Le serveur de listes de diffusion Sympa

Mars 2002

Serge Aumont & Olivier Salaün
CRU

Vincent Mathieu
Université de Nancy 2

Sympa est l'aboutissement de nos investissements pour répondre aux besoins d'un service national pour les universités dans le domaine des listes de diffusion. Cette activité a démarré en 1992 pour préparer la migration des listes de *bitnet.* Aujourd'hui, Sympa est un produit sous licence GPL dont les développements continuent sous le contrôle du CRU et avec de nombreuses contributions de la communauté des utilisateurs. Plus de 3000 sites de toutes origines chargent régulièrement les nouvelles versions de notre « ***SY****stème de* ***M****ulti-****P****ostage* ***A****utomatique* ».

Il serait vain de réécrire le manuel de référence de Sympa, mais la mise en œuvre de Sympa est souvent jugée complexe par les listmasters. C'est bien entendu la rançon de la grande richesse de fonctionnalités de cette application. Ce cours a donc pour objectif de vous aider à exploiter tous les dispositifs de Sympa visant à simplifier son administration tout en offrant des services améliorés.

Ce cours a été donné en mars 2002, il est basé sur la version 3.3.4 de Sympa.

1 Panorama des fonctionnalités de Sympa

Il n'est pas question d'énumérer toutes les spécificités de ce serveur de listes de diffusion, tout au plus pouvons nous donner un aperçu des points originaux de Sympa.
Une des caractéristiques de Sympa est d'offrir deux interfaces (messagerie et web) intégrées. L'interface WEB est unique et dispense à chacun une vue personnalisée du service de listes. Ainsi, les usagers, les propriétaires de listes et le(s) *listmaster(s)* utilisent la même interface.

1.1 Service aux usagers

Sympa propose aux usagers des listes de diffusion toute la panoplie des services classiques dans ce domaine : abonnement / désabonnement, option d'abonnement pour recevoir les messages sous différents formats, archives indexées, présentée par date ou par fil de discussion (thread), espace web permettant le dépôt de documents réservés aux abonnés de la liste etc.
Il convient de souligner que le service est accessible par deux interfaces :

- l'interface messagerie qui reconnaît les jeux de commandes de la plupart des robots usités et s'accommode sans difficultés des messages formatés avec toutes les possibilités de MIME (reconnaissance des commandes de messagerie placées dans des messages multipart/alternative).
- l'interface web qui se présente sous la forme d'un portail des listes de diffusion du site. Elle propose à chacun une vue personnalisée du service de listes de diffusion dès que la personne s'identifie grâce à un dispositif d'allocation/ré-allocation des mots de passe. L'usager peut alors visualiser l'ensemble de ses abonnements, positionner des préférences (choix de la langue de l'interface par exemple), supprimer des archives les messages dont il est l'auteur etc.

Chaque abonné peut choisir pour chaque liste des options d'abonnement comme la réception périodique et groupée des articles, le remplacement dans les messages des attachements par des URLs, la réception exclusive de la version texte ascii ou de la version html quand les deux alternatives sont contenues dans le message.

1.2 Service aux propriétaires et aux modérateurs de listes

Sympa distingue les modérateurs (qui valident ou refusent la diffusion de messages dans les listes) et les propriétaires (qui configurent la liste et gèrent les abonnements). Authentifiés avec son mot de passe le propriétaire de liste dispose sur l'interface web d'une vision spécifique de ses listes. Pour chaqune d'entre elle il peut :

- rédiger les messages de service (message de bienvenue, rappel d'abonnement, message de désabonnement,, …) et la page d'accueil de la liste.
- accéder et configurer les paramètres de sa liste (ceux que le listmaster a rendu éditable) en particulier pour définir la population des gens autorisés à s'abonner à sa liste, à y diffuser des messages etc.
- valider ou refuser (par mail ou par web) les demandes d'abonnement qui sont soumises à validation ainsi que la diffusion des messages soumis à validation.

1.3 Service au listmaster

Un ou plusieurs lismasters sont définis pour chaque robot de listes (Sympa permet de gérer plusieurs robots de listes relatifs à des domaines différents). Le listmaster valide sur l'interface web les demandes de création de liste quand elles sont soumises à autorisation. Il configure les comportements par défaut de Sympa ainsi que les messages de services non spécifiques à une liste. Il définit entre autre la population des personnes autorisées à créer des listes de diffusion etc.

1.4 Autres caractéristiques de Sympa

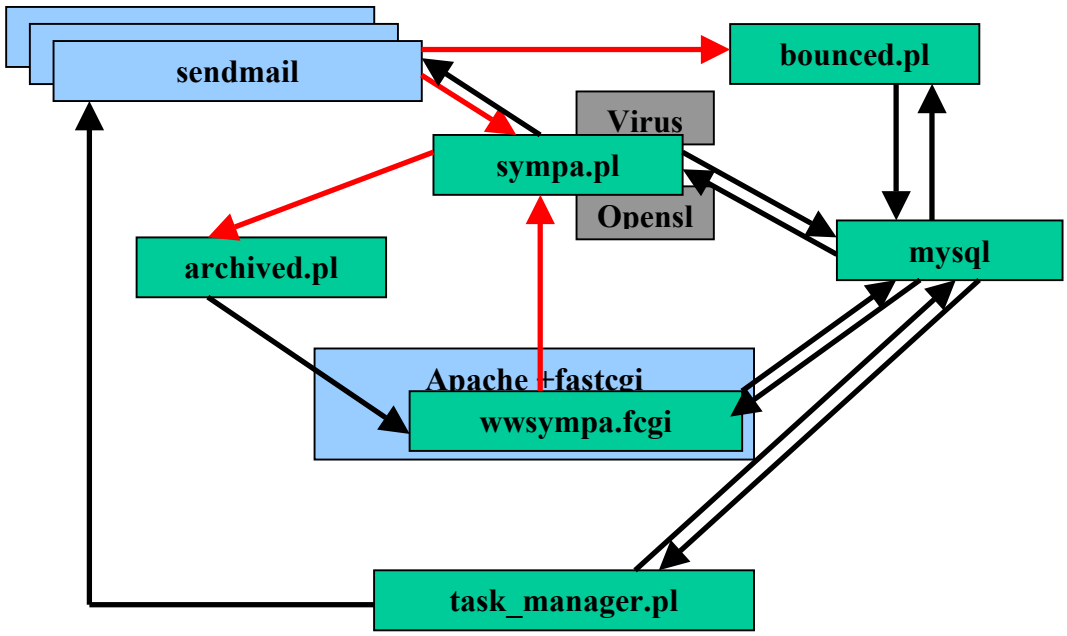
Le système de gestion des privilèges appelé *scénario* permet de définir le fonctionnement en fonction de tous les éléments du contexte (catégorie d'utilisateur, présence de pièces jointes dans le message, utilisation depuis un poste du réseau local etc). En outre Sympa intègre un système d'authentification par mot de passe ; celui-ci peut être commun à d'autres applications en particulier en utilisant l'authentification LDAP.

Sympa autorise deux modes de définition de la population des listes d'abonnés : les classiques listes construites par abonnement/désabonnement et les listes dynamiques directement extraites d'annuaires LDAP ou de serveurs de base de données.

Dans le domaine des certificats X509, Sympa peut reconnaître les signatures S/MIME mais aussi chiffrer des messages à destination des abonnés, l'authentification HTTPS est aussi possible sur l'interface web.
A noter que tous les messages diffusés sont traités par un anti-virus.

2 Panorama de l'organisation de Sympa

Un petit coup d'œil sur l'ensemble de l'organisation de Sympa est indispensable pour en appréhender le fonctionnement et l'ensemble des possibilités.



- *sympa.pl* : c'est le démon principal, il reçoit et diffuse les messages via sendmail (postfix, qmail ou exim). Il consulte et modifie les listes d'abonnés dans la base de données. Il alimente *archived.pl* après diffusion d'un message. Il appelle un antivirus externe pour tous les messages destinés relayés vers un modérateur ou diffusés.
- *wwsympa.fcgi* reçoit des requêtes du serveur HTTP (Apache ou Roxen). Grâce au module fastcgi, ce CGI est résident : il conserve les très nombreuses initialisations réalisées en mémoire. En particulier il maintient sa connexion avec le serveur SQL ainsi qu'une image mémoire des objets liste. *wwsympa.fcgi* fait des mises à jour dans la base de données. Il délègue la diffusion des messages à Sympa via un spoule
- *bounced.pl* est le démon de traitement des rapports de non-remise. Il reçoit ceux-ci du moteur SMTP et met à jour la base de données.
- *archived*.pl est le démon d'archivage. Il traite les messages du spool « outgoing » et les convertit en template HTML utilisés par wwsympa.fcgi.
- *task_manager.pl :* actuellement ce démon est utilisé uniquement pour émettre périodiquement les messages de rappel des abonnements et pour supprimer les bounces non significatifs. De nombreuses tâches lui sont réservées à l'avenir.

3 Organisation des données

Le modèle de donné de Sympa se compose de :
- la base de données Sympa,
- un ensemble de fichiers.
- Selon la configuration, des sources de données externes : des bases de données (LDAP ou SQL) pour la constitution des listes dynamiques,
- annuaires LDAP pour l'authentification.

Ne pas confondre la base de données Sympa (en général MySQL) et la ou les bases de données externes à Sympa utilisées pour des includes. Dans le second cas, Sympa ne fait que des accès en lecture à ces données externes.

3.1 Structure de la base de données

Bien que, pour des raisons « historiques », l'installation d'un SGBD pour la gestion des tables d'abonnés soit encore mentionnée comme optionnelle, il est impératif d'installer une base de données si l'interface web est installée. En effet, seule la base de données assure la gestion des conflits d'écriture entre les processus fastcgi et les différents démons. Le SGBD n'est donc pas seulement un moyen d'améliorer les performances de l'application. Plusieurs SGBD sont supportés : MySQL, Postgresql, Sybase et Oracle. Le mode d'accès à la base est configuré dans le fichier `sympa.conf`. La base est constituée de deux tables :

1. la table user_table qui permet de stocker les infos des utilisateurs (email, mot de passe et langue de préférence)
2. la table subscriber_table : les informations sur les abonnements

3.2 Structure des fichiers

On se reportera avec avantage à la section « organisation » du manuel de référence de Sympa : http://listes.cru.fr/sympa/doc/sympa/node3.html#SECTION00310000000000000000 et aux exemples de ce document. On distingue 3 familles de fichiers :

- `sympa.conf` et `wwsympa.conf` : la configuration globale de l'application
- les fichiers de configuration de robot virtuel : robot.conf, edit_list.conf, templates, scénarios, …
- Les fichiers relatifs à une liste : config, stats, archives, bounces, …
- Les spoules

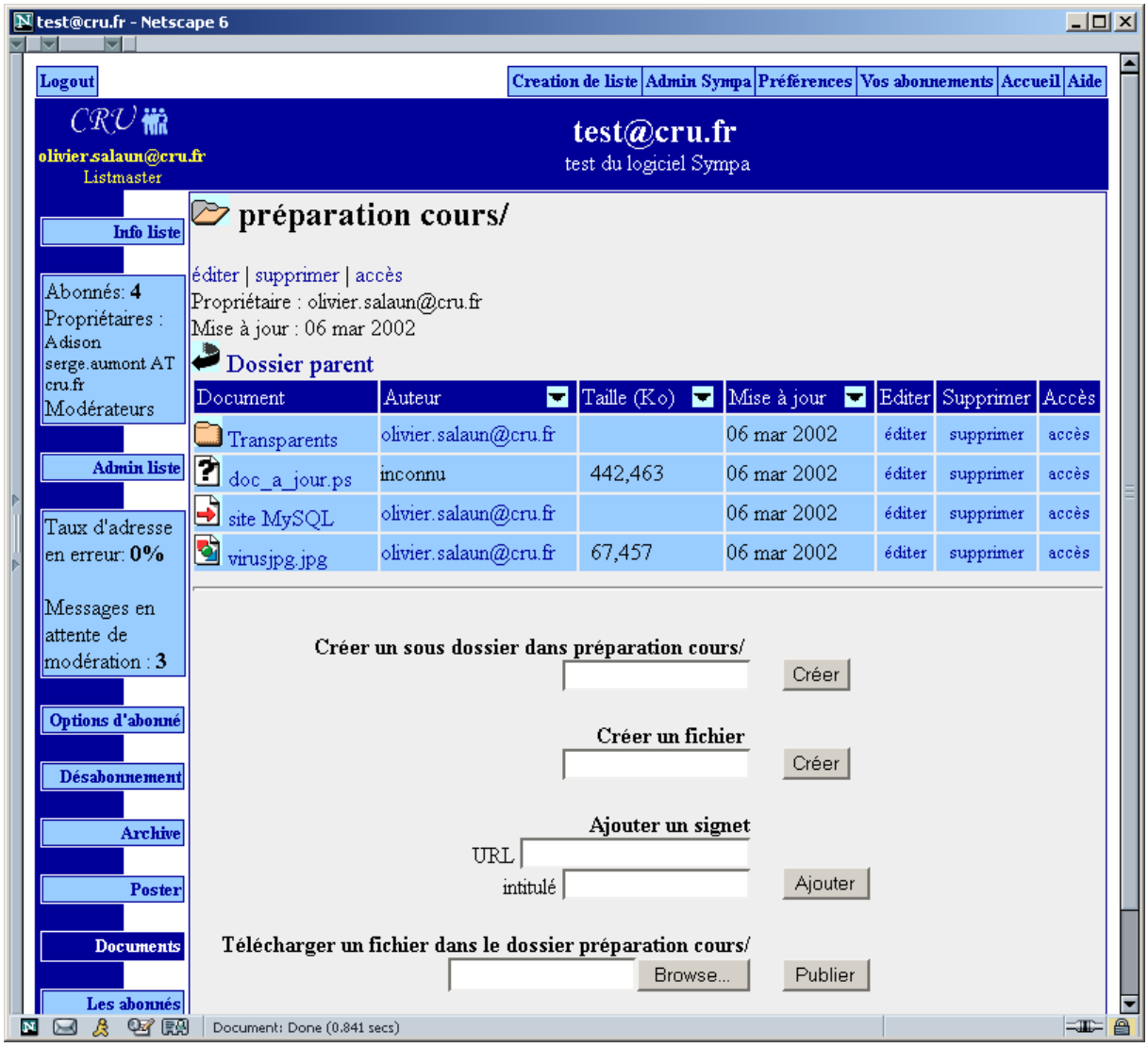
4 L'espace de dépôt de fichiers web

Un espace de dépôt de fichiers est disponible pour chaque liste ; il est accessible depuis la page web de la liste. Les ambitions de cet espace documentaire sont modestes, il ne s'agissait pas de proposer un outil de gestion de site web (voir Rearsite) mais juste un espace d'échange de documents et de signets entre membres d'une liste.

4.1 Fonctionnalités

La fonctionnalité principale est le dépôt (upload) de fichiers ; il est possible d'éditer, de renommer, de supprimer les fichiers déposés. Un descriptif est associable à chaque fichier. L'espace documentaire permet également de gérer des dossiers et des signets (URLs).

Par défaut, l'espace de dépôt d'une liste est inactif ; l'activation est effectuée par le propriétaire de la liste, via l'interface web d'administration.



Il est possible de déposer un fichier `index.html` dans un dossier, il remplacera l'index des fichiers lors de la consultation du dossier.

L'accès en consultation aux documents dans l'espace de dépôt est complètement contrôlé par WWSympa qui se base sur son système d'authentification (interne ou LDAP) pour identifier le client.

4.2 Définition et héritage des privilèges

L'espace de dépôt permet de définir 3 types de privilèges sur chacun de ses éléments :

- consultation : droit d'accès à un document, à un dossier
- édition : droit d'éditer un fichier, de le renommer/supprimer. Définit également le droit de dépôt dans un répertoire.
- contrôle : permet de définir les droits en consultation/édition sur un élément de l'espace de dépôt. Ce privilège permet également de changer le propriétaire du fichier/dossier.

Chaque nouvelle entrée (fichier / dossier / signet) hérite des privilèges du dossier auquel il appartient. Les privilèges associés au dossier racine sont définis par le paramètre de liste `shared_doc` (scénarisé), à l'exception du privilège de contrôle, assimilé au droit d'éditer le paramètre `shared_doc` (par défaut restreint au propriétaire de la liste).

Extrait de config de liste :

```
shared_doc
   d_read public
   d_edit private
```

Les privilèges sur un fichier dépendent bien sûr des droits définis pour ce fichier, mais également de ceux de son ascendance ; ce principe permet de garder le contrôle sur une hiérarchie de l'espace de dépôt. Soit le fichier dont le chemin complet est D1/D2/F1, les privilèges suivent le schéma suivant :

- consultation et édition : priv(F1) = priv(D1) ET priv(D2) ET priv(F1)
- contrôle : priv(F1) = priv(D1) OU priv(D2) OU priv(F1)

5 Les options d'abonnement

La configuration de chaque liste permet de définir les options de réception des messages disponibles (paramètre *available_user_options*) et l'option d'abonnement par défaut (paramètre *default_user_options*). Chaque abonné peut donc choisir une option d'abonnement. Citons les options :

- **`nomail`** : pas de réception utile pour suspendre la réception des messages tout en conservant les privilèges d'abonnés.
- **`digest`** : réception groupée et périodique des messages (Sympa utilise le format Mime multipart/digest, la périodicité est configurable)
- **`summary`** : idem digest mais réception limitée à la liste des messages
- **`urlize`** : les messages sont distribués normalement mais les attachements sont remplacés par des liens
- **`text`** et **`html`** : permet de spécifier la version préférée, dans le cas ou le message diffusé contient les deux formes (multipart/alternative)

6 Antivirus

Sympa est prévu pour utiliser les services d'un antivirus pour analyser les messages entrant ; les antivirus suivant sont reconnus :

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

Vous devez positionner les paramètres `antivirus_path` et `antivirus_args` dans `sympa.conf` pour activer la détection des virus (dans les messages entrants) dans

`sympa.pl`. Les messages contaminés sont mis de coté (répertoire `bad/`) ; l'auteur du message est prévenu (voir le template `your_infected_msg.tpl`).

Une semaine de virus au CRU (utilisation uvscan) :

```
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
```

En moyenne, le serveur Sympa du CRU, intercepte plus de 100 messages contaminés par semaine.

7 Le traitement des bounces (ou rapports de non remise)

Traditionnellement, les rapports de non remise pour les messages diffusés dans les listes sont transmis aux propriétaires de la liste ; leur boîte aux lettres est alors submergée par ces rapports peu lisibles pour le commun des mortels. Pour résoudre ce problème Sympa traite les bounces automatiquement, l'opération finale de désabonnement étant laissée (pour l'instant) aux propriétaires.

7.1 L'analyse des bounces, bounced.pl

Les bounces sont reçus à l'adresse `maliste-owner` puis stokes dans un spool par le programme `bouncequeue`.

Exemple d'alias :

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

Le démon `bounced.pl` traite les rapports de non remise dans le spool pour en extraire l'adresse de l'abonné en erreur et le type d'erreur. Le champ `bounce_subscriber` (table `subscriber_table`) est mis à jour ; le dernier rapport de non-remise de l'abonné est archivé (répertoire d'archives défini par le paramètre `bounce_path` de `wwsympa.conf`).

Le champ bounce_subscriber (table subscriber_table) :

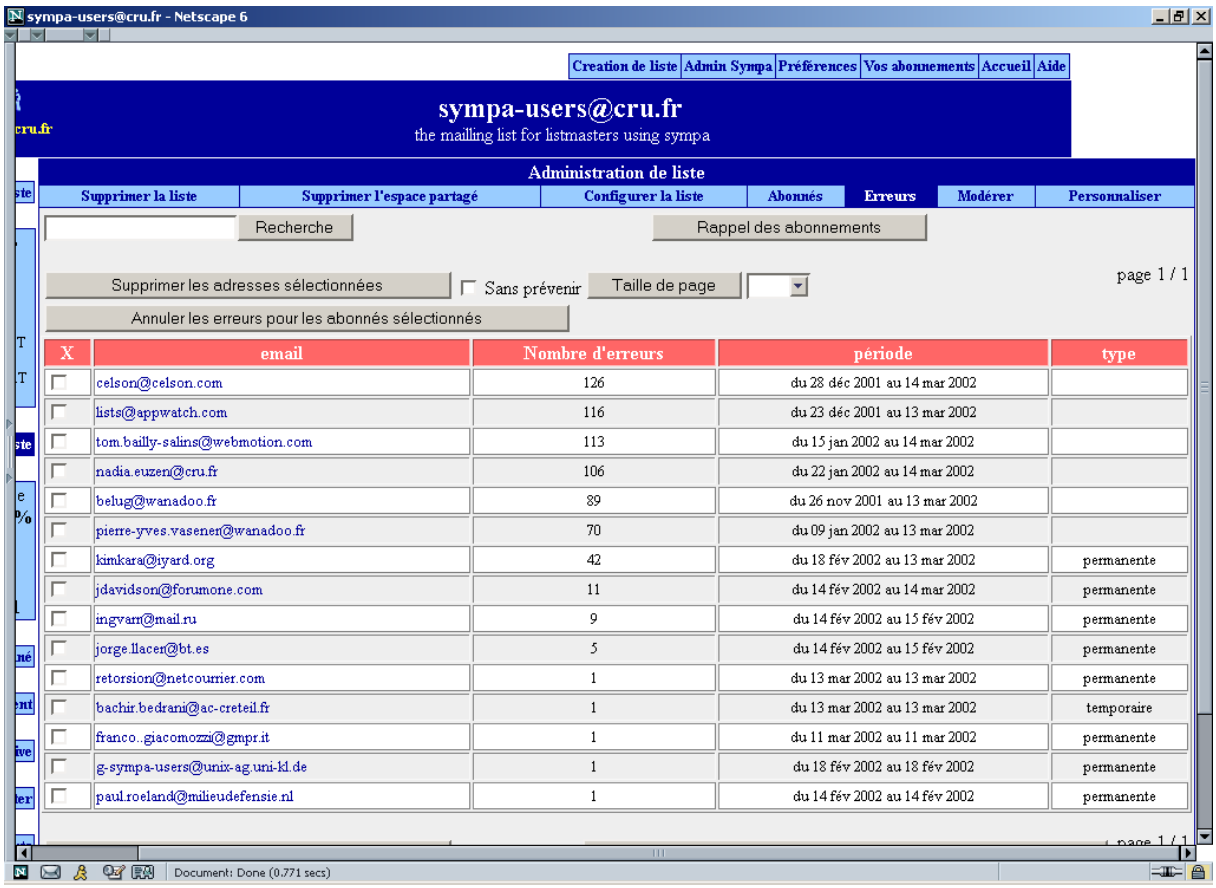
```
1014647647 1016081883 97 5.2.2

   1014647647 : date de 1ère erreur (25 février 2002)
   1016081883 : date dernière erreur (14 mars 2002)
   97 : nombre de bounces reçus
   5.2.2 : type de l'erreur (défini par RFC 1891)
               5 => erreur permanente
               2.2 => mailbox full
```

Si le taux d'abonnés en erreur dans une liste dépasse un seuil (défini par le paramètre bounce_warn_rate) le propriétaire recevra une alerte pour chaque message diffusé dans la liste. Ce message est actuellement défini dans les NLS (8,28).

7.2 L'interface web de gestion des bounces

L'interface web d'administration d'une liste permet de gérer les abonnés en erreur. La liste des adresses en erreur permet de visualiser la fréquence des erreurs, leur type. Le propriétaire de la liste peut alors choisir d'annuler les erreurs ou de désabonner l'adresse. L'abonné recevra alors un message de désabonnement défini dans le template `bye.tpl`(sauf case « sans prévenir » cochée).



Interface web de gestion des bounces 1

7.3 Expiration des bounces

Lorsqu'une adresse ne génère plus d'erreurs, les erreurs reçues doivent être oubliées par le système. Cette tâche est confiée au gestionnaire de tâches de Sympa, via le paramètre `expire_bounce` de sympa.conf. Le scénario par défaut (et le seul actuellement) expire les bounces des abonnés n'ayant pas généré de bounce 10 jours après la dernière diffusion dans la liste.

La tâche 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 Projets de développement

Depuis la version 3.3.3 Sympa effectue une corrélation entre les bounces et la diffusion des messages (via le fichier `msg_count`). Cette information devrait permettre dans les versions à venir d'effectuer un certain nombre de désabonnements automatiques. Dans certains cas cependant la décision restera à la charge du propriétaire de la liste.

8 Le gestionnaire de tâches

Sympa a besoin d'effectuer certaines opérations planifiées ; il utilise pour certaines de ces opérations les services du gestionnaire de tâches. Le gestionnaire de tâches est l'équivalent pour Sympa de la *crontab*. Le démon `task_manager.pl` planifie les tâches sur la base de certains paramètres de listes (expire_task, remind_task) et des paramètres globaux (crl_update, expire_bounce). Les tâches pouvant être complexes, le démon stocke l'état de chaque tâche dans un spool.

A terme, certaines opérations répétitives traitées actuellement par sympa.pl devraient l'être par le gestionnaire de tâches (envoi des digests, ménage dans les spools, …).

Exemple de tâche planifiée (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 ()
```

Les models de tâches sont personnalisables (à l'instar des scénarios) dans les répertoires `list_task_models` et `global_task_models`.

9 Robots virtuels

La notion de robot virtuel est à Sympa ce que les « hosts virtuels » sont à Apache : le moyen d'héberger plusieurs services de listes de diffusion à partir d'une seule installation de Sympa et une administration simplifiée. Du point de vue des utilisateurs (abonnés, propriétaires de listes, modérateurs), les services apparaissent comme totalement indépendants. Exemple : https://listes.cru.fr/wws ,
https://listes.renater.fr/wws ,
http://listes-dgcid.diplomatie.gouv.fr/wws, …

9.1 Organisation interne des Robots virtuels.

A chaque robot virtuel correspond une adresse de robot `sympa@domaine` . Les listes d'un robot sont gérées dans le domaine robot virtuel . Les listes de diffusion ne sont visibles que via le robot du même domaine.
Pour réussir la configuration d'un robot virtuel, il faut comprendre comment Sympa et WWSympa choisissent le contexte de robot virtuel à appliquer pour chaque message ou requête HTTP

1. sympa.pl traite des messages qui ont été déposés dans le spoule des messages via le programme **`queue`** et les alias du moteur de messagerie. Exemple :

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

   sympa.pl récupère l'argument du programme `queue` pour identifier la liste et le robot de référence (une entête `x-sympa-to: cours-sympa-20-03-2002@cru.fr` est ajoutée au message).

2. `wwsympa.fcgi` : chaque requête HTTP est qualifiée par la variable d'environnement SERVER_NAME (le host appelé par le client donc celui utilisé pour la définition du virtuel host Apache). Une table interne à Sympa construite à partir des fichiers de configuration de robot permet d'effectuer une correspondance entre ce nom de host et le domaine du robot virtuel. Par exemple `listes.cru.fr` sert le robot virtuel `cru.fr`.

9.2 Créer un robot virtuel

9.2.1 DNS

Définir dans le DNS un **`MX record`** pour le domaine virtuel et un **`CNAME`** ou un host (*A record*) pour le serveur web associé.

> Dans le cas d'un serveur HTTP**S** définir une nouvelle adresses IP et non un CNAME (contrainte liée à l'architecture de l'empilement du protocole http sur la couche ssl)

9.2.2 Configurer votre moteur SMTP

Votre MTA doit reconnaître le nouveau domaine.

- Avec sendmail et le kit Jussieu voir http://www-crc.u-strasbg.fr/docs/kit-jussieu/support/node115.html
- Avec sendmail et les macro m4 voir http://www.sendmail.org/virtual-hosting.html

- Avec Postfix voir http://www.postfix.org/virtual.5.html

9.2.3 Créer les alias :

```
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

Configurer un virtual host Apache. Il n'est pas utile de définir un/des serveur(s) FastCgi pour chaque robot virtuel, au contraire chaque serveur FastCgi est capable de servir n'importe quel robot virtuel. On trouve donc la définition des serveurs FastCgi dans la partie commune de la configuration d'Apache et le path de l'application dans chaque host virtuel :

```
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>
```

Le reste des opérations sera traité tôt ou tard via un formulaire de création de robot virtuel. En attendant ces raffinements. Il convient de créer « à la main » chaque environnement. Dans les exemples qui suivent, le robot virtuel sert le domaine *virtual.fr*.

9.2.5 robot.conf

Le robot virtual.fr est défini en répertoire `~sympa/etc/virtual.fr/` et un fichier `robot.conf` dans ce répertoire. Ce fichier de configuration permet de redéfinir éventuellement des paramètres de `sympa.conf` et de `wwsympa.conf` (exemple les couleurs des pages). Mais `robot.conf` doit obligatoirement définir le paramètre `http_host` qui sera utilisé pour établir la correspondance entre le nom du serveur HTTP référencé et le robot virtuel cible.

Description des paramètres de robot :

- `email` : la partie locale de l'adresse du robot : exemple `sympa`.
- `title` : le titre de la page d'accueil.
- `default_home` : la page d'accueil par défaut (home : classement par *topics* ou *lists* : classement alphabétique).
- `create_list` : nom du scénario contrôlant l'accès à la fonction de création de listes.
- `lang` : la langue par défaut du robot.
- `log_smtp` : activation du log des appels à sendmail.

- `log_level` : niveau de log.
- `listmaster` : adresses email des listmasters sparées par des « , ».
- `max_size` : taille maximale des messages (redéfinition possible pour chaque liste).
- `dark_color, light_color, text_color, bg_color, error_color, selected_color, shaded_color`

exemple :

```
# 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 Répertoire d'exploitation

Créer le répertoire qui contiendra les listes du robot virtuel : `~sympa/expl/virtual.fr`. Toute liste créée dans ce répertoire est affectée à ce robot.

9.2.7 Redémarrer Apache, Sendmail et les démons de Sympa

La prise en compte de nouveaux robots virtuels n'est pas dynamique pour ces trois applications.

9.2.8 Autre configuration

De même que chaque liste peut disposer d'un jeu spécifique de scénarios et de templates, ces éléments de configuration peuvent être définis au niveau d'un robot virtuel.

Lors de l'évaluation d'un scénario, par exemple le scénario **`intranet`** pour le contrôle de l'opération **`send`** dans la liste **`foo`** du robot **`virtual.fr`**, Sympa recherche le fichier **`send.intranet`** successivement dans les répertoires.

- `~sympa/expl/virtual.fr/foo/scenarii/` (uniquement pour les opérations relatives à une liste)
- ~sympa/etc/virtual.fr/scenarii/
- ~sympa/etc/scenarii/
- ~sympa/bin/etc/scenarii/

La même tactique est appliquée pour les templates (gabarits de messages et de pages html) :

- ~sympa/expl/virtual.fr/foo/ (attention par de sous répertoire templates)
- ~sympa/etc/virtual.fr/template/
- ~sympa/etc/template/
- ~sympa/bin/etc/template/

Ce dispositif permet de personnaliser complètement par liste, par robot ou pour le site l'aspect et le fonctionnement des droits d'accès de chaque page de l'interface web et du robot de messagerie.

10 Ecrire vos scénarios

Le système de scénario est un langage permettant de spécifier les droits associés à chaque opération. Presque toutes les opérations de Sympa sont sous contrôle de scénarios : création de listes, diffusion de message, accès à l'espace de dépot d'une liste, etc. La visibilité même d'une liste est contrôlée par scénario ce qui permet de dissimuler des listes à certaines catégories d'usagers. Il est donc possible de créer des vues spécifiques pour plusieurs catégories d'utilisateurs dans le but de créer des intranets.

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

Sympa est distribué avec un jeu de scénario par défaut qui sont installés dans `~sympa/bin/etc/scenarii`,
http://listes.cru.fr/sympa/distribution/current/src/etc/scenari/

> Ne jamais modifier ces fichiers, une mise à jour de Sympa écraserait votre travail. Si vous souhaitez modifier les scénarios standard, il est préférable de les copier et de les éditer dans le répertoire `~sympa/etc/scenarii`

10.1 Principe de fonctionnement des scénarios

Un scénario est une liste de règles appelées séquentiellement. Chaque règle est constituée

- d'une condition portant sur le contexte d'appel du scénario,
- de la méthode d'authentification utilisée (Entête From:, mot de passe ou authentification par certificat)
- d'une action qui sera appelée si la règle s'applique au contexte d'appel.

Exemple :

```
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
```

Cet exemple, appliqué au contrôle de la diffusion des messages, permet de limiter l'envoi des messages aux abonnés identifiés. On remarque des conditions prédéfinies comme `is_subscriber` et des variables ([`listname`] et [`sender`]) qui sont instanciées lors de

l'appel selon le contexte du moment. Trois méthodes d'authentification sont reconnues par les scénarios :

- smtp : indique que l'on se fie à l'entête `From:` d'un message.
- md5 : authentification par mot de passe (les mots de passe alloués par Sympa sont construits avec l'algorithme de hash appelé md5)
- smime : authentification forte basée sur un certificat X509 (signature S/MIME ou session HTTPS avec certificat du client)

Les actions en partie droite des règles de scénario sont des indicateurs rudimentaires. request_auth (message de retour avec demande d'authentification), do_it (rendre le service).

Si vous écrivez des scénarios, pensez à mettre un titre (`title.<lang>`), celui-ci apparaît dans les menus déroulants des formulaires de configuration de liste.

Tout scénario se termine par une règle implicite de rejet :

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

10.1.1 Les conditions utilisables dans un scénario :

- 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

Pour inverser une condition : préfixer celle-ci par un **!**

10.1.2 Les variables utilisables dans un scénario :

- [sender] : le *sender* du message courant, par extension, le demandeur de la requête courante
- [email] : dans les commandes qui acceptent un email en argument, utilisé uniquement dans le contexte unsubscribe
- [subscriber-><subscriber_key_word>] : les attributs d'abonné : email | gecos | bounce | reception | visibility | date | update_date |<additional_subscriber_fields>

Il est aussi possible d'ajouter des champs propriétaires dans la table subscriber et d'y accéder dans les scénarios et dans les templates (Voir http://listes.cru.fr/sympa/doc/sympa/node5.html#db-additional-subscriber-fields)

- [listname]
- [list-><list_key_word>] : tous les éléments du fichier de configuration d'une liste

- [conf-><conf_key_word>] : tous les éléments de la configuration de Sympa (sympa.conf)
- [msg_header-><smtp_key_word>] : tous les éléments d'entête du message (dans les scénarios `send` uniquement)

Modérer les messages contenant des attachements :

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

- [msg_body] : le corps du message

Bloquer les messages en français dans une liste francophone :

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

- [msg_part->type]
- [msg_part->body] Tout test appliqué à ces variables est un **OU** logique appliqué à chaque partie du message.
- [is_bcc] Indique que la liste cible n'est ni dans le champs `To:` ni dans le le champs `Cc:`

Une astuce anti-spam utilisée au CRU : une règle commune à tous les scénarios :

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

Cette règle est définie dans le fichier : `/home/sympa/etc/scenarii/include.send.header` (les fichiers include.<action>.header sont inclus en tête de tous les scénarios <action>.*).

- [remote_host] : la variable `REMOTE_HOST` telle que celle-ci est positionnée par le serveur HTTP. Cela suppose qu'Apache ait été configuré pour évaluer cette variable, mais cela permet de créer une notion d'intranet en particulier en utilisant cette variable dans le scénario `visibility` permettant de contrôler qui a le droit de connaître l'existence d'une liste donnée.

11 Les templates

Les compte-rendus de commandes envoyés par Sympa sont décrits dans des «templates» (exception faite de certaines commandes anciennes) ; cette séparation du code et des messages facilite les traductions et les personnalisations. Les templates sont des modèles de message et peuvent contenir des éléments de programmation (variables, conditions, boucles,...) qui seront évalués lors de l'envoi du message.

11.1 Organisation

Des templates par défaut sont fournis avec la distribution de Sympa, traduits dans 10 langues. Ils sont personnalisables à plusieurs niveaux, dans l'ordre de priorité suivant :

1. la liste (~sympa/expl/maliste/)
2. le robot virtuel (~sympa/etc/virtual.fr/etc/templates/)
3. le site (~sympa/etc/templates/)
4. la distribution (~sympa/bin/etc/templates/)

Attention : si vous personnalisez des templates, faites-le dans ~sympa/etc/templates/ (le répertoire ~sympa/bin/etc/templates lui est écrasé à chaque nouvelle installation).

Les fichiers ont l'extension.`tpl`. Si le template concerne une langue en particulier, le fichier aura l'extention `.<lang>.tpl`.

Les templates peuvent inclure des entêtes SMTP (pour définir des messages multipart notamment), auquel cas Sympa n'ajoute aucune entête SMTP au message. Un template est considéré comme MIME s'il commence par From : (voir exemple plus loin).

11.2 Le format

Les éléments syntaxiques des templates sont définis entre crochets '[]' ; leur interprétation n'est pas sensible à la casse.

11.2.1 Les variables

Exemples:

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

L'opérateur -> permet de faire référence à l'entrée d'un `HASH`.

La liste des variables utilisables pour chaque template est fournie dans la documentation : (voir les templates de site et les templates de liste).

11.2.2 Les conditions

Une structure de contrôle [IF..][ELSE][ENDIF] permet d'adapter la page en fonction du contexte lié à son usage :

Exemple:

```
[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 Les boucles

Les boucles permettent de parcourir une liste d'éléments (représentés en interne par un tableau ou un tableau associatif).

Exemple :

```
Les listes publiques :

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

```
[END]
```

`[elt->NAME]` est une variable particulière contenant la clef courante dans le tableau associatif (dans l'exemple ci-dessus, le nom de la liste). Dans le cas du parcours de tableau, la variable `[elt->INDEX]` contient l'index courant.

11.2.4 Inclusion de fichiers

L'inclusion d'un fichier peut être « passive » (INCLUDE) ou le fichier inclus peut être interprété comme un template (PARSE). Le chemin du fichier à inclure peut être fourni directement ou par l'intermédiaire d'une variable.

Exemple d'inclusions de fichiers textuels :

```
[INCLUDE 'archives/last_message']

[INCLUDE file_path]
```

Exemple d'inclusions de templates (eux-même parsés) :

```
[PARSE 'welcome.tpl']

[PARSE file_path]
```

11.2.5 Echappement

Il est possible d'interrompre l'interprétation d'un template (pour utiliser des [] par exemple). Les directives `[STOPPARSE]` et `[STARTPARSE]` permettent respectivement d'interrompre et de reprendre l'interprétation du template.

Exemple d'échappement d'une fonction JavaScript sensible :

```
<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 Liste des templates

D'une manière générale, un template est un modèle de rapport de commande. Dans un avenir proche, tous les rapports de commandes seront définis par des templates (actuellement certains utilisent les NLS).

Template	Commande associée	Description du message

bye.tpl	SIGNOFF	Message de désabonnement
global_remind.tpl	REMIND *	Rappel de l'ensemble des abonnements
helpfile.tpl	HELP	Aide sur Sympa
info_report.tpl	INFO	Informations sur une liste
invite.tpl	INVITE	Message d'invitation
list_created.tp		Notification de création de liste
list_unknown.tpl		Rapport de non-remise (liste inconnue)
listmaster_notification.tpl		Notifications pour le(s) listmaster(s)
lists.tpl	LISTS	Catalogue des listes
moderate.tpl		Notification message à modérer
modindex.tpl	MODINDEX	Liste des messages en attente de modération
reject.tpl	REJECT	Notification de rejet d'un message
remind.tpl	REMIND	Rappel d'abonnement
removed.tpl	DEL	Message de suppression
review.tpl	REVIEW	Liste des abonnés
sendpasswd.tpl		Rappel de mot de passe
stats_report.tpl	STATS	Statistiques sur la liste
summary.tpl		Message en mode "summary"
welcome.tpl	SUBSCRIBE / ADD	Message de bienvenue
x509-user-cert-missing.tpl		Notification d'abonné : pas de certificat pour chiffrer
your_infected_msg.tpl		Notification d'abonné : rejet de message contaminé

11.4 Exemples

Message de bienvenue (welcome.fr.tpl) incluant le dernier message diffusé :

```
From: abc-request@cru.fr
Subject: Bienvenue dans la liste ABC
Content-type: multipart/mixed; boundary="myboundary"

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

Bienvenue dans la liste ....

Veuillez trouver ci-dessous le dernier message diffusé dans la liste :

--myboundary
Content-Type: message/rfc822

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

--myboundary--
```

12 Support LDAP dans SYMPA

Sympa propose dans sa version 3.3.3 le support de LDAP pour les trois fonctionnalités suivantes : l'authentification dans wwsympa, la constitution de listes dynamiques et l'utilisation de filtres LDAP dans les scénarios.

Cette section du cours est une présentation succincte et pratique de LDAP, puis un guide pour la mise en œuvre de LDAP dans sympa pour les trois cas cités préalablement.
L'ensemble des possibilités de paramétrage n'est pas décrit ; pour cela, il faut se référer à la documentation de référence de sympa : http://listes.cru.fr/sympa/doc/sympa/ .

12.1 Présentation succincte de LDAP

Ce paragraphe décrit d'une manière très grossière et pratique la technologie LDAP.
LDAP est un protocole 'standard' permettant un accès aisé et rapide à des informations d'annuaire au sens large.
Les serveurs LDAP sont optimisés pour délivrer très rapidement des données en lecture ; l'écriture de données via le protocole LDAP est en général coûteux en temps. Les informations qui seront proposées par un serveur LDAP seront donc relativement stables, comme par exemple des informations concernant le personnel d'une entreprise, ses serveurs, les objets de communication proposés par ces serveurs, …
Les informations LDAP sont en général structurées d'un manière hiérarchique.

12.1.1 Structure d'une base LDAP

La structure d'un serveur LDAP est appelée **DIT** (Directory Information Tree).
Un serveur LDAP n'a pas de connaissance intrinsèque du type d'objets qu'il publie ; les différents objets sont définis par un **`schéma`** propre à un serveur LDAP.
La racine d'un serveur LDAP est appelée **`root DN`**. Un exemple de root DN pourrait être : *DC=MonEtablis,,DC=fr*. Cet exemple laisse supposer une équivalence entre dénominations de racine LDAP et DNS ; elle n'est absolument pas obligatoire (mais souhaitable pour des raisons d'unicité ?).
Un **`objet`** LDAP est identifié par son Distinguish Name (DN), qui est unique dans une base LDAP, et qui décrit son adresse dans l'arborescence.
Ainsi, le DN d'un objet LDAP situé dans le 'container' *OU=People* de la racine précédente pourrait être le suivant :

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

La partie la plus à gauche du DN (ici, *uid=tartempion*) est appelée **RDN** : Relative Distinguish Name. Elle apparaît également comme attribut de l'objet.

12.1.2 Composition d'un objet LDAP

Une entrée LDAP est adressée par son DN. Elle est composée de différents attributs, qui n'ont aucune signification pour le serveur (hormis les attributs *objectClass* et *userpassword*).
Ces attributs peuvent être multi-valués.

- L'attribut **objectClass** détermine le type d'objet manipulé ; en fait, le **schéma** d'une base LDAP décrit l'ensemble des objectClass supportés par cette base, et pour chaque objectclass, les attributs possibles et obligatoires.
- L'attribut **userPassword** : il permet d'affecter un mot de passe lié à une entrée LDAP ; ce mot de passe peut être en clair, ou crypté. Si le mot de passe est crypté, le type de cryptage est défini en faisant précéder le mot de passe du type de cryptage, encadré des caractères '{' et '}' ; par exemple, *{sha}xxxxxxx* pour un mot de passe de type sha.

Un objet LDAP comporte nécessairement un attribut *objectClass*, multi-valué. Le RDN fait partie des attributs de l'entrée LDAP. L'unicité du RDN n'est pas assurée dans la base LDAP.
La syntaxe et l'utilisation de chaque attribut est décrite dans le schéma : numérique, 'case sensitive', mono ou multi-valué, obligatoire ou non, ….

12.1.3 Groupes LDAP

Deux classes LDAP 'standard' définissent des objets de type groupe : groupOfUniqueNames et GroupOfNames.
Ces objets comportent différents attributs banalisés, et un attribut multi-valué qui contient les DN des objets LDAP faisant partie du groupe ; cet attribut normalisé a comme nom respectif *uniqueMember* ou *Member*, le premier étant plus courant.
De nombreuses interfaces LDAP d'authentification supportent ces groupes (et particulièrement le premier, *groupOfUniqueNames*).

12.1.4 Niveau de protocole

La définition ''standardisée' du protocole LDAP est actuellement en version 3. A ne pas confondre avec la version des différents serveurs LDAP (Iplanet : 5.xx, OpenLDAP : 2.xx, …).

12.1.5 Caractères accentués

Depuis la version 3 du protocole LDAP, les caractères accentués sont codés en **UTF8**.

12.1.6 Format d'échange

Un format d'échange d'objets LDAP a été normalisé : le format **LDIF** (???). Les caractères qui ne sont pas ascii purs sont encodés en format 'encode64'.

12.1.7 Droits d'accès

Des droits d'accès sont définis (ou non) au niveau du serveur LDAP afin de permettre l'accès en lecture, écriture, création, …, en fonction de l'authentification dans ce serveur.

12.1.8 Déroulement d'une requête LDAP

Une requête LDAP se déroule habituellement de la façon suivante :

- Etablissement d'une connexion TCP/IP au serveur LDAP (machine IP et port TCP).
- Etablissement d'une connexion logique (bind) auprès du serveur LDAP ; cette connexion logique est composée d'un DN, et du mot de passe correspondant.
  une connexion anonyme est possible avec un DN et un mot de passe vides.
- Génération d'une ou plusieurs requêtes, porteuses des éléments suivants :
  . le DN d'une branche de recherche, ou de la racine du serveur.
  . la portée de la recherche : **sub** (comme subtree) pour un objet situé à n'importe quel niveau de la sous-arborescence de la recherche; **one** pour un objet situé strictement au niveau de l'arborescence de recherche ; **base** pour l'objet lui-même.
  . un filtre de recherche.
- Eventuellement, la liste des attributs recherchés, et d'autres options.
- Rupture de la connexion logique (unbind)
- Rupture de la connexion TCP

Depuis la version 3 de LDAP, il est possible de générer une requête LDAP sans bind préalable ; dans ce cas, une connexion anonyme est supposée.

12.1.9 Filtres de recherche LDAP

Le filtre LDAP le plus simple est de la forme « (*NomAttribut=ValeurAttribut*) ». Ainsi, pour interroger toutes les entrées ayant l'attribut '*cn*' valué à '*toto*', le filtre LDAP sera *(cn=toto)*.
La partie droite '*ValeurAttribut*' peut comporter le caractère '*' ; il remplace toute occurrence d'un caractère quelconque.
Il est possible (et même fréquent), de générer un filtre plus élaboré, comportant des opérateurs logiques : ET (&), OU (|), non (!).
Par exemple :

(&(objectClass=inetOrgPerson)(cn=math*)(|(serv=cri)(serv=pers)))
Retourne toutes les entrées de type *inetOrgPerson*, ayant l'attribut *cn* commençant par *math* appartenant aux services *cri* ou *pers*.

12.1.10 Replica

La mise en exploitation d'un service LDAP fiable et performant nécessite bien souvent la mise en œuvre de réplicas de la base LDAP. La technique de réplica n'est pas décrite dans le protocole LDAP ; elle est donc spécifique à chaque implémentation.
Très souvent, le réplica est de type maître – esclave.
Dans le cas de l'utilisation de réplicas, il faut paramétrer les clients afin qu'ils puissent lancer les requêtes vers un serveur de backup en cas d'indisponibilité du premier serveur.

12.1.11 Paramétrage habituel d'un client LDAP

Les clients LDAP se paramètrent globalement de la même manière. Comme on l'a vu précédemment, ils ont besoin des paramètres suivants :

- HOST : le nom de la machine supportant le serveur LDAP.
- PORT : le port TCP correspondant au serveur LDAP.
- SUFFIXE : la base de la recherche
- SCOPE : la portée de la recherche
- FILTRE : le filtre de recherche
- Dans le cas où la requête nécessite un bind non anonyme, le DN et le mot de passe de bind seront indiqués dans le paramétrage.

D'autres paramètres peuvent êtres utiles en fonction de l'application.
Dans le cas d'utilisation de réplicas, la syntaxe est très souvent la suivante :
HOST ldap1.univ.fr:392,ldap2.univ.fr:389

Préciser les attributs désirés
Lorsqu'on n'a pas besoin de tous les attributs des objets recherchés, il est souhaitable de préciser la liste des attributs désirés; ceci peut limiter considérablement la quantité d'informations retournées lors des requêtes.

Préciser l'objectClass dans les requêtes
L'objectClass permet de sélectionner le type d'objet que l'on désire atteindre. Il est fréquents que le même attribut soit présent dans des objets de type différents. Ainsi, les attributs *CN* (Common Name), *description*, …

Authentifier avec LDAP
Si l'administrateur du serveur LDAP a paramétré correctement le serveur LDAP, il est impossible à tout client de lire l'attribut user password. La technique d'authentification à l'aide de LDAP est la suivante :

- connexion anonyme, afin de récupérer le DN de la personne à identifier, en fonction de l'identifiant passé par l'utilisateur :

exemple de requête : *(&(objectCLass=inetOrgPerson)(uid=IdentifiantUser))*

- Bind avec le DN de l'utilisateur, et le mot de passe saisi. Si le bind est accepté, le mot de passe est valide.

12.1.12 Exemple d'objets LDAP

Voir en annexe 1 un exemple de type personne et un exemple de type groupe de personnes

12.2 Pré-Requis pour le support de LDAP dans sympa

Le module `perl-ldap` doit être installé sur la machine qui supporte le logiciel sympa pour pouvoir utiliser les fonctionnalités LDAP de sympa.

12.3 Utilisation de LDAP pour l'authentification dans wwsympa

12.3.1 Authentification dans wwsympa

L'interface wwsympa est à géométrie variable en fonction du type d'utilisateur 'connecté'.
Les types d'utilisateurs sont les suivants :

- anonyme : utilisateur non logué dans l'interface
- abonné d'une ou plusieurs listes
- modérateur et/ou propriétaire d'une ou plusieurs listes
- listmaster

Les trois derniers cas nécessitent bien sûr une authentification dans l'interface wwsympa.
De manière native, un utilisateur s'authentifie à l'aide de son adresse mail et d'un mot de passe qui est connu de sympa (mémorisé dans la base de données sympa, de manière cryptée, mais avec possibilité de décryptage).
Le mot de passe initial de l'utilisateur est affecté par sympa ; l'utilisateur a la possibilité de le modifier ultérieurement via wwsympa. Il peut également demander à sympa de lui transmettre son mot de passe par messagerie.

12.3.2 Authentification avec LDAP

Pour bénéficier de l'authentification LDAP, il faut préalablement paramétrer sympa ; ceci se fait à l'aide du fichier auth.conf, qui se trouve dans le répertoire ~sympa/etc/.
Lorsque ce fichier est présent, et convenablement paramétré, l'authentification dans wwsympa se fait de cette manière :

- L'utilisateur saisit son adresse mail ou son 'login' LDAP, et son mot de passe.
- Si le nom d'utilisateur correspond à une adresse mail valide, sympa tente d'authentifier via son mécanisme natif décrit auparavant.
- Si l'authentification native échoue, ou si le nom d'utilisateur ne correspond pas à une adresse mail valide, wwsympa va tenter d'authentifier dans LDAP.

12.3.2.1 Fichier auth.conf

Voir la documentation de référence à
http://listes.cru.fr/sympa/doc/sympa/node8.html#SECTION00811000000000000000
Ce fichier est constitué de un ou plusieurs paragraphes qui commencent par le mot clé ***ldap***.
Chaque paragraphe est composé d'un ensemble de couples **mot-clé** <-> valeur.
Des commentaires sont possibles en commençant la ligne par le caractère #. Les lignes vides sont ignorées.

Voici un exemple de paragraphe ldap :

```
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
```

Les mot clé *host*, *suffix*, *scope* correspondent au paramétrage de base d'une requête LDAP comme exposé auparavant.
La directive *host* permet le support du mécanisme de réplica ; ainsi, on peut préciser plusieurs serveurs LDAP ; sympa tentera le bind dans le premier, puis dans le suivant si le premier est indisponible, etc…
La syntaxe est la suivante :

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

Comme indiqué précédemment, l'utilisateur peut se loguer dans l'interface wwsympa soit par un 'login' LDAP, soit en saisissant son adresse email.
Dans le premier cas, le filtre utilisé pour la recherche sera celui précisé par la directive *get_dn_by_uid_filter*, et dans le second, par la directive *get_dn_by_email_filter*.
A noter que wwsympa remplacera les occurrences de "[sender]" par le login entré par l'utilisateur dans l'interface wwsympa.
Grâce à l'un ou l'autre filtre, l'entrée LDAP correspondant à l'utilisateur est chargée. wwsympa va ensuite contrôler le mot de passe grâce à un bind LDAP avec le DN de cette entrée, et le mot de passe saisi.
Si ce bind est accepté, l'utilisateur est authentifié. Il reste maintenant à wwsympa à faire le lien entre cet utilisateur 'LDAP' et un utilisateur sympa, connu par son adresse mail.
C'est le rôle des directives email_attribute et alternative_email_attribute.
email_attribute indique le nom de l'attribut LDAP qui contient l'adresse mail canonique de la personne.
alternative_email_attribute est une directive facultative, qui contient le nom d'une liste de noms d'attributs qui seraient utilisés dans l'annuaire LDAP pour des alias de messagerie.
Dans le cas de l'exemple ci-dessus, on peut remarquer que l'attribut *mail* contient l'adresse de mail canonique, et que l'attribut *maildrop* contient éventuellement une adresse mail alternative (alias).

12.3.2.2 Authentification vers plusieurs serveurs LDAP

Il peut y avoir plusieurs paragraphe *LDAP* dans le fichier auth.conf Dans ce cas, wwsympa tente l'authentification LDAP d'abord vers le serveur qui est décrit dans le premier paragraphe LDAP, puis vers le second si la tentative est infructueuse, et ainsi de suite.
Il serait par exemple possible, pour une université qui dispose d'un serveur LDAP du personnel et un serveur des étudiants, de permettre une authentification vers ces 2 serveurs distincts.

12.3.3 Changements dans wwsympa suite à une authentification de type LDAP

wwsympa garde la trace du type d'authentification à l'aide d'un cookie. Certains menus proposés par wwsympa sont modifiés dans le cas d'une authentification LDAP.
En particulier, dans les préférences utilisateur, le changement de mot de passe n'est plus proposé.
En outre, si l'utilisateur dispose d'adresses mail alternative dans l'annuaire LDAP, sympa propose de les mémoriser dans ses tables internes.

Utilisation Intranet

A l'université de Nancy2, l'authentification pour l'accès aux différentes ressources informatiques (intranet, messagerie, …) s'appuie sur l'annuaire LDAP, et sur l'attribut *uid* qui contient le login de l'utilisateur. Il est donc important que les utilisateurs utilisent la même technique d'authentification pour l'accès à wwsympa.
Sympa est utilisé pour des listes internes, et également pour des listes totalement ou partiellement externes.
Nous voulions donc que wwsympa puisse toujours authentifier avec son mécanisme natif pour les personnes externes, mais nécessairement avec ldap et le login pour les utilisateurs internes.
Ceci a été réalisé d'une manière simple grâce à la souplesse des templates sympa : un petit bout de code javascript a été ajouté au fichier de template (loginbanner.tpl) qui gère le login des utilisateurs.
Si ce login est une adresse de messagerie, et s'il se termine par 'univ-nancy2.fr', un message d'avertissement apparaît et demande à l'utilisateur d'entrer son 'login LDAP'.

12.4 Listes issues de requêtes LDAP

Sympa sait gérer différents types de listes : les listes 'natives', qui sont mémorisées dans sa base de données, et des listes de type 'include'.
Ceci correspond à la directive *user_data_source* du fichier de configuration de la liste, qui peut prendre respectivement les valeurs `database` ou `include`.
Les listes de type include sont générées à l'aide de requêtes auprès de serveurs SQL ou LDAP.
Il existe deux types de définition de listes LDAP : *ldap_query* et *ldap_2level_query*.
Voir la documentation de référence :

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

12.4.1 Listes de type 'ldap_query'

Ces listes sont générées dynamiquement à l'aide d'une simple requête LDAP. Elles sont définies dans le fichier de configuration de la liste grâce au paragraphe `include_ldap_query`

En voici un exemple :

```
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
```

On reconnaît le paramétrage habituel d'une requête LDAP. Ici, on désire construire une liste des personnels enseignants de l'UFR droit, ayant l'attribut mail valué. La directive *attrs* donne le nom de l'attribut contenant l'adresse mail.
Dans le cas où une requête non anonyme est nécessaire, il est éventuellement possible de préciser le DN (*user*) et le mot de passe (*passwd*) de connexion. La directive *select* permet d'indiquer à sympa comment agir dans le cas où l'attribut contenant l'adresse mail serait multivalué.
La liste générée sera donc constituée de toutes les personnes répondant à la requête saisie.

12.4.2 Listes de type 'ldap_2level_query'

Ces listes sont générées dynamiquement à l'aide de requêtes LDAP complexes, à deux niveaux. Une première requête permet de récupérer une liste de valeurs ; ensuite, pour chaque valeur récupérée, une nouvelle requête permettra de récupérer les adresses mail correspondantes.
Ce type de listes permet en particulier de traiter les groupes LDAP (*groupOfNames* ou *groupOfUniqueNames*).

Le paragraphe **include_ldap_2level_query** permet de définir une telle liste. En voici un exemple, qui traite des groupes de 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
```

Ici, la première requête permet de construire la liste des DN des personnes qui sont dans les groupes LDAP *cri* ou *ufrmi* ; la seconde récupère l'attribut mail correspondant.
La valeur *[attrs1]* peut être utilisée dans les directives *suffix2* et/ou *filter2* ; sympa la remplacera d'une manière itérative par toutes les valeurs retournées par la première requête.
Lorsque *[attrs1]* est utilisé pour la directive *suffix2*, l'attribut retourné par la première requête doit être obligatoirement un DN. Il est ainsi logique de positionner l'attribut *scope2* à la valeur *base*.
A noter qu'il est possible de faire des traitements très complexes, avec l'utilisation de la directive *regex1* (ou *regex2*) qui permettent d'appliquer un filtre de type 'expression régulière' au résultat de chaque requête ; il faut pour cela que *select1* (ou *select2*) ait la valeur *regex*.
A noter également que les listes de ce type peuvent générer une charge importante : si la liste comporte n membres, il y aura n + 1 requêtes LDAP pour la constituer.

Sympa permet de définir plusieurs paragraphes de type include pour une liste. Il est ainsi possible de constituer une liste issue de plusieurs requêtes LDAP et/ou SQL.

12.4.3 Listes include et cache sympa

Afin d'améliorer les performances de sympa et wwsympa et de limiter la charge vers les servers SQL ou LDAP avec les listes include, sympa gère un cache de ces listes dynamiques.

Ainsi, ces listes ne sont par regénérées à chaque fois que sympa ou wwsympa en a besoin, mais conservées dans un cache interne. La fréquence de rafraîchissement de ce cache est paramétrée dans le fichier de configuration de la liste, grâce à la directive *ttl*.

12.4.4 ttl et cache de listes

Sympa rafraîchit les listes lors d'une référence à celle-ci et son le `ttl` est dépassé. Wwsympa ne fait pas ce travail qui peut induire de très long temps de réponse si les listes dynamiques

sont nombreuses et si les sources externes sont lentes. En conséquence wwsympa partage avec sympa.pl un fichier « .db » dont sympa assure la mise à jour.
Nous prévoyons remplacer ce cache partagé via un fichier par un cache en base de données accessibles à tout les processus sympa.pl, wwsympa.fcgi, task_manager.pl, bounce.pl et permettant d'obtenir les mêmes performances pour les listes en mode include que pour les listes en mode database. Il sera alors possible de gérer les bounces et de positionner les options d'abonnement pour toutes les listes. La mise à jour du cache de listes en base de données sera confiée au gestionnaire de taches (task_manager.pl)

12.5 Utilisation de 'filtres ldap' dans les scénarios

12.5.1 Les Filtres Nommés :Named Filters (NF)

Ces filtres apportent le support de LDAP an niveau des scénarios. Un filtre va définir une requête LDAP.
Il sera utilisable dans le champ 'condition' d'un scénario et permettra donc d'indiquer une liste de personnes autorisées à effectuer une action, à l'aide de cette requête.

Les Filtres Nommés (on utilisera l'abréviation **NF**) sont des fichiers ascii écrits dans le répertoire ~sympa/etc/search_filters/.
Ces fichiers ont nécessairement pour extension '.*ldap*'.

Voici un exemple de contenu du NF "*EnsLettres.ldap*" :

```
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])))
```

C'est un paramétrage 'banal' d'une requête LDAP. Sympa va remplacer toutes les occurrences de '*[sender]*' par le contenu de '*[sender]*' des scénarios utilisant ce filtre, en général, l'adresse mail de l'utilisateur.
Dans cet exemple, le filtre va donner certains droits à l'utilisateur ayant un attribut *mail* valué, de type *E* (Enseignant ?) appartenant au campus Lettres et dont l'attribut *mail* ou *maildrop* correspond à son adresse mail.

12.5.2 Utilisation des Filtres Nommés (NF) dans un scénario

Les NF sont utilisables dans le champ condition d'un scénario.
La syntaxe est : `search(`*`NomDuFiltre`*`)`.

Voici un exemple de scénario utilisé pour les personnes ayant le droit d'envoyer un mail dans une liste ; il est appelé 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
```

Ici, tous les abonnés de la liste *listname* et les enseignants du campus lettres ont le droit d'envoyer un mail à la liste *listname*.

12.5.3 Etendue de ces filtre

Les filtres nommés associés aux scénarios Sympa offrent des possibilités de personnalisation très poussés.
Ainsi, les différents rôles gérés par Sympa (listmaster, owner, moderator, subscriber) pourraient éventuellement et au cas par cas être issus de requêtes LDAP.

Cache des filtres nommés
Sympa gère un cache des filtres nommés. Lors de la mise au point, il est nécessaire d'arrêter/relancer sympa pour tester les modifications apportées à un filtre.

12.6 Conclusion Sympa et LDAP

Sympa offre nativement des possibilités de personnalisation très puissantes. Le support LDAP est bien intégré à différents niveaux, et profite donc de ces possibilités.
Quelques améliorations sont encore souhaitables, en particulier du côté de la gestion des listes dynamiques et de leur rafraîchissement ; ceci est prévu dans les prochaines versions de Sympa.
D'autre évolutions liées à LDAP semblent être prévues ; il suffit de consulter les sources pour s'en convaincre.

13 Configuration MySQL

Sympa requiert les services d'une base de données pour la gestion des abonnés (au choix MySQL, PostgreSQL, Oracle ou Sybase). Nous présenterons ici les étapes de configuration avec MySQL en particulier car nous l'utilisons sur notre plate-forme de développement.
Pour la suite, nous supposons que vous disposez d'un serveur MySQL.

13.1 Installer les modules Perl

Sympa accède à sa base de données via DBI, l'interface standard de Perl pour l'accès aux bases de données. Vous devez installer le module DBI, non fourni avec la distribution standard de Perl. DBI requiert un driver (DBD) pour chaque type de base de données.

Modules Perl à installer :

- `DBI` >= 1.06
- `DBD::mysql` >= 2.407 (fourni avec Msql-Mysql-modules)

L'installation de ces modules est prise en charge lors de l'installation de Sympa. Vous pouvez également les obtenir :

- Sous forme de tar.gz sur le site CPAN le plus proche (http://cpan.cict.fr/)
- Sous forme de RPM sur RPMFind (http://www.rpmfind.net/linux/RPM/Development_Languages_Perl.html)

Si vous installez le module DBD::mysql sous forme de RPM, il requiert l'installation du RPM mysql-devel, fournissant les librairies d'accès à MySQL.

13.2 Créer la base de données

La base de données de Sympa est constituée de 2 tables :

- user_table : les préférences utilisateurs
- subscriber_table : les données relatives à l'abonnement

Le script de création de la base (ci-dessous) est fourni avec la distribution (src/etc/script/create_db.mysql). Utilisez le client en ligne `mysql` pour exécuter le 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)
);
```

Une fois la base créée, vous pouvez définir les droits d'accès à la base `sympa` pour l'usager `sympa` :

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

13.3 Configurer sympa.conf

Les paramètres d'accès à la base de données sont définis comme suit dans `sympa.conf` :

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

Contrairement aux clients en ligne (mysql, mysqladmin), les librairies C et Perl (DBI) n'exploitent pas par défaut le fichier de configuration my.cnf. Pour prendre en compte votre my.cnf, ajoutez dans sympa.conf : db_options mysql_read_default_file=/etc/my.cnf

Le paramètre de liste `user_data_source` définit le type de source de données pour les abonnés. Sa valeur par défaut est `database` (utilisation de la base de données décrite dans `sympa.conf`). Les autres valeurs possibles sont `include` et `file`.

13.4 Outils d'accès à la base de données

13.4.1 mysql

`mysql` est le client en ligne de commande fourni avec MySQL. Il permet d'exécuter des requêtes SQL. Il implémente les commodités du shell Bash : complétion, historique.

Exemple : liste des abonnés de la liste 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` permet de gérer vos bases de données MySQL via une interface web. Il permet facilement (sans connaissance SQL) de :

- créer/modifier/supprimer des bases/tables/champs
- rechercher/insérer/éditer/supprimer des enregistrements

14 Optimisations / répartition de charge

14.1 Optimisation de la diffusion

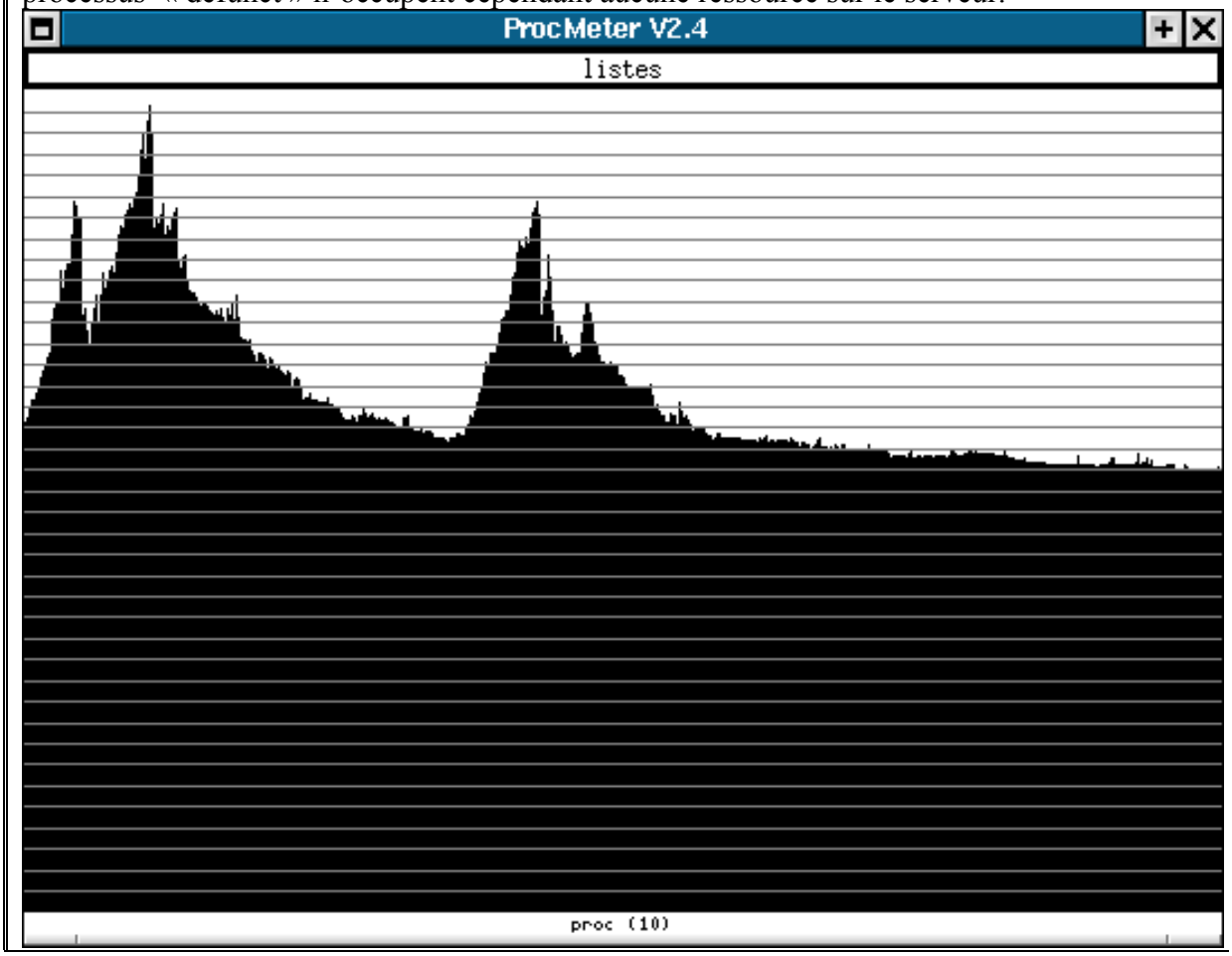
Sympa n'inclut pas de client SMTP, il utilise les services du MTA local (sendmail, postfix, exim ou qmail) via un appel système (les destinataires sont passés en paramètre, le message envoyé dans le STDIN). Plusieurs paramètres de sympa.conf définissent l'interfaçage avec le MTA :

- `sendmail`, `sendmail_args` : programme (compatible sendmail) à utiliser pour la diffusion des messages.
- `maxsmtp` (défaut 20) : nombre maximum de processus sendmail fils lancés par Sympa.
- `nrcpt` (défaut 25): nombre max. de destinataires par appel à sendmail ; définit le facteur de groupage.
- `avg` (défaut 10): nombre maximum de domaines internet différents par appel à sendmail.

Le facteur de groupage (paramètres `nrcpt` et `avg`) est à paramètré en fonction de plusieurs critères :

- le MTA utilisé : les MTAs ont un comportement très variable vis à vis du groupage. Sendmail évalue le groupage sur la base des seuls destinataires passés en paramètre, mais il peut envoyer une seule fois le corps du message pour plusieurs destinataires. Postfix tente une remise vers les MTA adjacents en conservant le groupage des destinataire (plusieurs RCPT TO : pour un seul DATA). Qmail n'effectue aucun groupage.
- la taille des listes : dans une petite liste, il y a peu d'abonnés du même domaine, on atteint rapidement `avg` destinataires par paquet. Dans une grande liste, avec plus d'abonnés par domaines, c'est le paramètre `nrcpt` qui bornera le groupage.
- la politique anti-spam des sites distants : certains ISP (Yahoo au moins) rejettent certaines sessions SMTP dont le nombre de destinataires est jugé suspect. Veillez donc à maintenir le paramètre `nrcpt` en-dessous d'un seuil raisonnable.

Ce diagramme représente l'évolution du nombre de processus sur un serveur Sympa. Les processus fils de Sympa, une fois leur tache terminée, ne sont libérés par sympa.pl que lorsque ce dernier atteint son quota (maxsmtp) ou une fois le traitement du message courant terminé. Cela conduit à des libérations massives de processus (en dents de scie). Les processus « defunct » n'occupent cependant aucune ressource sur le serveur.



14.2 Optimisation avec sendmail

Deux objectifs concurrents peuvent être recherchés :

1. augmenter la réactivité du service en diminuant le temps de traitement d'une diffusion : on augmente le `maxsmtp` quitte à charger la machine.
2. diminuer la charge de la machine (baisser `maxsmtp`) par exemple pour conserver pendant des distributions copieuses de bons temps de réponse sur l'interface www; quitte à ce que celle-ci dure plus longtemps.

On obtient un bon compromis en optimisant les processus fils de Sympa.

14.2.1 Non canonisation des adresses

Dans Sympa, l'appel à Sendmail est bloquant jusqu'à ce que sendmail obtienne un éventuel CNAME du DNS pour chacun des destinataires passés en argument. Cette résolution n'est pas nécessaire ; par ailleurs si le DNS est peu disponible, le serveur de listes est paralysé.

Nous mettons cette fonctionnalité en place uniquement pour les appels sortant via Sympa en utilisant un sendmail.cf spécifique à Sympa. Dans cette configuration, le nombre de processus sendmail en mémoire augmente beaucoup plus vite.

Voir FEATURE « nocanonify ».

14.2.2 Abaisser les timers

On peut optimiser certains timer des sessions SMTP dont les valeurs par défaut colle assez mal avec l'état lamentable de la disponibilité de certains mailhosts. En effet ces timers qui s'expriment en minutes voir en 10zaines de minutes. Il suffit alors d'une douzaines de serveurs saturés parmis vos gros correspondants pour que les sessions SMTP disponibles soit alors constituées majoritairement (voir exclusivement) de sessions en attente de réponse du serveur distant. Ces ajustement vise à faire baisser rapidement le nombre de sendmail en machine pour traiter d'autres messages. Bien entendu, il faut régler la gestion des spools en conséquence.

Le timer `iconnect` est le plus intéressant, nous avons positionné :

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

Voir les options de configuration.

14.3 Optimisation MySQL

14.3.1 Structure de la base

Depuis la version 3.23 de MySQL, les tables sont stoquées par défaut au format MyISAM, offrant de meilleures performances que son prédécesseur ISAM. Pour passer d'anciennes tables au format MyISAM :

```
#/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
```

Le commande `myisamchk` permet d'effectuer des opération de maintenance sur les tables de la base de données. Exemple : augmentation de la taille du buffer pour le tri

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

La requête OPTIMIZE TABLE permet également d'optimiser une table MySQL en libérant l'espace non utilisé et en défragmentant les fichiers de données.

```
# /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 du serveur

Au démarrage, le démon MySQL lit les valeurs de ses options définies dans le fichier de configuration `/etc/my.cnf`. Pour obtenir la liste des options au « runtime » :

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

Des exemples de fichiers de configuration adaptés à différents usages sont fournis avec la distribution MySQL (`my-huge.cnf`, `my-large.cnf`, `my-medium.cnf`, `my-small.cnf`), chez nous sous /usr/share/mysql/. Vous pouvez les prendre comme base pour optimiser votre installation.

Par défaut le serveur MySQL ne logue pas les opérations effectuées sur les bases. Pour le faire loguer, ajoutez l'entrée suivante à votre /etc/my.cnf :

log = /var/log/mysql

14.4 Répartition de charge

Il peut être intéressant de répartir le service de listes sur plusieurs serveurs. La solution la plus simple consiste à déporter les applications extérieures à Sympa :

- la base de données (MySQL, PostgreSQL,…) : voir le paramètre `db_host` de sympa.conf.
- le serveur de messagerie (sendmail , postfix, …) : vous pouvez transmettre le trafique sortant du serveur de listes à un autre serveur (ou plusieurs en faisant du Round Robin DNS).

Voir l'expérience d'utilisation de mini_sendmail

Vous pouvez également séparer les composantes de Sympa, à savoir :

- traitement des mails : `sympa.pl`
- interface web : `wwsympa.fcgi`
- gestion des rapports de non-remise : `bounced.pl`
- gestion des archives web : `archived.pl`

Mais la séparation de ces applications imposent de partager les données entre les serveurs :

- base de données
  - ⇨ toutes les applications doivent y avoir accès
- fichiers de configuration (sympa.conf & wwsympa.conf)
  - ⇨ NFS
- ~sympa/etc/
  - ⇨ NFS
- ~sympa/spool/
  - ⇨ NFS
- ~sympa/expl/
  - ⇨ NFS
- les rapports de non-remise (utilisés par bounced.pl et wwsympa.fcgi)
- les archives web (utilisées par archived.pl et wwsympa.fcgi)

15 L'installation de Sympa

L'installation de Sympa est simple. Après avoir téléchargé la version courante à partir su site http://www.sympa.org, et avoir créé un usager dédié `sympa.sympa`, extraire les fichiers du tar, puis procéder à l'installation :

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

Bien entendu, ceci n'est pas suffisant. Il faut aussi installer ou configurer les produits annexes tel que l'antivirus, le serveur de messagerie, la base de données et le serveur http. Vous trouverez dans le manuel de référence de Sympa les indications pour réaliser ces configurations. Un certains nombres de choix devront être fait au moment de l'installation ou plus tard lors de l'évolution du service :

- Choix de l'agent de messagerie (sendmail, exim, easy-sendmail, postfix, qmail)
- Choix d'un SGBD (MySQL, Sybase, Oracle, PosgreSQL)
- Choix d'un antivirus (MacAfee/Uvscan, Fsecure/fsav, Sophos, AVP, Trend Micro/VirusWall)
- Choix d'un serveur http (Apache, Roxen)
- Choix d'une installation binaire (.rpm, .deb) ou source.
- Un seul serveur ou répartition sur plusieurs machines

16 Intégration de Sympa avec d'autres applications

16.1 Partage de l'authentification web

Sympa vous permet d'unifier votre système d'authentification avec vos autres applications web. Il utilise un cookie HTTP pour transporter les données d'authentification ; ce cookie ne contient cependant aucune notion de privilège.

16.1.1 Utiliser le système d'authentification de Sympa

Dans ce cas de figure, vous réutilisez les fonctions de login, de logout et de rappel de mots de passe de Sympa. Vous pouvez intégrer le formulaire de login dans votre application, le CGI appelé restant wwsympa.fcgi. A votre charge d'exploiter le cookie HTTP d'authentification de Sympa (vérification, extraction de l'adresse email).

Pour que wwsympa.fcgi redirige l'utilisateur de façon transparente vers votre application, ajouter `/referer` à la fin de l'URL d'appel à wwsympa.
Exemple :
<A HREF="/wws/loginrequest/referer">Page de Login</A>

16.1.2 Sympa reconnaît votre authentification

Votre application gère le système d'authentification, mais positionne un cookie Sympa pour être reconnue de lui. L'application doit partager le « secret » de Sympa (paramètre `cookie` de sympa.conf) utilisé pour généré le cookie. Dans ce cas, Sympa reconnaîtra vos utilisateur comme authentifiés. Exemple d'intégration de ce type : Rearsite.

16.2 Partage des données

16.2.1 Définition des listes par extraction d'une base de données

Par défaut les membres d'une liste sont des abonnés, gérés dans la base de données dédiée à Sympa. Les membres de la liste peuvent être le résultat d'une requête dans une base de données (ou un annuaire LDAP). Dans ce cas, la liste des membres est mise à jour régulièrement (paramètre de liste `ttl`) par intérogation de la base. Plusieurs sources de données peuvent être définis pour une même liste.

Extrait de config de liste :

```
# 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
```

Cache des données : actuellement le cache des données est stocké dans un fichier DB. De prochains développements devraient permettre de gérer le cache dans la base de données (MySQL), ce qui aurait pour avantages de :
- permettre de définir des listes mixtes include/abonnement

- rendre disponibles les options d'abonnement à tous (y compris extraction DB)
- diminuer la taille des processus Sympa, les données étant gérées par MySQL

16.2.2 Ajouter vos données à la base de données de Sympa

Vous avez la possibilité d'étendre la structure des tables `user_table` et `subscriber_table` pour y ajouter des champs pour vos usages propres (exemple : système de paiement pour les listes). Sympa préservera ces champs et les rendra même disponibles dans les templates et les scénarios. Les champs supplémentaires doivent être déclarés dans sympa.conf.

Les champs ajoutés à la table `subscriber_table` sont visibles depuis l'interface web d'administration des abonnés.

Extrait de 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

Le mot de passe n'est pas la seule façon de s'authentifier. L'émergence des infrastructures de gestion de clefs (IGC) permet de mettre en œuvre des méthodes à la fois plus sûres et plus pratiques. Sympa utilise une librairie de chiffrement (OpenSSL) permettant d'exploiter les certificats X509 dans toute l'application. Cette section a pour objet de vous aider à mettre en œuvre ces fonctionnalités rapidement. Pour en savoir plus on se reportera au manuel de référence de Sympa et à l'article « Sympa S/MIME and Sympa mailing lists manager » http://listes.cru.fr/sympa/documentation/article_smime/sympasmime.html .

17.1 HTTPS

Comme chacun le sait on distingue deux modes de fonctionnement de HTTPS :

- Le mode chiffrement sans authentification des clients. Dans ce cas on n'utilise pas de certificat client ; le seul certificat X509 en jeux étant celui du serveur. Ce type d'installation est particulièrement simple à réaliser, mais le service rendu par la couche SSL se limite au chiffrement des échanges qui ne peuvent être interceptés par quiconque. Dans le cas de Sympa, les données sensibles sont bien entendu les mots de passe et les cookies. Une telle installation consiste à :
  - Installer OpenSSL et mod_ssl pour Apache
  - Installer un certificat serveur

  Se référer à la très bonne documentation de mod_ssl (http://www.modssl.org) pour ces opérations. La seule modification de la configuration de sympa consiste à mettre à jour la variable `wws_url` de `sympa.conf` (https au lieu de http).

- Le mode chiffrement avec authentification des clients. Dans ce cas, les clients présentent (automatiquement) leur certificat. Sympa a été modifié pour exploiter les données extraites du certificat client ce qui permet de supprimer la phase d'authentification. Le bouton login de sympa ne s'affiche plus et toutes les pages sont affichées avec les

privilèges de la personne identifiée par l'adresse de messagerie présente dans son certificat avec la méthode d'authentification `smime`. Ce résultat est obtenu en combinant les deux éléments de configuration suivant :

- Le serveur Apache est configuré pour demander un certificat au client : `SSLVerifyClient Optional` ou `SSLVerifyClient Require`
- L'option de mod_ssl StdEnvVar est positionnée dans Apache pour permettre à ce cgi d'hériter dans des variables d'environnement extraites des certificats : `SSLOptions +StdEnvVars`

17.2 S/MIME

Sympa peut aussi exploiter les certificats X509 à travers des messages signés et/ou chiffrés reçus et/ou émis par lui. Le but est alors d'authentifier le « sender » d'un message en se basant sur la signature S/MIME du message ou de diffuser des messages chiffrés.

17.2.1 Validation des signatures S/MIME

Openssl étant installé, il suffit de renseigner les paramètres `openssl` et `trusted_ca_options` de `sympa.conf`. On a bien entendu intérêt d'assurer la cohérence avec les autorités de confiance dans le contexte d'Apache en partageant les mêmes fichiers de certificats au format PEM :

Exemple extrait de la configuration du serveur de listes du CRU :

```
##  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 peut alors valider les signatures de message S/MIME.

17.2.2 Diffusion chiffrée

La diffusion chiffrée de messages dans une liste est possible à condition qu'un certificat X509 et la clef privée associée aient été installés pour la liste. Cette installation peut être réalisée avec le script ~sympa/bin/`p12topem.pl` à partir d'un certificat et de sa clef au format `pkcs#12`.

Le message de bienvenue de la liste est alors signé (en utilisant S/MIME) ce qui permet aux interfaces de messageries des abonnés de stocker ce certificat. Ils peuvent alors poster des messages chiffrés à destination de la liste comme vers tout autre correspondant.

Sympa ne peut chiffrer un message à l'intention d'un abonné que s'il a dans son cache le certificat de celui-ci. Il faut donc qu'il ait reçu au moins un message signé de chaque abonné des listes pour lesquelles le chiffrement est mis en œuvre. En effet, Sympa conserve dans le répertoire `~sympa/expl/X509-user-certs` les certificats extraits de tout les messages signés reçus.
Il est donc conseillé d'imposer l'abonnement via un message signé S/MIME pour ces listes (utilisation de la méthode d'authentification `smime` dans le scénario `subscribe`).

18 Les ressources disponibles

Cet article est disponible sur le site www.sympa.org dans la rubrique « documentation ». Vous y trouverez également la documentation de référence aux formats HTML, Postscript et PDF. Une foire aux questions répond aux questions les plus souvent posées dans les listes de diffusion. Enfin les listes sur Sympa (sympa-fr, sympa-users, sympa-dev, sympa-announce et sympa-translation) vous sont ouvertes. Nous rappelons que des archives web de ces listes sont disponibles.

Plusieurs entreprises fournissant un support commercial sur Sympa sont référencées dans la rubrique « technical support » du site web.

Le code de Sympa est géré avec CVS (Concurrent Versions System). Le serveur CVS et sa version web vous donnent un accès à la version de Sympa en cours de développement. D'une manière générale, il n'est pas raisonnable d'installer chez vous la branche courante de l'arbre CVS qui n'a peut-être par encore été validé en exploitation au CRU. L'accès web à CVS peut vous être utile pour connaître l'état d'avancement d'un module, rechercher un bugfix, récupérer un patch.