is made up of a DN and the password of the client.
  An anonymous connection is also available with empty DN and password ;
- Generating of one or several requests, carrying the following elements :
  . The DN of a tree search, or the root of the server ;
  . the range of the search : **sub** (like sub-tree) for an object located on any level of the search directory; **one** for an object located strictly on the same level of the search directory ; **base** for the object himself ;
  . A search filter.
- Possibly, entering the list of attributes wanted, and other options ;
- Breaking off the logical connection (unbind) ;
- Breaking off the TCP connection.

Since version 3 of LDAP, it is possible to generate an LDAP request without a preliminary bind ; in this case, an anonymous connection is assumed.

12.1.9 LDAP search filters

The simplest LDAP filter takes the following shape « (*AttributeName=AttributeValue*) ». Thus, to interrogate all the entries which have the attributes '*cn*' valuated as '*toto*', the LDAP filter will be *(cn=toto)*. The value of the attribute (AttributeValue) can have the character '*' ; it replace any occurrence of an unspecified character.
It is possible (and even frequent), to generate a more elaborate filter, using logical operators : AND (&), OR (|), NOT (!).
For instance :
(&(objectClass=inetOrgPerson)(cn=math*)(|(serv=cri)(serv=pers)))
gives all the entries of *inetOrgPerson* type, having the attribute *cn* starting by *math* belonging to the *cri* or *pers* service.

12.1.10 Replicates

The introduction of a reliable and a powerful LDAP service very often requires the setting up of replicates of the LDAP base. Replicate technique is not described in the LDAP protocol ; it is therefore specific to each implementation.
Very often, the replicate is of master/slave type.
In case of replicate use, it is necessary to setup clients in order to enable them to make requests to a backup server in the case of an unavailability of the first server.

12.1.11 Usual configuration of an LDAP client

LDAP clients are setup in the same way. As we have previously seen, they require the following parameters :

- HOST : the name of the computer supporting the LDAP server ;
- PORT : the TCP port of the LDAP server ;
- SUFFIXE : base of the search ;
- SCOPE : range of the search ;
- FILTRE : filter of the search ;
- If the request requires an non anonymous bind, the DN and the password of the bind will be indicated in the parameter settings.

Other parameters can be useful according to the software.
In case of replicate use, the syntax is very often as follows :

HOST ldap1.univ.fr:392,ldap2.univ.fr:389

Specify desired attributes
When we don't need all the object attributes, it is advisable to specify the list of desired attributes ; this can considerably limit the amount of information given back by the requests made.

Specify the objectClass in requests made
The objectClass allows to select the type of object we wish to reach. It often happends that the same attribute is present in different type objects, such as attributes *CN* (Common Name), *description*, …

LDAP authentication
If the LDAP server administrator has correctly configured the LDAP server, it is impossible for clients to read the user password attribute. Authentication technique based on LDAP is as follows :

- An anonymous connection, in order to retrieve the DN of a person to identify, according to the identifier given by the user :
  Example of request : *(&(objectCLass=inetOrgPerson)(uid=IdentifiantUser))*

- Bind with the DN user and the password entered. If the bind is accepted, the password is correct.

12.1.12 Example of LDAP objects

See appendix 1 for an example of the type of person and an example of a group of people.

12.2 Prerequisite for LDAP support in Sympa

The `perl-ldap` module must be installed on a computer supporting the software Sympa so as to use the LDAP functionalities of Sympa.

12.3 Use of LDAP for authentication in wwsympa

12.3.1 Authentication in wwsympa

The wwsympa interface is geometricaly variable according to the type of user 'connected'.
The types of user are as follows :

- anonymous : user not logged in in the interface ;
- subscriber to one or more lists ;
- moderator and/or owner of one or more lists ;
- listmaster.

The last three cases require of course authentication in the interface wwsympa.
In an internal way, a user identifies himself with his e-mail and a password known by Sympa (stored in Sympa's data base, in an encrypted way, but with the possibility of being decoded). The initial password of the user is provided by Sympa ; the user can change it later on via wwsympa. He can also ask Sympa to remind his password.

12.3.2 Authentication with LDAP

To use LDAP authentication, first you need to configure Sympa ; this can be done with the help of the auth.conf file, located in ~sympa/etc/ folder.
When this file is present, and well configured, authentication in wwsympa can be done in this way :

- The user enters his e-mail or his LDAP login, and his password ;
- If the user name matches a valid e-mail, Sympa tries to authenticate via its internal mechanism described before ;
- If authentication fails, or the user name does not match a valid e-mail, wwsympa will try authentification with the LDAP system.

12.3.2.1 auth.conf file

Consult the reference handbook at the following address :
http://listes.cru.fr/sympa/doc/sympa/node8.html#SECTION00811000000000000000
This file contains one or more paragraphs starting by the ***LDAP*** keyword.
Each paragraph is made up of a set of **key words** <-> value couples.
If you wish to make a comment, start the line by the character #. Blank lines will be ignored.

Here is an example of an LDAP paragraph :

```
ldap
host ldap.univ.fr:389
suffix ou=People,dc=univ,dc=fr
scope one
timeout 10
get_dn_by_uid_filter (&(objectclass=inetOrgPerson)(uid=[sender]))
get_dn_by_email_filter (|(mail=[sender])(maildrop=[sender]))
email_attribute mail
alternative_ email_attribute maildrop
```

The keywords *host*, *suffix* and *scope* match the basic parameter of an LDAP request as shown before.
The *host* directive enables the support of replicate mechanism ; therefore, we can specify several LDAP servers ; Sympa will try to bind to the first one, then the following one if the first is unavailable, and so on …
Syntax is as follows :

```
host ldap.univ.fr:389,ldap2.univ-fr:392,ldap1.univ.fr:389
```

As indicated previously, the user can login in the wwsympa interface either with an LDAP login, or by entering his e-mail.
In the first case, the filter used for the search will be the one specified by the directive *get_dn_by_uid_filter* and, in the second one, by the directive *get_dn_by_email_filter*.
Note that wwsympa will replace all occurrences of "[sender]" with the login entered by the user in the wwsympa interface.
Thanks to one filter or the other, the LDAP entry matching the user is loaded. wwsympa then controls the password thanks to an LDAP bind with the DN of this entry, and the entered password.
If this bind is accepted, the user is authenticated. It is now up to wwsympa to establish the link with this 'LDAP' user and a Sympa user, known by his e-mail.
It is the role of email_attribute and alternative_email_attribute directives.
email_attribute indicates the name of the LDAP attribute that contains the canonical e-mail of the person.
alternative_email_attribute is an optional directive, which contains the name of a list of attributes that are being used in the LDAP directory for electronic mail aliases.
In the above example, we can notice that the mail attribute contains the canonical e-mail, and that the *maildrop* attribute contains possibly an alternative e-mail address (alias).

12.3.2.2 Authentication with several LDAP servers

There can be several *LDAP* paragraphs in the file auth.conf. In this case, wwsympa tries LDAP authentication firstly towards the server described in the first LDAP paragraph, then towards the second if the first attempt fails, and so on.
It would be possible, for example, for a university which lays out one LDAP server for staff and one for students, to enable an authentication towards these 2 different servers.

12.3.3 wwsympa behavior with an LDAP authentication

wwsympa keeps a record of the type of authentication (using a HTTP cookie). Some menus proposed by wwsympa are modified in case of an LDAP authentication.
In particular, in the user preferences, changing the password is no longer available.
Moreover, if the user has alternative e-mails in the LDAP directory, Sympa proposes to store them in its internal tables.

Intranet use
At the University of Nancy2, authentication to access various data-processing resources (Intranet, electronic mail …) is based on an LDAP directory, and the *uid* attribute which contains the user login. Therefore, it is important that users use the same authentication methods to access to wwsympa.
Sympa is used for internal lists, and also for lists completely or partially external.
We thus wanted wwsympa to always be able to authenticate with its native mechanism for external people, but with LDAP and login for internal users.
This was carried out in a simple manner thanks to the flexibility of Sympa templates : a small part of JavaScript code was added to the template file (loginbanner.tpl) which manages user login.
If this login is an e-mail, and ends by 'univ-nancy2.fr', a warning message appears and asks him to enter his 'LDAP' login.

12.4 Lists issued from LDAP requests

Sympa can manage various types of lists : 'native' lists, which are stored in its data base, and 'include' lists.
This match the *user_data_source* parameter in the list's configuration files, which can be set to `database` or `include`.
The "`include`" lists are made up with the help of requests to SQL or LDAP servers.
There are two different definitions of LDAP lists : *ldap_query* and *ldap_2level_query*.
Consult the reference documentation at the following address :

http://listes.cru.fr/sympa/doc/sympa/node14.html#SECTION001420000000000000000

12.4.1 Lists of type 'ldap_query'

These lists are dynamically made up with the help of a simple LDAP request. They are defined in the configuration file of the list thanks to the `include_ldap_query` paragraph.

Here is an example :

```
include_ldap_query
host ldap.univ.fr
```

```
port 389
suffix ou=People,dc=univ,dc=fr
scope one
timeout 10
filter (&(objectClass=n2pers)(mail=*)(n2PersType=E)(n2PersServ=ufrdroit))
attrs mail
select first
```

We recognize the usual setting of an LDAP request. Here, we wish to build a list of teachers of the law university section having the e-mail attribute valued. The *attrs* directive gives the name of the attribute containing the e-mail address.
If a non anonymous request is necessary, it is possible to specify the DN (*user*) and the password (*passwd*) of the connection. The *select* directive makes it possible to tell Sympa how to act if the attribute containing the e-mail address has multiple values
The generated list will thus be all the people corresponding to the request.

12.4.2 Lists of type 'ldap_2level_query'

These lists are dynamically made up with the help of complex LDAP requests, on two levels. A first request makes it possible to retrieve a list of values ; then, for each retrieved value, a new request will allow it to retrieve the matching e-mail address.
This type of lists enables in particular to treat LDAP groups (*groupOfNames* or *groupOfUniqueNames*).

The paragraph **include_ldap_2level_query** allows us to define such a list. Here is an example, which treats groups of the type *groupOfUniqueNames* :

```
include_ldap_2level_query
host ldap.univ.fr
port 389
suffix1 ou=Groups,dc=univ,dc=fr
scope1 one
filter1 (&(objectClass=groupOfUniqueNames)(|(cn=cri)(cn=ufrmi)))
attrs1 uniquemember
select1 all
suffix2 [attrs1]
scope2 base
filter2 (objetcClass=n2pers)
attrs2 mail
select2 first
```

Here, the first request makes it possible to build a list of DN's people which are in the groups LDAP *cri* or *ufrmi* ; the second one retrieve the matching e-mail attribute.
The *[attrs1]* value can be used in the directives *suffix2* and/or *filter2* ; Sympa will replace it, repetitively by all the values given back by the first request.
When *[attrs1]* is used for the directive *suffix2*, the attribute returned by the first request must be a DN. Therefore it's logical to assign the value *base* to the attribute *scope2*.
Note that it's possible to make complex processing, with the use of directive *regex1* (or *regex2*) which makes it possible to apply a filter of the type 'regular expression' to the result of each request ; *select1* (or *select2*) must then have the value *regex*.
Note also that lists of this type can generate an important load : if the list has n members, to create the list, there will be n + 1 LDAP requests.

Sympa makes it possible to define several paragraphs of the type "include" for a list. It is thus possible to have a list issued from several LDAP and/or SQL requests.

12.4.3 Include list and Sympa's cache

In order to improve the performances of Sympa and wwsympa and to limit the load on SQL or LDAP servers with the include lists, Sympa manages a cache of these dynamical lists.

Therefore, these lists are not rebuilt each time Sympa or wwsympa needs it, but are kept in an internal cache. How frequently this cache is refreshed is settled in the configuration file of the list, thanks to the `ttl` directive.

12.4.4 ttl and cache of lists

Sympa refreshes a list when a reference is made and when his `ttl` has run out. wwsympa doesn't do this work, which may lead to a long response time, if the dynamical lists are numerous and if external sources are slow. Consequently, wwsympa shares a file « .db » with sympa.pl that Sympa updates.
We expect to replace this shared cache via a file by a cache in a database accessible to each of the processes sympa.pl, wwsympa.fcgi, task_manager.pl, bounce.pl making it possible to have the same performances for the lists in include mode as lists in database mode. It would be then possible to manage bounces and put subscription options for the whole lists. Update of the cache would be dedicated to the task manager (task_manager.pl).

12.5 Use of 'LDAP filters' in scenarios

12.5.1 Named Filters (NF)

These filters bring LDAP support at the scenario level. A filter will define an LDAP request.
It would be usable in the 'condition' part of a scenario and enable thus to indicate a list of people authorized to make an act, with the help of this request.

Named Filters (we will use abbreviation **NF**) are ASCII files written in the ~sympa/etc/search_filters/ folder.
These files have necessarily the '*.ldap*' extension.

Here is an example of the contents of the "*EnsLettres.ldap*" NF:

```
host ldap2.univ.fr:392,ldap.univ.fr:389
suffix ou=People,dc=univ,dc=fr
scope sub
filter (&(objetClass=n2pers)(mail=*)(n2Type=E)(n2Campus=lettres)
(|(mail=[sender])(maildrop=[sender])))
```

It is a 'usual' configuration of an LDAP request. Sympa will replace all occurrences of '*[sender]*' by the content of '*[sender]*' for scenarios using this filter, in general, the e-mail address of the user.
In this example, the filter will give some rights to the user having a valued *mail* attribute, from type *E* (Teacher ?) belonging to the Lettres campus and whose *mail* or *maildrop* attribute corresponds to his e-mail address.

12.5.2 Use of Named Filters (NF) in a scenario

The NF are usable in the condition part of a scenario.
The syntax is : `search(NomDuFiltre).`

Here is an example of scenario used for people having the right to send an e-mail in a list ; it is called send.EnsLettresSimple :

```
title.fr Envoi autorise aux abonnes de la liste et aux enseignants du campus
lettres
is_subscriber([listname],[sender])  smtp,md5,smime      do_it
search(EnsLettres.ldap,[sender])    smtp,md5,smime      do_it
```

Here, every subscriber of list '*listname'* and the teacher of 'lettres' campus have the right to send an e-mail to the list '*listname'*.

12.5.3 Scope of these filters

NF associated with Sympa's scenarios offer very advanced possibilities of customization.
Thus, the different roles managed by Sympa (listmaster, owner, moderator, subscriber) could eventually issue from LDAP requests.

Cache of NF
Sympa manages a cache of NF. While setting up, it is necessary to start/restart Sympa in order to test modifications brought to a filter.

12.6 Conclusion on Sympa and LDAP

Sympa natively offers very advanced possibilities of customization. The LDAP support is well integrated at different levels, and offers then these possibilities.
Some improvements are to be desirable, particularly in managing dynamical lists and their refreshment ; this will be done in future versions of Sympa.
Other improvements linked to LDAP seem to be scheduled; take a look at the sources in order to be convinced.

13 MySQL configuration

Sympa requires the services of a database for managing subscribers (one of MySQL, PostgreSQL, Oracle or Sybase). We will introduce the configuration steps with MySQL especially because we use it on our development platform.
For the following, we assume that you a have a MySQL server installed.

13.1 Setting up Perl modules

Sympa accesses its database via DBI, the standard Perl API to access databases. You must set up the DBI module, not part of the standard distribution of Perl. DBI requires a driver (DBD) for each type of database you'll be using.

Perl modules to set up :

- `DBI` >= 1.06
- `DBD::mysql` >= 2.407 (give with Msql-Mysql-modules)

The setup of these modules is taken in charge while installing Sympa. You can also get them :

- Under the shape of tar.gz on the nearest site of CPAN (http://cpan.cict.fr/) ;
- Under the shape of a RPM on RPMFind (http://www.rpmfind.net/linux/RPM/Development_Languages_Perl.html).

If you setup the DBD::mysql module under the shape of a RPM, it will require the setup of the mysql-devel RPM, given the access libraries to MySQL.

13.2 Create the database

The database of Sympa has 2 tables :

- user_table : user preferences ;
- subscriber_table : data related to the subscription.

The script which creates the base (see below) is given with the distribution (src/etc/script/create_db.mysql). Use the `mysql` command line client to execute the script :

```
# /usr/bin/mysql -u root -p < create_db.mysql

CREATE DATABASE sympa;

## Connect to DB
\r sympa

CREATE TABLE user_table (
   email_user                varchar (100) NOT NULL,
   gecos_user                varchar (150),
   password_user       varchar (40),
   cookie_delay_user   int,
   lang_user           varchar (10),
   PRIMARY KEY (email_user)
);

CREATE TABLE subscriber_table (
   list_subscriber           varchar (50) NOT NULL,
   user_subscriber           varchar (100) NOT NULL,
   date_subscriber           datetime NOT NULL,
   update_subscriber   datetime,
   visibility_subscriber     varchar (20),
   reception_subscriber      varchar (20),
   bounce_subscriber   varchar (30),
   comment_subscriber  varchar (150),
   PRIMARY KEY (list_subscriber, user_subscriber),
   INDEX (user_subscriber,list_subscriber)
);
```

When the base is created, you can redefine the access rights to the base `sympa` for user `sympa` :

```
# /usr/bin/mysql -u root -p
mysql> privilege all on sympa.* to sympa@localhost identified by
'your_password';
mysql> flush privileges;
```

13.3 Configuration of sympa.conf

Access parameters to the database are defined as followed in `sympa.conf` :

```
## extrait de sympa.conf
db_type         mysql
db_host         localhost
db_name         sympa
db_user         sympa
db_passwd       your_password
```

Contrary to line clients (mysql, mysqladmin), C and Perl (DBI) libraries don't use by default the my.cnf configuration file. To take account of your my.cnf, add in sympa.conf :
db_options mysql_read_default_file=/etc/my.cnf

The list parameter `user_data_source` defines the type of data sources for subscribers. The default value is `database` (use of the data base describes in `sympa.conf`). Other possible values are `include` and `file`.

13.4 Access tools to the database

13.4.1 mysql

`mysql` is the line client given with MySQL. It can execute SQL requests. It also has the advantages of the shell Bash : completion, historic.

Example : list of subscribers of the list cours-sympa-20.03.2002

```
# /usr/bin/mysql sympa
mysql> SELECT user_subscriber,date_subscriber FROM subscriber_table WHERE
list_subscriber = 'cours-sympa-20.03.2002';
+-------------------------------------+---------------------+
| user_subscriber                     | date_subscriber     |
+-------------------------------------+---------------------+
| alain.defrance@univ-xxx.fr              | 2002-02-13 10:05:24 |
| ameyer@xxx.fr                           | 2002-02-26 08:18:41 |
| anas.agoumi@etu.xxx.fr                  | 2002-02-12 12:00:17 |
| andre.lagadec@xxx.gouv.fr               | 2002-02-08 15:37:20 |
.....................................
+-------------------------------------+---------------------+
79 rows in set (0.01 sec)
```

13.4.2 phpMyAdmin

`phpMyAdmin` makes it possible to manage your MySQL databases via a web interface. It allows easily (without SQL knowledge) to :

- create/modify/remove bases/tables/fields
- search/insert/edit/remove recordings

14 Optimizations / and load balancing

14.1 Optimization of the distribution

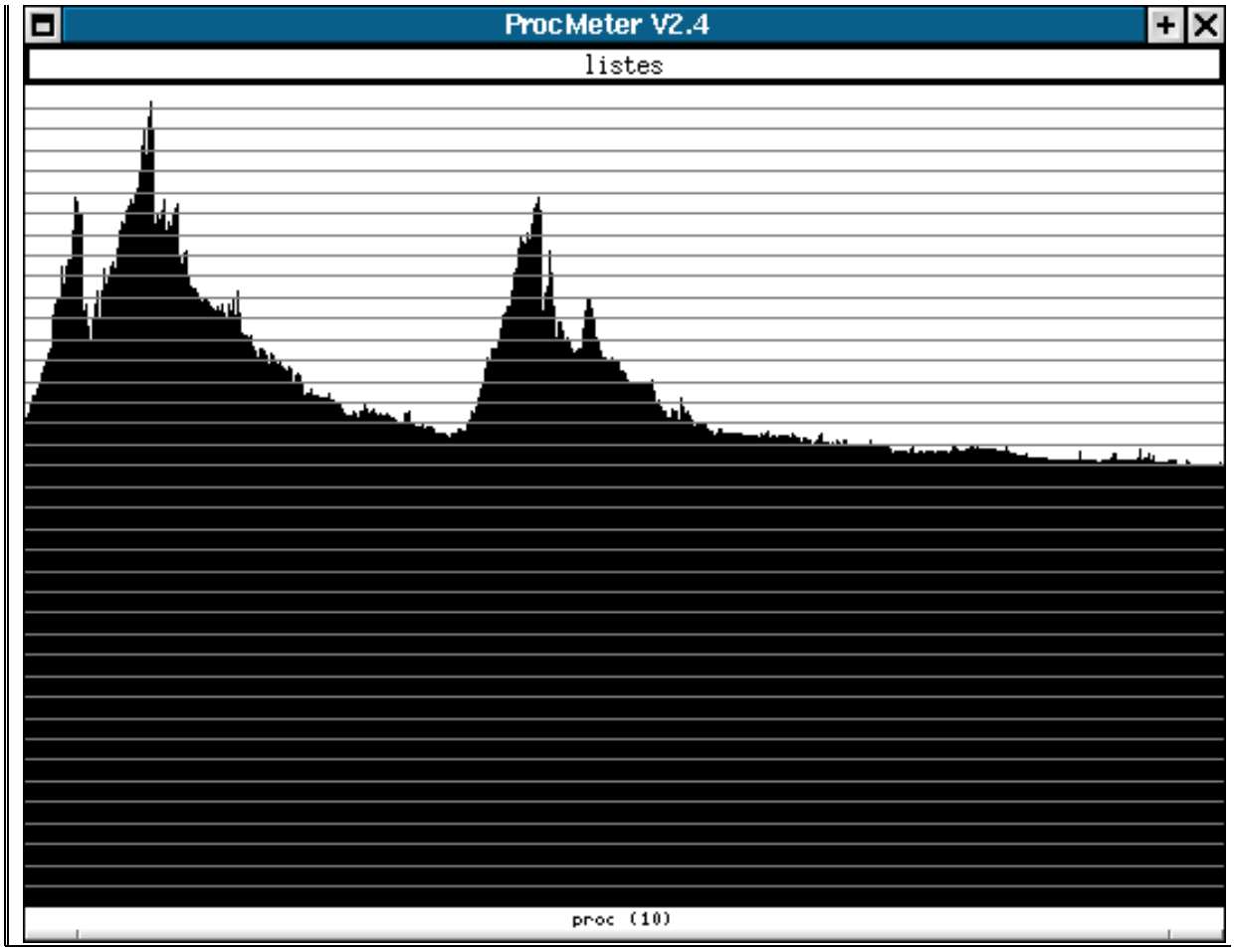
Sympa does not include a SMTP client, it uses the services of the local MTA (sendmail, postfix, exim or qmail) via a system call (the recipients are passed in parameters, the message sent in the STDIN). Several parameters of sympa.conf define the way to use the MTA :

- `sendmail`, `sendmail_args` : program (compatible with sendmail) to use to send messages ;
- `maxsmtp` (default 20) : maximum number of son sendmail processes thrown by Sympa ;
- `nrcpt` (default 25): max. number of recipients by call to sendmail ; define the collecting factor ;
- `avg` (default 10): maximum number of internet domains by call to sendmail.

The grouping factor (parameters `nrcpt` and `avg`) is set according to many criterias :

- The MTA used : MTAs have a very different behavior towards the grouping. Sendmail estimates the grouping on the base of recipient only, received as arguments, but it can send the body of the message only once for several recipients. Postfix tries a delivery to contiguous MTA and keeps the grouping of recipients (several RCPT TO : for one DATA). Qmail doesn't do any grouping ;
- Size of the lists : in a small one, there are few subscribers from the same domain, very quickly we reach `avg` recipients by packet. In a big list, with more subscribers per internet domains, it is the `nrcpt` parameter that will limit the grouping.
- An anti-spam policy from remote sites : some ISP (Yahoo at least) reject some SMTP sessions whose recipient number is judged to be suspicious. Make sure to keep the `nrcpt` parameter under a reasonable limit.

This diagram shows the evolution of the number of processes on a Sympa server. Child processes of Sympa, when their tasks are over, are not made free by sympa.pl until this one has reached its limit (maxsmtp) or when the processing of the current message is over. This leads to a massive release of processes (serrate). However, note that « defunct » processes don't use any resources on the server.



14.2 Optimization with Sendmail

Two rival goals can be researched :

1. increase the reactivity of the service by decreasing the processing time of a message: we increase `maxsmtp` even if it may load the server ;
2. decrease the load of the computer (decrease `maxsmtp`) for example to keep, during periods of high use, fast response time on the web interface ; though it may take more time.

We get a good arrangement by optimizing Sympa's child processes.

14.2.1 Non canonification of addresses

In Sympa, the call to Sendmail jams until Sendmail has a potential CNAME from the DNS for each recipient passed in argument. This DNS resolution is not necessary ; moreover if the DNS is unavailable, the lists server is paralyzed.

We put this functionality in place only for outgoing sendmail calls via Sympa by using a sendmail.cf specific to Sympa. In this configuration, the number of sendmail processes in memory increases much more quickly.

See FEATURE « nocanonify ».

14.2.2 Pull down timers

We can optimize some timers of SMTP sessions whose default values don't fit the very bad state of availability of some mailhosts. Indeed, these timers are in minutes or in sets of ten minutes. Then, only a dozen overloaded servers amongst your big correspondents are enough to make SMTP sessions available made for the most part (even exclusively) of sessions waiting for an answer from the remote server. These settlements target a quick decrease in the number of sendmail in memory in order to process other messages. Of course, consequently the management of spools has to be set up .

The `iconnect` timer is the most interesting, we have changed its value :

```
O Timeout.iconnect=17s
```

See configuration options.

14.3 MySQL optimization

14.3.1 Structure of the base

Since version 3.23 of MySQL, tables are stored by default in the MyISAM format, offering better performances than its ancestor ISAM. To move older table to the MyISAM format do :

```
#/usr/bin/mysql sympa
mysql> ALTER TABLE subscriber_table TYPE = MYISAM;
Query OK, 46307 rows affected (4.49 sec)
Records: 46307 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE user_table TYPE = MYISAM;
Query OK, 140355 rows affected (8.51 sec)
Records: 140355 Duplicates: 0 Warnings: 0
```

The `myisamchk` command makes it possible to do maintenance operations on tables of the database. Example : increasing the size of the buffer for sorting

```
# /usr/bin/myisamchk -O sort_buffer_size /var/lib/mysql/sympa/*.MYI
```

The OPTIMIZE TABLE request makes it possible to also optimize a MySQL table by releasing non-used space and by defragging data files.

```
# /usr/sbin/mysql sympa
mysql> OPTIMIZE TABLE subscriber_table;
+------------------------+----------+----------+----------+
| Table                  | Op       | Msg_type | Msg_text |
+------------------------+----------+----------+----------+
| sympa.subscriber_table | optimize | status   | OK       |
+------------------------+----------+----------+----------+
1 row in set (7.74 sec)
```

14.3.2 Configuration of the server

At startup, the MySQL daemon reads values of its options defined in the configuration file `/etc/my.cnf`. To have a list of options at « runtime » :

```
# mysqladmin variables
+-------------------------------------------+
| Variable_name          | Value            |
+-------------------------------------------+
| ansi_mode              | OFF              |
| back_log               | 50               |
| basedir                | /usr/            |
……
```

Examples of configuration files adapted to different uses are given with MySQL (`my-huge.cnf`, `my-large.cnf`, `my-medium.cnf`, `my-small.cnf`), in /usr/share/mysql/ for us. You can take them to optimize you setup.

By default the MySQL server doesn't log operations on bases. To have logs, add the following entry to you /etc/my.cnf :

log = /var/log/mysql

14.4 Load balancing

It can be interesting to balance services of lists on several servers. The simplest solution is to carry off external software from Sympa :

- The data base (MySQL, PostgreSQL …) : see the parameter `db_host` of sympa.conf.
- The e-mail server (sendmail , postfix …) : you can transmit out going traffic of the lists servers to an another server (or several by doing Round Robin DNS).

See experiment use of mini sendmail.

You can also separate Sympa components, to know :

- mail treatment : `sympa.pl ;`
- web interface : `wwsympa.fcgi ;`
- management of non-given report : `bounced.pl ;`
- Management of web archives : `archived.pl.`

But the separation of this software makes it necessary to separate data between servers :

- database
  - ⇨ every software must have an access
- configuration files (sympa.conf & wwsympa.conf)
  - ⇨ NFS
- ~sympa/etc/
  - ⇨ NFS
- ~sympa/spool/
  - ⇨ NFS
- ~sympa/expl/
  - ⇨ NFS
- reports of non-delivery (used by bounced.pl and wwsympa.fcgi)
- web archives (used by archived.pl and wwsympa.fcgi)

15 Setup of Sympa

The setup of Sympa is simple. After downloading the current version from http://www.sympa.org, and creating a dedicated user `sympa.sympa`, extract files from the tar, and proceed to the setup :

```
% ./configure
% make
% make install
```

Of course, this is not the only thing to do. You must also install other products like antivirus, e-mail server, the database and the HTTP server. You will find steps to make these configurations in Sympa reference manual. Some choices must be done during the setup or later when upgrading the service :

- Choice of the e-mail agent (sendmail, exim, easy-sendmail, postfix, qmail) ;
- Choice of a RDBMS (MySQL, Sybase, Oracle, PosgreSQL) ;
- Choice of an antivirus software (MacAfee/Uvscan, Fsecure/fsav, Sophos, AVP, Trend Micro/VirusWall) ;
- Choice of a HTTP server (Apache, Roxen) ;
- Choice of a binary setup (.rpm, .deb) or from sources. ;
- One server or distribution on several computer.

16 Integrating Sympa with other software

16.1 Share the web authentication

Sympa allows you to unify your authentication system with your other web software. It uses a HTTP cookie to carry authentication data ; however this cookie doesn't have any idea of privilege.

16.1.1 Use the authentication system of Sympa

In this case, you will re-use Sympa's login functions, logout and password reminder. You can integrate a login form in your software, the CGI called remain wwsympa.fcgi. Your duty is to use Sympa's HTTP authentication cookie (check, e-mail address extraction).

In order that wwsympa.fcgi redirects users in a transparent way to your software, add */referer* to the end of a wwsympa's called URL.
Example :
<A HREF="/wws/loginrequest/referer">Page of Login</A>

16.1.2 Sympa recognizes your authentication

Your software manages the authentication system, but puts a Sympa cookie in order to be recognized by it. The software must share a « secret » with Sympa (`cookie` parameter in sympa.conf) used to generate the HTTP cookies. In this case, Sympa will recognize your users as identified. Example of this type of integration : Rearsite.

16.2 Data share

16.2.1 Definition of lists by extraction from a database

By default, members of a list are subscribers, managed in the dedicated base of Sympa. Members of the list can be the result of a request in the database (or an LDAP directory). In this case, the lists of members are regularly updated (parameter of list `ttl`) by questioning the base. Several sources of data can be defined for the same list.

Configuration extract of list :

```
# les membres de la liste sont extraits d'une  source de données externe
user_data_source include

# Mise à jour des données toutes les 12h (43200 s)
ttl 43200

# Définition d'une liste d'étudiants
include_sql_query
      db_type mysql
      host sqlserv.admin.univ-x.fr
      user stduser
      passwd mysecret
      db_name studentbody
      sql_query SELECT DISTINCT email FROM student
      connect_options mysql_connect_timeout=5
```

Data cache : currently the data cache is stored in a file DB. Future developments would makes it possible to manage the cache in a data base (MySQL), which would give the following advantages :

- Enable you to define mixed lists include/subscription ;
- Give available subscription options to everyone (included DB extraction) ;
- Reduce the size of Sympa's process, data being managed by MySQL.

16.2.2 Add your data to Sympa's base

You have the possibility to extend the structure of the tables `user_table` and `subscriber_table` in order to add fields for your private use (example : payment mechanism for lists). Sympa will keep these fields and will even make them available in templates and scenarios. The added fields must be declared in sympa.conf.

Fields added to the `subscriber_table` table are made available from within the web interface which manages subscribers.

Extract of sympa.conf :

```
# champs sup. dans subscriber_table
db_additional_subscriber_fields     reglement_ok,duree_abo

# champs sup. Dans user_table
db_additional_user_fields           adresse_postale,num_rib
```

17 S/MIME et HTTPS

The password is not the only way to authenticate. The arrival of Public Keys Infrastructures (PKI) make it possible to set up methods which are both more reliable and practical. Sympa uses an encryption library (OpenSSL) making it possible to use X509 certificates in the whole software. This section aims at helping you to put these functionalities in place quickly. If you would like to know more read the reference handbook and the article « Sympa S/MIME and Sympa mailing lists manager » http://listes.cru.fr/sympa/documentation/article_smime/sympasmime.html.

17.1 HTTPS

As everyone knows we distinguish two operation modes from HTTPS :

The encrypted mode without client authentication. In this case we don't use the client's certificate ; the only X509 certificate to be used is the server's. This type of setup is particularly simple to deploy, but the service given back by the SSL layout is limited to an encrypted exchange that no one can listen. In the case of Sympa, sensible data are of course passwords and cookies. Such a setup consist of :

- Installing OpenSSL and mod_ssl for Apache
- Installing a server certificate

Read the very good documentation of mod_ssl (http://www.modssl.org) for these operations. You will only have to modify the Sympa configuration : update the variable `wws_url` of `sympa.conf` (https instead of http).

The encrypted mode with client authentication. In this case, clients provide their certificate (automatically). Sympa has been modified in order to use data extracted from client's certificates which enables the removal of the authentication step. The login button of Sympa no more appear and each page is shown with the privileges of the person identified by the e-mail address given by his/her certificate with the authentication `smime` method. This result is reached by grouping the following two points of configuration :

- The Apache server is configured to ask the client for a certificate: `SSLVerifyClient Optional` or `SSLVerifyClient Require ;`
- StdEnvVar option of mod_ssl is defined in Apache in order to allow Sympa to inherit environmental variables extracted from certificates : `SSLOptions +StdEnvVars.`

17.2 S/MIME

Sympa can also use X509 certificates through signed and/or encrypted messages and/or emitted by him. The goal is then to authenticate the sender of a message based on the S/MIME signature of the message or to distribute encrypted messages.

17.2.1 Validation of S/MIME signature

Openssl being installed, you only have to fill the `openssl` and `trusted_ca_options` parameters `in sympa.conf`. We of course have to ensure coherence with the trusted authorities in the Apache context by sharing the same certificate files of with the PEM format :

An example extracted from our own sympa.conf :

```
##  path to OpenSSL command
openssl         /usr/local/ssl/bin/openssl

#  directory where trusted CA certificat are stored
trusted_ca_options   -CAfile /usr/local/apache/conf/ssl.crt/ca-bundle.jhgjh
```

Sympa can then validate signature of S/MIME messages.

17.2.2 Encrypted messages Distribution

Encrypted distribution of messages in a list is possible if an X509 certificate and an associated private key are installed for the list. This setup can be done with the ~sympa/bin/`p12topem.pl` script from a certificate and its key in the `pkcs#12` format.

The welcome message of the list is then signed (by using S/MIME) which allows the subscriber's e-mail client to store this certificate. He/she can then send encrypted messages to the list as he/she would for anybody else.

Sympa can encrypt a message for a user only if it has the certificate of this user in its cache. It is required that it received at least one S/MIME signed message from each user of the lists. Indeed Sympa keeps all the certificates extracted from signed messages received in its `~sympa/expl/X509-user-certs` directory.
It is therefore advisable to require subscription command to be S/MIME signed for these lists (use of `smime` authentication method in the `subscribe` scenario).

18 Available ressources

This article is available on www.sympa.org web site in the « documentation » section. You will also find there reference documentation in HTML, Postscript and PDF formats. A FAQ answers the most frequently asked questions in the support mailing lists. Eventually we manage mailing lists about Sympa (sympa-fr, sympa-users, sympa-dev, sympa-announce and sympa-translation). We remind you that web archives of these lists are also available.

Several companies providing commercial support on Sympa are referenced in the « technical support » section of the web site.

Sympa's code is managed with CVS (Concurrent Versions System). The CVS server and its web interface give you access to the latest development version of Sympa. In a general way, it is not reasonable to install the current CVS version, which may have not been tested at the CRU. Web access to CVS can help you to find out about development state of a module, bug fixes, or patches download.