AIC NETWORK™ L'Opérateur Télécom des Entreprises	**CAHIER DES CHARGES** FONCTIONNEL DETAILLE

Réf. :	SMSROUTE V1	
Projet :	Création du site SMSROUTE et développement de l'API interne	
Émetteur :	RAVOUNA Jérémy Chef de projet AIC NETWORK	Tél : 0825 157 802 Mail : jravouna@avanim-prod.com
Date d'émission :	le 21/02/2008	

Validation			
Nom	Date	Validation (O/N)	Commentaires

Historique des modifications			
Version	Date	État	Description de la modification
1	18/02/2008	A VALIDER	Première release
2	21/02/2008	VALIDE	Première release OK

Accès au serveur de développement			
Adresse	Login	Mot de passe	Description

AVANT PROPOS

- Le présent cahier des charges fonctionnel détail les fonctions nécessaires à l'envoi de SMS via une interface d'application (API) fournit par un prestataire téléphonique que nous appellerons fréquemment « fournisseur ».
  Le fournisseur dispose d'une API Web et FTP.

- Le présent cahier des charges détail les fonctions nécessaires au fonctionnement de la solution Web et de l'API à mettre en place. Dans un but évident de rapidité et de fiabilité de développement il est possible que certains modules soient développés seuls.

- Le site Internet contenant cette plate forme s'intitulera SMSROUTE et sera basé sur Joomla CMS 1.5. Voilà pourquoi il nous paraît important de souligner que le développement de l'application SMSROUTE et de son API client pourront et devront tenir compte du framework et de la base de donnée Joomla 1.5. (ex : la table client et users seront des extensions de jos_user de Joomla).

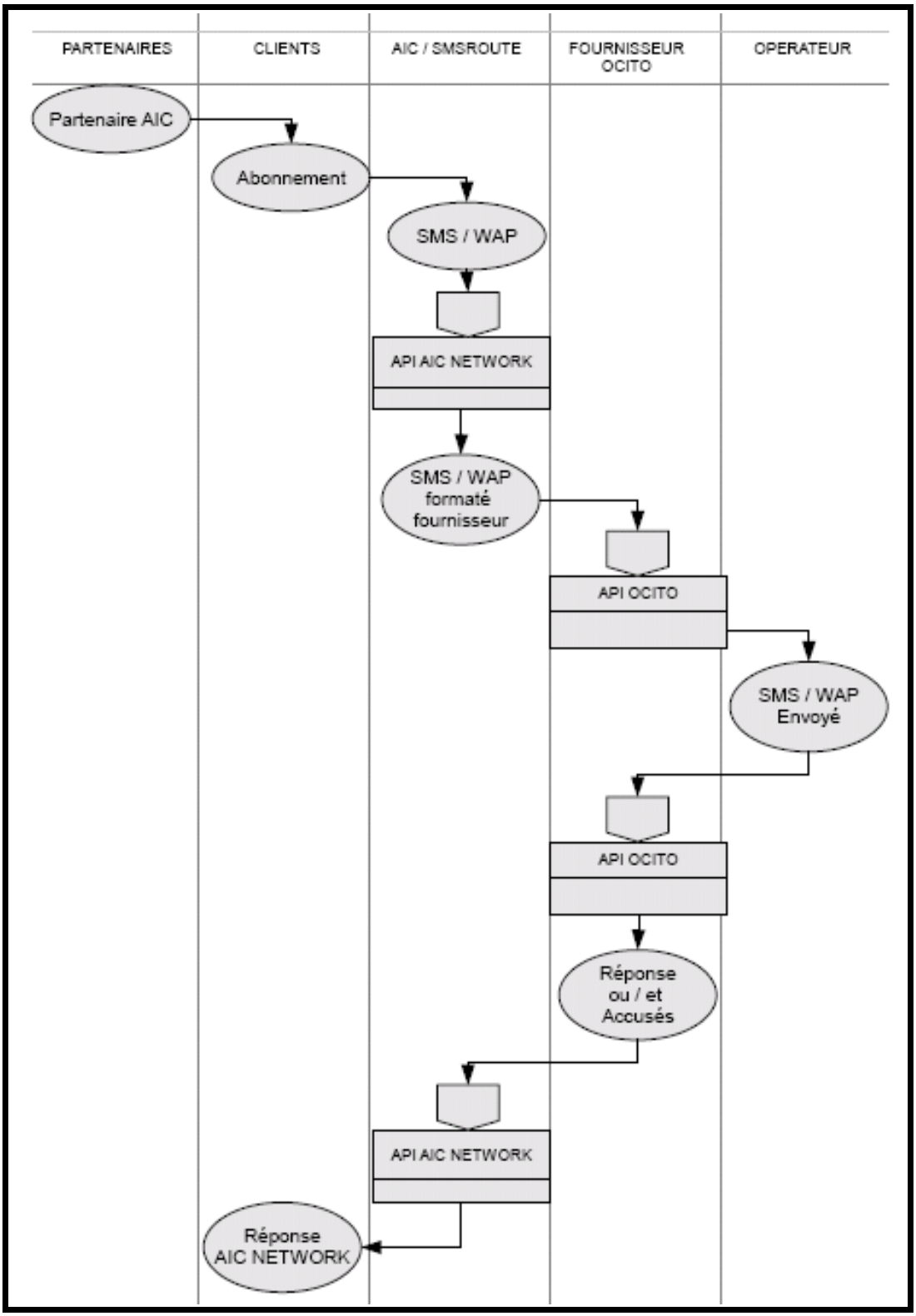
- Afin de permettre de futures évolutions telles que la connexion à distance par DLL, nous utiliserons une gestion de session via le framework Joomla 1,5 (voir la classe Jsession).

- Le ou les prestataires retenu devront être à même de développer l'application mais également d'analyser les modèles de traitement et d'informer la société AIC de modifications nécessaire sur les modèles. (MLD, UML ou MOT).

- Nous conseillons de mémoriser les documentations API du fournisseur (HTTP et FTP) ainsi que les modèles de données et de traitements fournis par AIC NETWORK.

SOMMAIRE :

- Schéma basique de l'applicatif :



1 Présentation du projet

- AIC NETWORK, société de télécommunication et opérateur téléphonique pour les entreprises, désire mettre en place une solution d'envoi de sms via Internet pour ses clients.

- Présent sur le marché depuis 1997, AIC NETWORK fournit tous les produits et services liés aux télécommunications commercialisés par un réseau de partenaires. Téléphonie Voip, Serveur et Hébergement, Accès internet et Messagerie.

2 Architecture fonctionnelle

2.1 Architecture

- Aujourd'hui AIC network dispose d'une plate forme « SMSROUTE » mais désire déployer ce service en utilisant l'API de son partenaire. En effet cette API fournisseur permet un envoi en masse de SMS et MAIL WAP via un fichier CSV pré formaté via une API FTP, ainsi que l'envoi événementiel de SMS via son API HTTP.

- Il s'agit, pour ce projet, de développer une api professionnelle interne à la société AIC NETWORK en relation avec l'API fournisseur. Mais également de développer un site Internet interfacé avec la solution open source Joomla 1.5, permettant l'utilisation des API SMS ROUTE par un client.

- Le système devra être suffisamment souple pour pouvoir accueillir des fournisseurs téléphoniques supplémentaires avec les formules adéquates. C'est dans cette optique que le modèle de données joint propose une table catalogue.

- L'opérateur fournissant son API recevra des envois de sms en HTTP et en FTP, mais émettra également des réponses de réception, des accusés d'acquittement des messages ainsi que des accusés de réception. (Voir la 'reference API' en annexe).

- Les offres AIC NETWORK devront s'adapter au besoin du client. AIC pourra ainsi recevoir des SMS sous forme de mail (SMTP), d'appel de son api web (HTTP) ou bien pour un mailing de masse, par fichier uploadé (FTP). Cette dernière solution sera mise en place de façon asynchrone. (CF modèle de traitement)

- Ce développement représente une base pour d'autre projet à plus ou moyen court terme, tel que la création d'une DLL avec activation. Ainsi nous insistons sur l'absolue nécessité d'obtenir une application évolutive.

2. Spécificités techniques

Langages dominants : (X)HTML, JS(AJAX), PHP 5, SQL
Base de données : MYSQL 5
Serveurs : Apache2, Proftpd (extension mysql)
Spécificités : Développement en POO.

3 Les fonctions

3.1 Modèle détaillé des fonctions

- Liste des fonctions attendues :

Coté AIC NETWORK :

- Permettre l'inscription et l'authentification des clients
- Facturer et régler en ligne via smsroute.com
- Proposer un catalogue client complet
- Recevoir des messages en HTTP (api web) de clients
- Recevoir des messages en FTP (fichiers csv) de clients
- Recevoir des messages en SMTP (mail formatés) de clients
- Réceptionner les confirmations de l'API fournisseur
- Réceptionner les accusés d'acquittement et de réception de l'API fournisseur

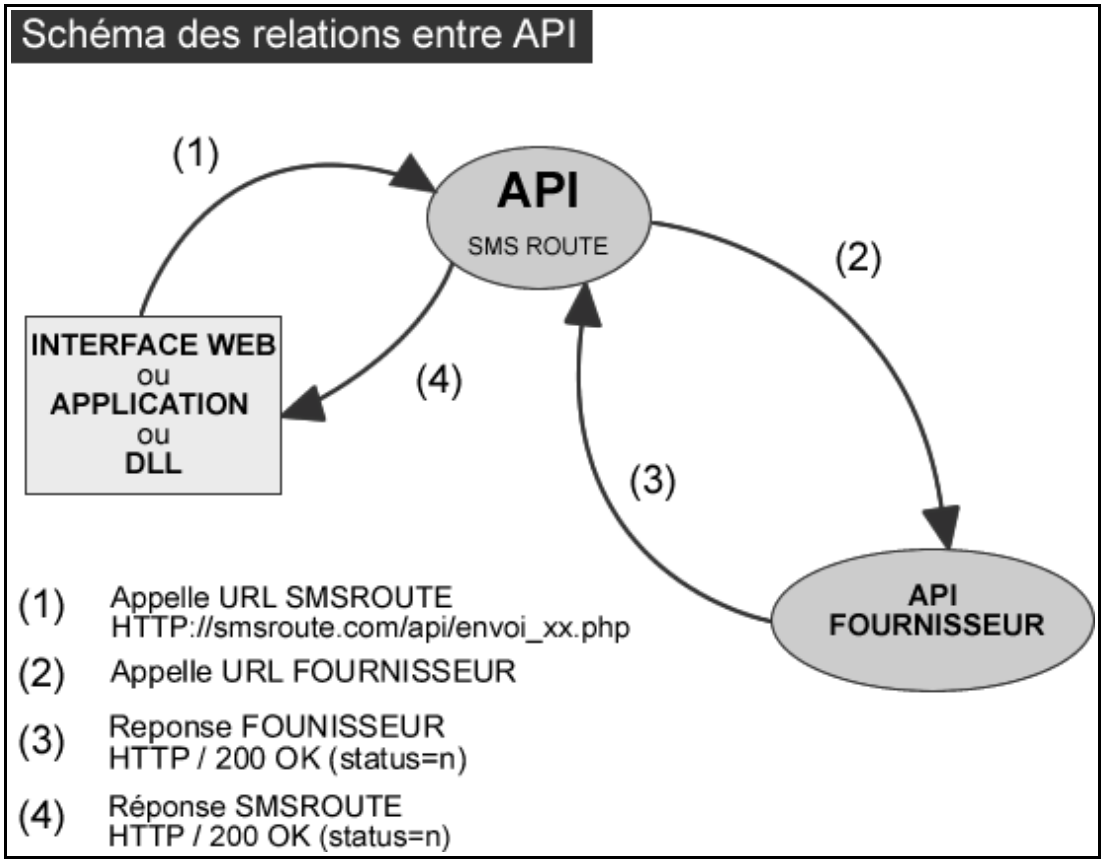
Coté OCITO (ou autre prestataire dans le futur) :

- Recevoir de la part de l'API SMSROUTE, des messages FTP (fichiers) ou HTTP (api Web)
- Émettre des réponses de prise en charge
- Émettre des accusés de réception.
- Émettre des SMS via un opérateur téléphonique standard

Coté clients AIC :

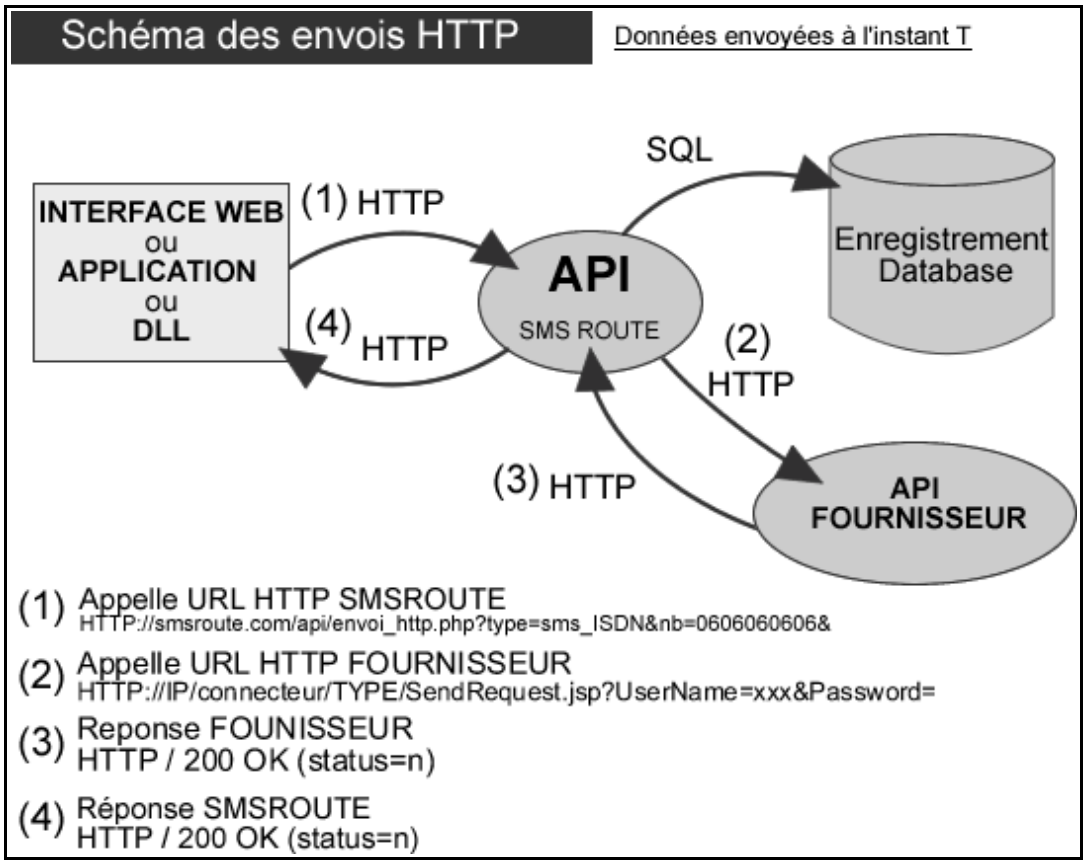
- Émettre des SMS en HTTP, FTP OU SMTP
- Réceptionner les réponses de l'API AIC

- Explications :

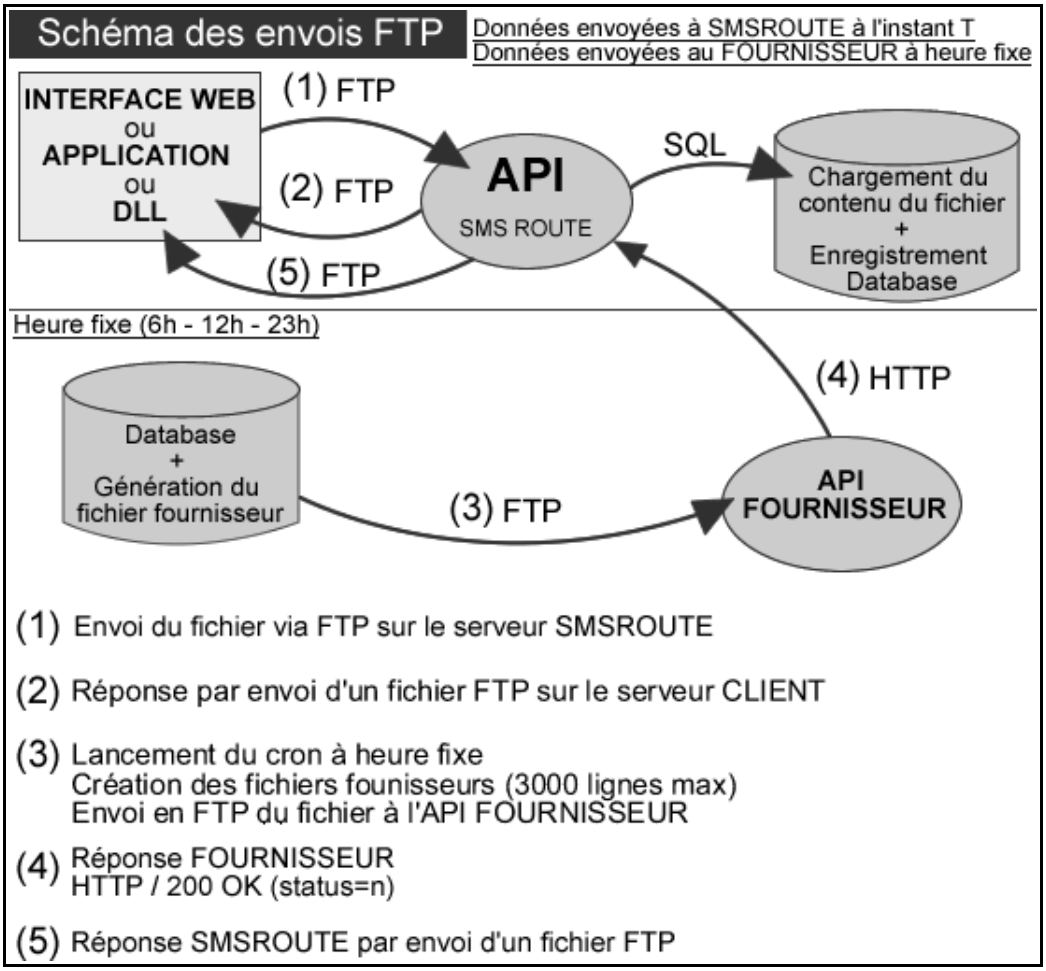


3.2 Description de l’ensemble des fonctions :

- Fonction d'envoi de messages SMS ou WAP via l'api SMSROUTE HTTP :
    - L'api SMSROUTE permettra d'envoyer des SMS en mode SMS texte (ISDN et ALIAS) ainsi qu'en mode Push WAP (texte, alias ou mail) en utilisant le protocole HTTP
    - Description fonctionnelle :
        - *traitements* : L'api recevra des requêtes HTTP venant soit de l'interface Web, soit directement via le protocole HTTP (Telnet ou autre terminal), pour cela une URL de la forme « http://smsroute.net/api/envoi_http.php?sujet=xxx,... » sera mise à disposition de l'utilisateur. Après traitement des données suivant le schéma de traitement (MOT), l'api SMSROUTE enverra de manière automatisée et instantanée les données à l'API fournisseur en utilisant le protocole HTTP.
        - *écrans*



- Fonction d'envoi de messages SMS ou WAP via l'api SMSROUTE FTP :
    - L'api SMSROUTE permettra d'envoyer des SMS en mode SMS texte (ISDN et ALIAS) ainsi qu'en mode Push WAP (texte, alias ou mail) en utilisant le protocole FTP
    - Description fonctionnelle :
        - *traitements* : L'api recevra des fichiers csv formatés sur son serveur FTP venant soit de l'interface Web permettant l'upload de fichier, soit directement via un serveur FTP où l'utilisateur déposera un fichier CSV. Le serveur FTP (proftpd) utilisera le module mysql pour authentifier les utilisateurs par rapport à la base Joomla. Après traitement des données suivant le schéma de traitement (MOT), l'api SMSROUTE enverra de manière automatisée et différée les données à l'API fournisseur en utilisant l'upload d'un fichier re-formaté via FTP.
        - *écrans*

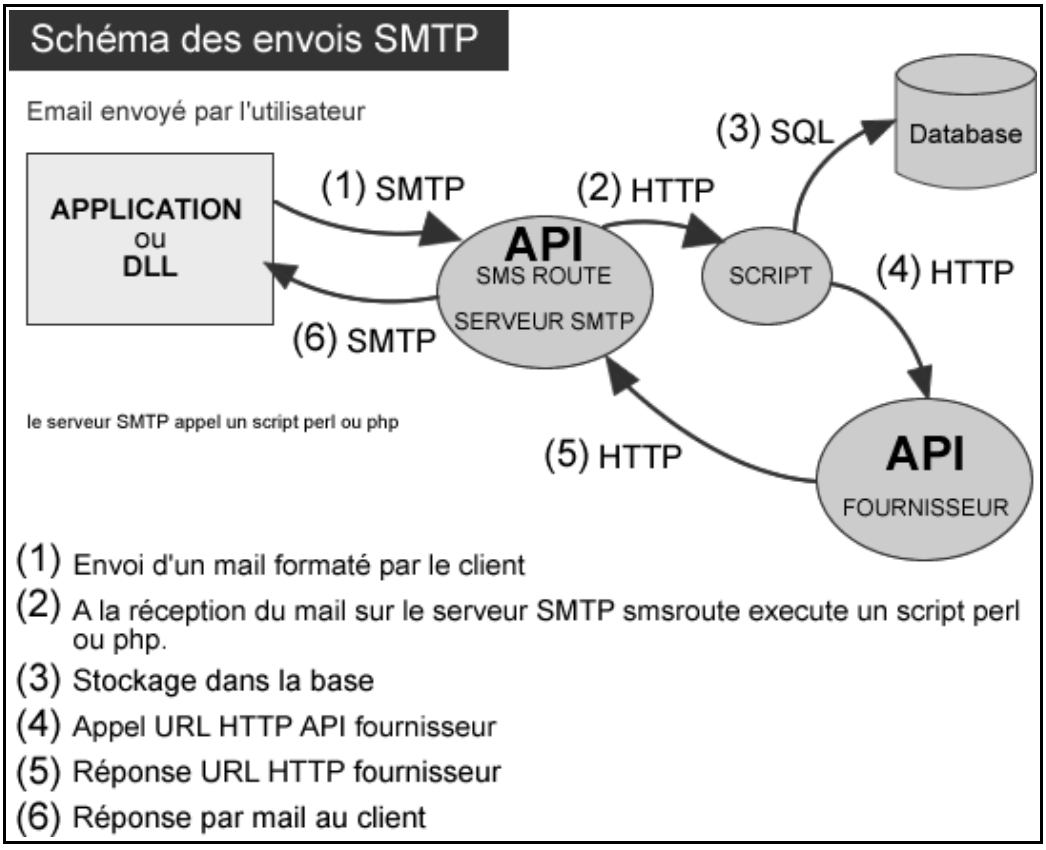


- Fonction d'envoi de messages SMS ou WAP via l'api SMSROUTE SMTP :
    - L'api SMSROUTE permettra d'envoyer des SMS en mode SMS texte (ISDN et ALIAS) ainsi qu'en mode Push WAP (texte, alias ou mail) en utilisant le protocole SMTP

    - Description fonctionnelle :

        - *traitements* : L'api recevra des mails formatés provenant de l'utilisateur. L'objet du mail contiendra une clef d'envoi SMTP qui sera générée lors d'une création de compte SMSROUE. Après traitement des données, l'api SMSROUTE enverra de manière automatisée et instantanée les données à l'API fournisseur en utilisant le protocole HTTP. (cette procédure n'est pas détaillée dans le modèle de traitement joint en annexe).

        - *écrans* :



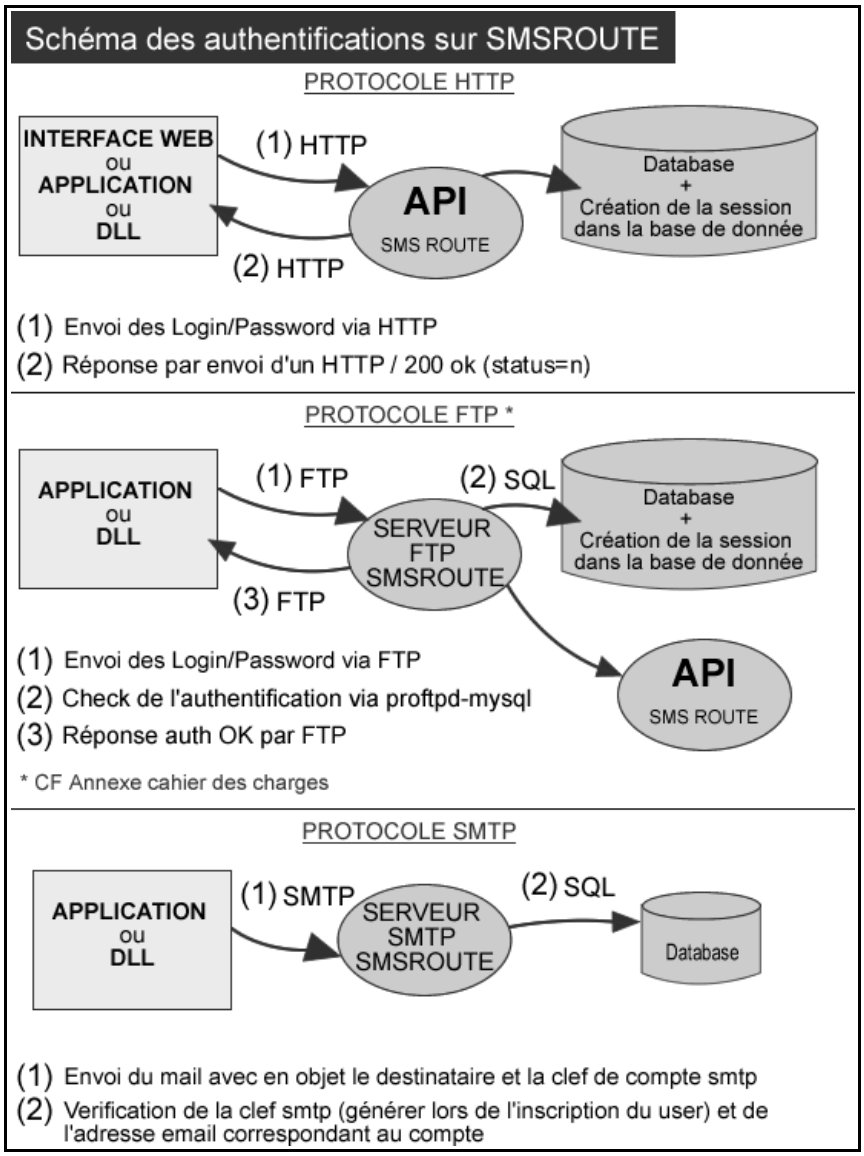
- Fonction d'authentification utilisateur via l'API SMSROUTE :

    - L'api SMSROUTE permettra d'authentifier à distance ou via l'interface Web, un utilisateur possédant un login et mot de passe. La partie « client » et la partie « utilisateur » sont à différencier. En effet le client pourra se loguer uniquement sur l'interface afin de créer ses utilisateurs. L'utilisateur, lui, pourra utiliser l'API en mode Web ou autre. La gestion de la session d'identification sera faite par PHP et stockées dans une base de donnée.

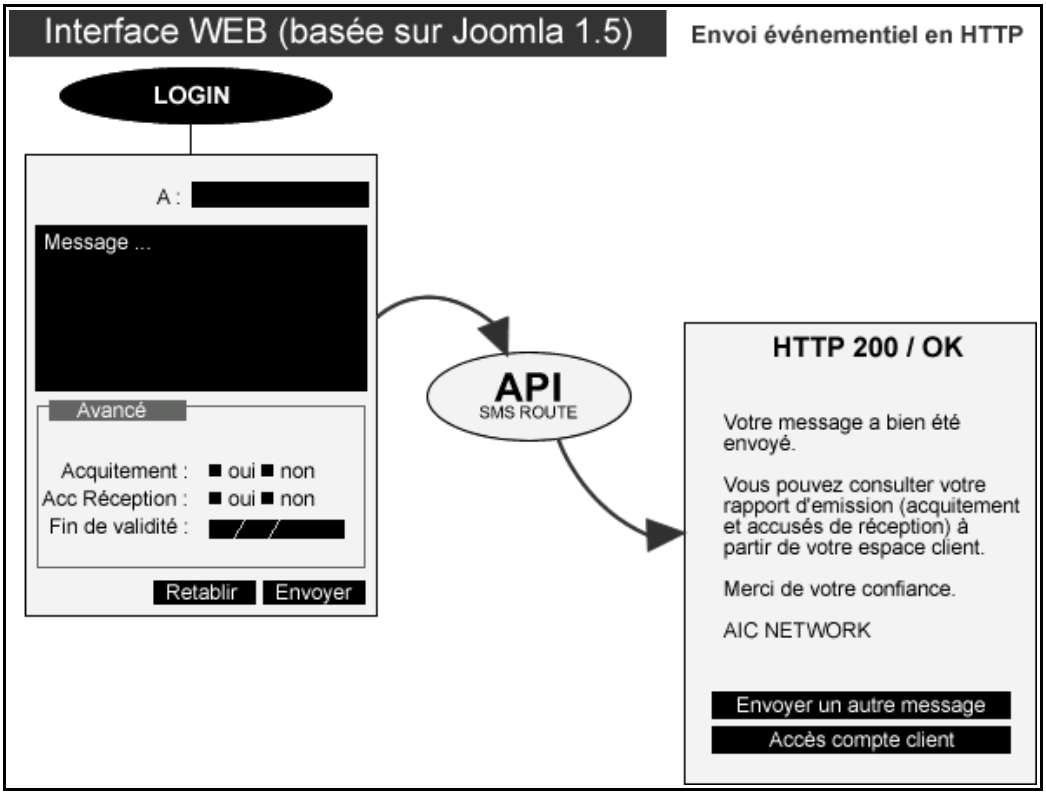
    - Description fonctionnelle :

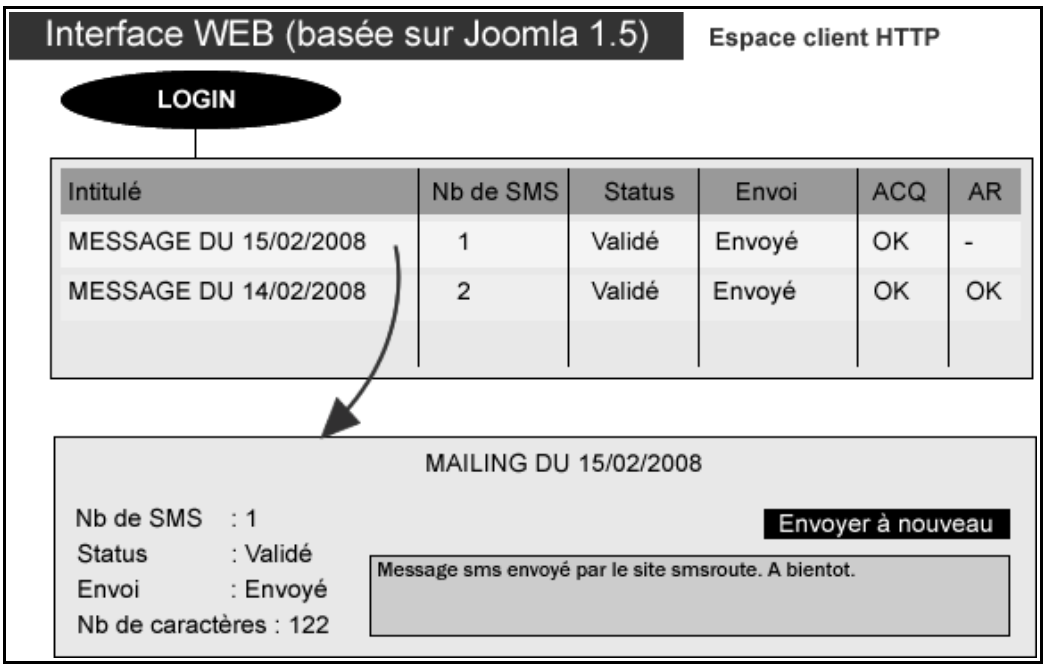
        - *traitements* : L'api recevra des mails formatés provenant de l'utilisateur. Après traitement des données suivant le schéma de traitement (MOT), l'api SMSROUTE enverra de manière automatisée et instantanée les données à l'API fournisseur en utilisant le protocole HTTP.

        - *écrans*

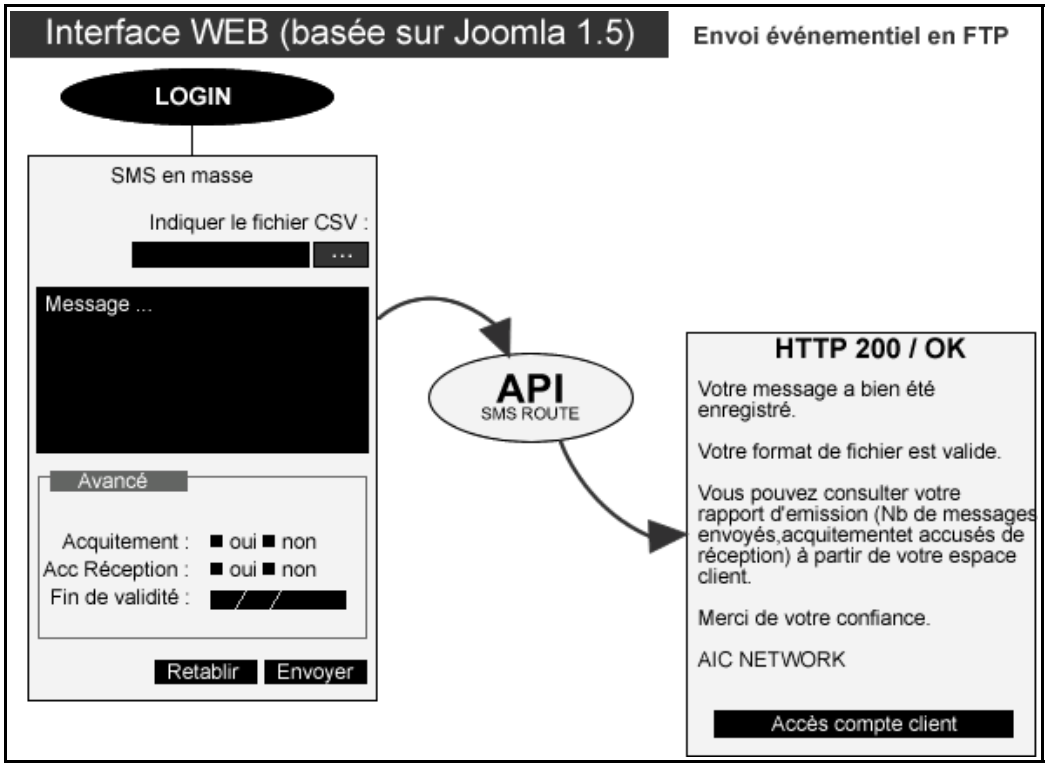


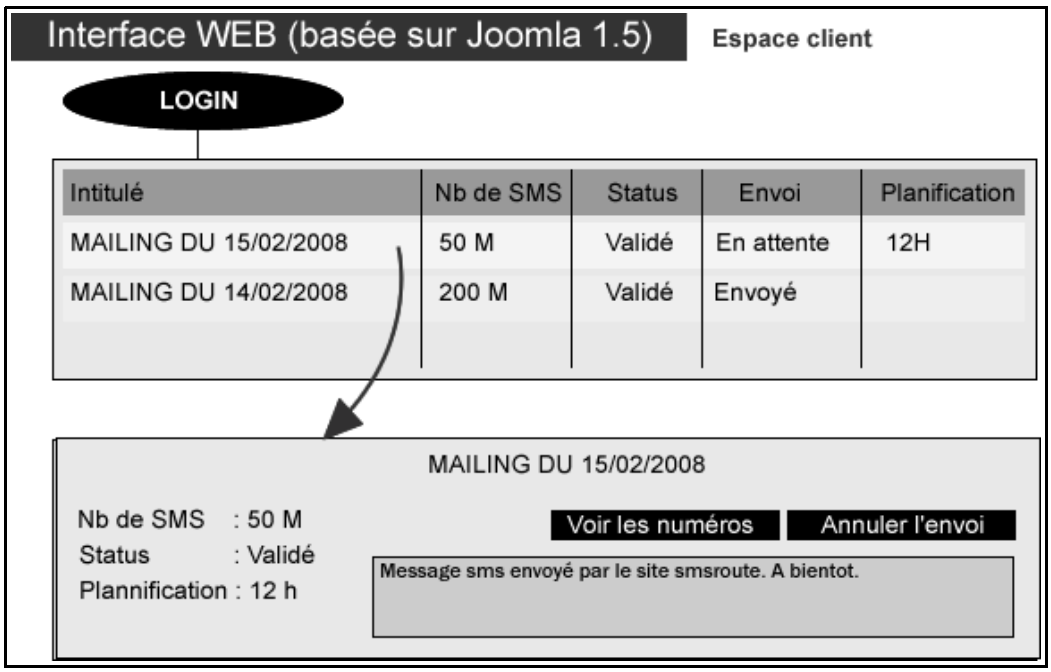
- Fonction d'envoi de messages SMS ou WAP via l'interface Web (événementiel) :
    - L'interface Web permettra d'envoyer des SMS en mode SMS texte (ISDN et ALIAS) ainsi qu'en mode Push WAP (texte, alias ou mail) en utilisant l'api AIC NETWORK
    - Description fonctionnelle :
        - *traitements* :
        - *écrans :*





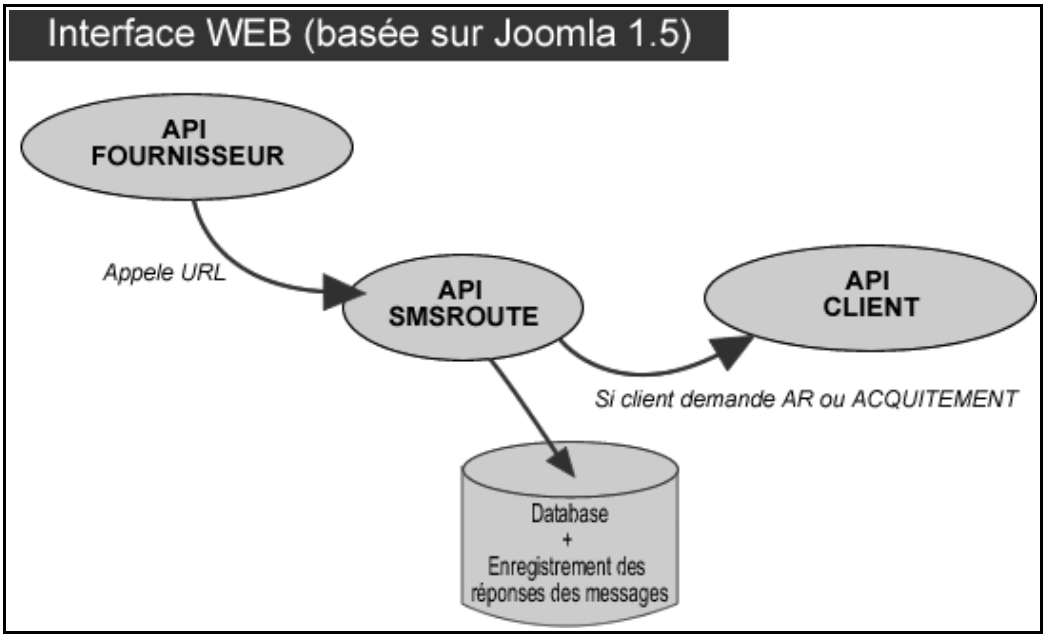
- Fonction d'envoi de messages SMS ou WAP via l'interface Web (Mass Mailing) :
    - L'interface Web permettra d'envoyer des SMS en mode SMS texte (ISDN et ALIAS) ainsi qu'en mode Push WAP (texte, alias ou mail) via l'upload d'un fichier csv pré formaté.
    - Description fonctionnelle :
        - *traitements* :
        - *écrans* :





- Fonction de réception de SMS MO :
    - L'interface Web permettra de recevoir des SMS MO. (message sms émis depuis un téléphone portable vers le numéro AIC).
    - Description fonctionnelle :
        - *traitements* : Les messages SMSMO pourront être accessibles uniquement via le compte client du site SMSROUTE. La réception du mot clef « STOP » déclenchera immédiatement le black listing du numéro pour ce client.
        - *écrans* :

- Fonction de réception des accusés d'acquittement et de réception :
    - L'API fournisseur enverra des réponses HTTP à l'API SMSROUTE afin que celle ci soit notifiée de la bonne prise en charge de son fichier, mais également des acquittement ou accusés de réception.
    - Description fonctionnelle :
        - *traitements* : Les réponses seront dans un premier temps des confirmations de prise en charge des requêtes d'envoi (confirmation de format valide). Dans un deuxième temps si l'utilisateur en fait la demande (suivant tarification), l'API fournisseur pourra être amenée à appeler une URL de l'API SMSROUTE afin d'envoyer des acquittements ou des accusés de réception.
        - *écrans*



4 Les données

4.1 Modèle détaillé des données

CF MLD EN ANNEXE

4.2 Description des données

CF MCD/UML EN ANNEXE

4.3 Reprise de l'existant

- Aujourd'hui AIC NETWORK dispose d'une plate forme SMS ROUTE permettant l'envoi de façon basique des SMS. Cet envoi s'effectue via une interface web simple, et l'administrateur dispose d'un logiciel assez évolué pouvant tracé les messages envoyés et autre fonctions plus avancées.
- L'idée à terme est de ne plus utiliser du tout l'ancienne plate forme.

5 Les traitements

5.1 Schéma général des traitements

CF MOT EN ANNEXE

5.2 Descriptif des flux

CF MOT EN ANNEXE

5.2.1 Fichiers (Upload FTP)

- *le support physique :* Disque dur / via serveur FTP
- *le format :* CSV
- *provenance :* Disque dur du client ou application cliente
- *la fréquence de réception :* événementielle
- *la fréquence d'accès :* trois fois par jour via un cron
- *le volume (moyen, mini, maxi) :* nombre de ligne définit par le fournisseur (default : 3000)
- *les points particuliers :* Doit être vérifier avant le stockage dans la base SMSROUTE précédent l'envoi
- *les données véhiculées : Numéro* de téléphone, message, ID d'application, LineId

5.2.2 Éditions

- *les données :* FICHIER CSV
- *la fréquence et les conditions d'édition :* Édition lors de l'envoi au fournisseur
- *le nombre d'exemplaire* : Fichier scindé en un ou plusieurs fichiers formaté correctement pour l'api fournisseur et contenant un nombre limité de ligne (défault 3000).
- *le volume* : Nombre de ligne définit dans la table fournisseur
- *les destinataires* : API Fournisseur

5.2.3 Messages

Message d'erreur en sortie standard

6 Versionning

Module	I/O/A *	Version	Délais (Hors tests)
Environnement serveur	I	0.0.0	1 H
Installation de Joomla	I	0.0.1	3 H
API HTTP SEND (communications avec API Four.)	A	0.1.1	3 J
API FTP SEND (upload et communications API four.)	A	0.1.2	3 J
API FTP AUTH Joomla / proftp-mysql	I/O	0.1.3	2J
API FTP CRON	I/O	0.1.4	2 J
API HTTP RESP FOUR. (http et smtp)	A	0.2.1	1 J
API FTP RESP FOUR.	A	0.2.2	1 J
API HTTP REPONSE CLIENT	A	0.3.1	1 J
API FTP REPONSE CLIENT	A	0.3.2	1 J
API SMTP (configuration serveur smtp)	I	0.4.1	2 J
API SMTP (receive smtp, send http, response smtp)	A/O	0.4.2	2 J
API SMS-MO	A/O	0.4.3	2 J
INTERFACE WEB HTTP	A/I	0.5.1	2 J
INTERFACE WEB FTP	A/I	0.5.2	2 J
BACK OFFICE ADMIN (Joomla)	I	0.6.1	1 J
BACK OFFICE PARTENAIRE	A/I	0.6.2	1 J
BACK OFFICE USER	A/O	0.6.3	2 J
BATTERIE DE TEST (Release Beta)	A/I/O	0.7.1	7 - 10 J
RELEASE PRODUCTION VERSION 1.0		1.0.0	

ANNEXE CAHIER DES CHARGES :

ETUDE DU LOGIN SUR PROFTPD VIA L'EXTENTION MYSQL PROFTPD

A / Liens relatifs :

http://www.lea-linux.org/cached/index/Reseau-partfic-proftpd_mysql.html
http://archives.2037.org/viewtopic.php?t=36495

Exemple fichier de configuration :

```
# On inclut le fichier modules.conf
# =====================
Include /etc/proftpd/modules.conf

# Configuration de base
# =====================
ServerName "Mon serveur FTP"
ServerType standalone
ServerIdent on "Bienvenue sur mon ftp. Veuillez-vous identifiez"
DeferWelcome on
ServerAdmin "ftp_admin@mydomain.com"

MultilineRFC2228 on
DefaultServer on
ShowSymlinks on
AllowOverwrite on

TimeoutNoTransfer 600
TimeoutStalled 600
TimeoutIdle 1200

ListOptions "-l"
Defaultroot ~

DenyFilter \*.*/

Port 21

# A configurer selon sa connection
# ================================
MaxInstances 6
MaxLoginAttempts 3
MaxClientsPerUser 10
MaxClientsPerHost 2
MaxHostsPerUser 4
MaxClients 6 "Limite a 6 utilisateurs"

# ProFTPD est excecuté avec des droits réduits
# ============================================
User nobody
Group nogroup

Umask 022
AllowStoreRestart on
AllowRetrieveRestart on

# Mod MySQL
# =========
# Les mots de passe sont crypté ds la base avec la fct ENCRYPT (MySQL)
```

```
SQLAuthTypes Crypt
SQLAuthenticate users* groups*

# Modifiez cette ligne selon l'utilisateur et le mot de passe définit plutôt
SQLConnectInfo proftpd@localhost proftpd password

# On donne à ProFTPD le nom des colonnes de la table usertable
SQLUserInfo ftpuser userid passwd uid gid homedir shell
SQLUserWhereClause "LoginAllowed = 'true'"

# On donne à ProFTPD le nom des colonnes de la table "grouptable"
SQLGroupInfo ftpgroup groupname gid members

# Créer le repertoire home de l'utilisateur si il n'existe pas
SQLHomedirOnDemand on

# Met à jour les compteurs à chaque connection d'un utilisateur
SQLLog PASS updatecount
SQLNamedQuery updatecount UPDATE "count=count+1, accessed=now() WHERE userid='%u'" ftpuser

#Met à jour les compteurs à chaque upload ou download d'un utilisateur
SQLLog STOR,DELE modified
SQLNamedQuery modified UPDATE "modified=now() WHERE userid='%u'" ftpuser

# Mod quota
# =========
QuotaEngine on
QuotaDirectoryTally on
QuotaDisplayUnits Mb
QuotaShowQuotas on

# Définit les requêtes SQL pour que ProFTPd recupere les infos sur les quotas

SQLNamedQuery get-quota-limit SELECT "name, quota_type, par_session, limit_type, bytes_up_limit,
bytes_down_limit, bytes_transfer_limit, files_up_limit, files_down_limit, files_transfer_limit FROM
ftpquotalimits WHERE name = '%{0}' AND quota_type = '%{1}'"

SQLNamedQuery get-quota-tally SELECT "name, quota_type, bytes_up_total, bytes_down_total,
bytes_transfer_total, files_up_total, files_down_total, files_transfer_total FROM ftpquotatotal WHERE name
= '%{0}' AND quota_type = '%{1}'"

SQLNamedQuery update-quota-tally UPDATE "bytes_up_total = bytes_up_total + %{0}, bytes_down_total =
bytes_down_total + %{1}, bytes_transfer_total = bytes_transfer_total + %{2}, files_up_total = files_up_total +
%{3}, files_down_total = files_down_total + %{4}, files_transfer_total = files_transfer_total + %{5} WHERE
name = '%{6}' AND quota_type = '%{7}'" ftpquotatotal

SQLNamedQuery insert-quota-tally INSERT "%{0}, %{1}, %{2}, %{3}, %{4}, %{5}, %{6}, %{7}" ftpquotatotal

QuotaLimitTable sql:/get-quota-limit
QuotaTallyTable sql:/get-quota-tally/update-quota-tally/insert-quota-tally

RootLogin off
RequireValidShell off

# Gestion des logs
# ================
# Enregistre les requêtes SQL dans /var/log/proftpd/mysql.log
SQLLogFile /var/log/proftpd/mysql.log
# Enregistre les authentifications
LogFormat auth "%v [%P] %h %t \"%r\" %s"
ExtendedLog /var/log/proftpd/auth.log AUTH auth
# Enregistre les accès aux fichiers
LogFormat write "%h %l %u %t \"%r\" %s %b"
```

```
ExtendedLog /var/log/proftpd/access.log WRITE,READ write

# Recupère le nom à partir de l'ip de la machine de l'utilisateur ( resolution DNS )
IdentLookups on
```

DESCRIPTION DE LA DIRECTIVE SERVER SQLLOG

SQLLog

Syntax: SQLLog *cmd-set query-name ["IGNORE_ERRORS"]*
Default: None
Context: server config, `<VirtualHost>`, `<Global>`
Module: mod_sql
Compatibility: 1.2.1 and later

This directive is used to log information to a database table. Multiple `SQLLog` directives can be in effect for any command; for example, a user changing directories can trigger multiple logging statements.

The first parameter to `SQLLog`, the *cmd-set*, is a comma-separated (**no spaces**) list of FTP commands for which this log command will trigger. The list of commands is too long to list in entirety; commands include `CWD, DELE, HELP, LIST, MKD, MODE, NLST, PASS, PASV, PORT` and many more. For the complete list check the FTP RFCs. Normally `mod_sql` will log events after they have completed successfully; in the case of the `QUIT` command, `mod_sql` logs prior to the server's processing of the command. (Note, however, that the client *may not* issue a `QUIT` before logging out; in this case, use a command of `EXIT` rather than `QUIT`. `EXIT` is not a real FTP command, but it is used here to provide a means for having `SQLLog` work whenever a session ends.)

FTP commands in the command set will only be logged if they complete successfully. Prefixing any command with "ERR_" will cause logging to occur only if there was an error in the command's processing. To log both errors and successful completion of a given command *X*, therefore, you'll need both "*X*" and "ERR_*X*" in your *cmd-set*.

The special command "*" matches all FTP commands, while "ERR_*" matches all errors.

The second parameter is the name of a query defined by a `SQLNamedQuery` directive. The query must be an `UPDATE`, `INSERT`, or `FREEFORM` type query; explicit `SELECT` queries will not be processed.

The third parameter is optional. If you add "IGNORE_ERRORS" as the third parameter, `SQLLog` ***will not*** check for errors in the processing of the named query. Any value for this parameter other than the string "IGNORE_ERRORS" (case-insensitive) will not cause errors to be ignored.

Normally, `SQLLog` directives are considered important enough that errors in their processing will cause `mod_sql` to abort the client session. References to non-existent named queries will **not** abort the client session, but may result in database corruption (in the sense that the expected database `UPDATE` or `INSERT` will not occur). Check your directives carefully.

Examples:

```
SQLLog PASS updatecount
SQLNamedQuery updatecount UPDATE "count=count+1 WHERE userid='%u'" users
```

together, these replicate the deprecated "`SQLLoginCountField` count" directive; if the current user was "joe", this would translate into the query "UPDATE users SET count=count+1 WHERE

userid='joe'". This query would run whenever a user was first authenticated.

```
SQLLog CWD updatedir
SQLNamedQuery updatedir UPDATE "cwd='%d' where userid='%u'" users
```

together these replicate the logging side of the deprecated "`SQLLogDirs` cwd" directive; if the current user was "joe" and the current working directory were `/tmp`, this would translate into the query "UPDATE users SET cwd='/tmp' WHERE userid='joe'". This query would run whenever a user changed directories.

```
SQLLog RETR,STOR insertfileinfo
SQLNamedQuery insertfileinfo INSERT "'%f', %b, '%u@%v', now()" filehistory
```

would log the name of any file stored or retrieved, the number of bytes transferred, the user and host doing the transfer, and the time of transfer (at least in MySQL). This would translate into a query like: "INSERT INTO filehistory VALUES ('somefile', 12345, 'joe@joe.org', '21-05-2001 20:01:00')"

PROPOSITION TECHNIQUE POUR L'AUTHENTIFICATION FTP

- Installation du module proftpd-mysql
- Installation de Joomla 1,5,1
- Authentification via le module proftpd-mysql en utilisant la table Joomla modifiée
- A l'authentification, on check si déjà logué sinon on régénère la session (voir la méthode `_construct` de la classe `Jsession`)
- Au délog on délogue dans joomla