

LdapSaisie

Benjamin Renard

**brenard@easter-eggs.com
benjamin.renard@zionetrix.net**

LdapSaisie
par Benjamin Renard

Table des matières

1. Introduction.....	1
1.1. Fonctionnalités	1
2. Installation.....	3
2.1. Pré-requis	3
2.2. Téléchargement	4
2.2.1. A partir du paquet Debian	4
2.2.2. A partir de Git.....	4
2.2.3. A partir des snapshot	5
2.3. Arborescence du projet	5
2.4. Tutoriel d'installation	6
3. Mise à jour.....	11
3.1. Procédure de mise à jour	11
3.1.1. Installation via paquet Debian	11
3.1.2. Installation à partir des sources	11
3.2. Mise à jour 2.4.1 -> 3.0.0.....	12
3.2.1. Fichier config.inc.php.....	12
3.2.2. Fichiers CSS	13
3.2.3. Fichiers PHP	13
3.2.4. Fichiers templates :.....	15
3.2.5. Tous les fichiers : Modification des URLs.....	18
4. Configuration	20
4.1. Configuration globale.....	20
4.1.1. Variables globales	20
4.1.2. Variables et constantes indépendantes.....	34
4.1.3. Format paramétrable.....	40
4.1.4. Paramètres étendus des recherches dans l'annuaire	41
4.2. Configuration LSobject	41
4.2.1. Configuration des attributs.....	45
4.2.2. Création automatique du conteneur des LSobjets dans un subDn	92
4.2.3. Déclencheurs	93
4.2.4. customActions	94
4.2.5. LSrelation	97
4.2.6. LSform.....	99
4.2.7. LSsearch	102
4.2.8. ioFormat	110
4.3. Configuration des LSaddons	113
4.3.1. LSaddon_asterisk	113
4.3.2. LSaddon_exportSearchResultAsCSV	113
4.3.3. LSaddon_LSaccessRightsMatrixView	114
4.3.4. LSaddon_mail.....	114
4.3.5. LSaddon_maildir	116
4.3.6. LSaddon_mailquota.....	116
4.3.7. LSaddon_phpldapadmin.....	117
4.3.8. LSaddon_showTechInfo	118
4.4. Configuration des LSauthMethods.....	119

4.4.1. LSauthMethod_HTTP	119
4.4.2. LSauthMethod_CAS	121
4.4.3. LSauthMethod_anonymous.....	123
5. API	124
5.1. Authentification.....	124
5.2. Méthodes exposées.....	124
6. Contribution	135
6.1. LSaddons.....	135
6.1.1. Structure d'écriture	135
6.1.2. Les vues personnalisées.....	138
6.1.3. Les commandes <i>CLI</i> personnalisées	140
6.2. LSformElements	144
6.3. LSformRules	146

Chapitre 1. Introduction

LdapSaisie est une application web d'administration d'annuaire LDAP développée en PHP/Javascript. Cette application a pour but d'abstraire la complexité d'un annuaire par l'intermédiaire d'une interface d'administration simple et intuitive. L'application a été conçue avec pour objectif premier une modularité maximum, ce qui permet l'extension ou l'adaptation facile de l'application par l'intermédiaire de modules, d'extensions et de greffons. Cette application peut être utilisée pour administrer le système d'information basé sur l'annuaire LDAP et également en parallèle pour permettre aux utilisateurs d'avoir accès aux données les concernant et éventuellement de les modifier.

1.1. Fonctionnalités

De part sa modularité, LdapSaisie est facilement extensible. Cependant, voici une liste non-exhaustive de ses fonctionnalités :

- Gestion d'annuaire simple et multi-branches
- Gestion d'un nombre illimité de types d'objets
- Gestion d'un nombre illimité de populations se connectant à l'interface
- Gestion fine des droits des utilisateurs, permettant la maîtrise des droits d'accès sur les objets de l'annuaire et leurs attributs, tout en permettant la délégation de droits.
- Gestion d'un grand nombre de types d'attributs :
 - Texte (court ou long)
 - Date (format paramétrable)
 - Booléen (valeurs paramétrables)
 - Image/Photo
 - Mot de passe (génération de mot passe avec gestion d'une politique fine)
 - Adresse mail
 - Flux RSS
 - Lien web (URL)
 - Adresse XMPP
 - *Maildir*
 - Quota de mails
 - Clef publique SSH
 - Liste déroulante à choix simple ou multiple
 - Relation à d'autres objets de l'annuaire/ Exemple : membres d'un groupe, parrain d'un utilisateur, ... (valeur clé paramétrable)

Note : Chaque type d'attribut à des fonctionnalités qui lui sont propres et qui rendent plus facile et agréable l'utilisation de l'interface (génération automatique de mot de passe, génération des valeurs d'un champ à partir d'autres, ...).

- Gestion d'un grand nombre de règles de vérification des valeurs des attributs :
 - Alpha-numérique
 - Lettres uniquement
 - Longueur maximale/minimale d'une chaîne de caractères
 - Valeur différente de zéro
 - Pas de signe de ponctuation
 - Valeur numérique
 - Comparaison de valeur
 - Date
 - Adresse mail
 - Poids d'une image
 - Taille d'une image
 - Type de fichiers images
 - Politique de mot de passe (longueur/caractères autorisés/caractères obligatoires)

- Gestion simplifiée des relations entre les objets de l'annuaire
- Interface facilement personnalisable grâce à l'utilisation d'un système de template.
- Possibilité de positionner des déclencheurs permettant d'exécuter vos propres scripts, fonctions ou méthodes au moments précis où l'utilisateur crée, modifie ou supprime un objet ou un de ses attributs. Ces déclencheurs, en fonction de leur positionnement, peuvent influencer le comportement de l'application en empêchant par exemple, la validation des données d'un formulaire.
- Gestion fine de l'affichage des attributs en fonction de l'écran (=vue) sur lequel se trouve l'utilisateur.
- Gestion des dépendances entre attributs, permettant par exemple de régénérer automatiquement la valeur d'un attribut caché lors de la modification d'un autre.
- Possibilité de gérer des attributs entièrement cachés, dont les valeurs seront modifiées lors de la modification d'un attribut en dépendance.

Chapitre 2. Installation

2.1. Pré-requis

- Le service Apache HTTP avec le module `mod_rewrite` d'activé. Les règles de réécriture d'URL sont définies dans le fichier `.htaccess` fourni avec l'application et il est donc nécessaire d'autoriser une telle configuration à ce niveau via la directive `AllowOverride` devant inclure à minima `FileInfo`.
- L'utilisateur exécutant le serveur web doit avoir les droits d'écriture sur le dossier `'tmp'`. En cas d'installation à partir du paquet Debian, ce dossier est remplacé par un lien symbolique vers le dossier `/var/cache/ldapsaisie/`.
- PHP (<http://www.php.net/>) 5.6 (ou supérieur) avec `magic_quotes_gpc` et `register_globals` à `off`. L'outil CLI de PHP est par ailleurs nécessaire pour l'utilisation des outils CLI fournis avec l'application (fourni par le paquet `php-cli` dans Debian).
- Le support LDAP dans PHP (<http://www.php.net/>) (paquet `php-ldap` dans Debian)
- Le support mhash dans PHP (<http://www.php.net/>) (paquet `php5-mhash` dans Debian Lenny, intégré à `php-common` dans les versions supérieurs)
- Le support json dans PHP (<http://www.php.net/>) (**pear install pecl/json** sur RedHat, intégré au paquet `php5-common` précédemment)
- `Net_LDAP2` (http://pear.php.net/package/Net_LDAP2) (paquet `php-net-ldap2` dans Debian ou **pear install net_ldap2**)
- Le support mbstring dans PHP (<http://www.php.net/>) (paquet `php-mbstring` depuis Debian Stretch, intégré au paquet `php-common` dans Debian)
- Smarty (<http://www.smarty.net/>) (paquet `smarty3` dans Debian)
- La librairie `File_CSV_DataSource` (https://pear.php.net/package/File_CSV_DataSource) (paquet `php-file-csv-datasource` dans Debian)
- La librairie `Console_Table` (https://pear.php.net/package/Console_Table) (nécessaire pour le fonctionnement de l'outil CLI, paquet `php-console-table` dans Debian)
- Les librairies `Mail` (<https://pear.php.net/package/Mail>) et `Mail_Mime` (https://pear.php.net/package/Mail_Mime) (nécessaire pour l'envoi de courriels, paquets `php-mail` et `php-mail-mime` dans Debian)
- La librairie `Net_FTP` (https://pear.php.net/package/Net_FTP) (nécessaire pour le fonctionnement du LSaddon FTP, paquet `php-console-table` dans Debian)
- La librairie `PhpSecLib` (<https://github.com/phpseclib/phpseclib>) (nécessaire pour le fonctionnement du LSaddon SSH, paquet `php-console-table` dans Debian)

Avertissement

La librairie Net_LDAP2 (http://pear.php.net/package/Net_LDAP2) oblige le fait que la racine DSE de l'annuaire soit lisible en anonyme sinon la connexion à l'annuaire échouera systématiquement.

Note : Cette documentation est écrite à l'aide du langage Docbook. Les mécanismes d'exportation de celle-ci requiert un certain nombre de programmes et librairies :

- make (paquet make dans Debian)
- la feuille de style `html XSL` de Norman Walsh pour Docbook (fichier `/usr/share/xml/docbook/stylesheet/nwalsh/html/docbook.xsl` fournis par le paquet `docbook-xsl` dans Debian)
- `xmllint` (validation XML) (paquet `libxml2-utils` dans Debian)
- `jw` (exportation PDF) (paquet `docbook-utils` dans Debian)
- `dbtoepub` (exportation EPUB) (paquet `dbtoepub` dans Debian)

2.2. Téléchargement

2.2.1. A partir du paquet Debian

L'installation à partir du paquet Debian peut être réalisée soit en téléchargeant manuellement le paquet, soit en déclarant le dépôt APT suivant dans votre fichier `/etc/apt/sources.list` :

```
deb http://ldapsaisie.easter-eggs.org/debian stable main
```

Il ne vous restera ensuite plus qu'à installer le paquet `ldapsaisie` avec la commande suivante :

```
apt-get install ldapsaisie
```

Le fichier `/etc/ldapsaisie/apache.conf` est un exemple de configuration du serveur web Apache. La configuration du logiciel ce fera ensuite dans le dossier `/etc/ldapsaisie/local/`.

2.2.2. A partir de Git

Le dépôt Git peut être récupéré anonymement en utilisant la commande suivante :

```
git clone git://git.labs.libre-entreprise.org/ldapsaisie.git
```

La racine web de l'application se trouvera alors dans le dossier `/ldapsaisie/src/public_html/`.

2.2.3. A partir des snapshot

Toutes les nuits, un snapshot de l'arbre Git est réalisé et est téléchargeable au format `tar.gz` à l'adresse suivante : <http://ldapsaisie.easter-eggs.org/download/ldapsaisie-snapshoot.tar.gz> (<http://ldapsaisie.easter-eggs.org/download/ldapsaisie-snapshoot.tar.gz>)

2.3. Arborescence du projet

Racine

`doc/`

Les fichiers sources de la documentation (docbook).

`lsexample/`

Les fichiers relatifs à l'annuaire d'exemple.

`src/`

Les sources de l'application.

`public_html/`

La racine web de l'application : celle-ci ne contient que les fichiers `.htaccess` et `index.php` qui configure et déclenche la réécriture d'URL.

`conf/`

Contient les fichiers de configuration.

`LSobjects/`

Configuration des LSobjects.

`LSaddons/`

Configuration des LSaddons.

LSauth/

Configuration des LSauthMethod.

includes/

Contient les fichiers des ressources.

addons/

Les addons au projet.

class/

Les fichiers de définition des classes PHP (<http://www.php.net/>).

js/

Les fichiers Javascript.

libs/

Les librairies utilisées.

lang/

Les fichiers d'internationalisation.

templates/

Les fichiers *template* de l'interface. Il y a un sous-dossier par template.

css/

Les fichiers css de l'interface. Il y a un sous-dossier par template CSS.

images/

Les images de l'interface. Il y a un sous-dossier par template d'image.

local/

Les fichiers personnalisés de l'installation.

tmp/

Les fichiers temporaires (y compris le cache des templates).

2.4. Tutoriel d'installation

Cette section décrit les différentes étapes de l'installation de LdapSaisie. Deux méthodes d'installation sont présentées ici, l'une à partir des sources du projet et l'autre à partir du paquet Debian.

Dans ce tutoriel, nous partirons du principe que vous avez pleinement la main sur votre serveur (installation de nouveau paquet et configuration de votre serveur web). Nous partons également du principe que votre annuaire LDAP est déjà en place. Nous utiliserons pour cette exemple de mise ne oeuvre l'annuaire correspondant au schéma et à la configuration présente dans les sources du projet dans le dossier `lsexemple`.

1. La première étape consiste à installer le logiciel en tant que tel. Pour cela, référez vous au chapitre Téléchargement.

En cas d'installation à partir du paquet Debian, la configuration du logiciel se fera dans le dossier `/etc/ldapsaisie/local/`. Les fichiers placés dans ce dossier prévaleront toujours aux fichiers fournis par le paquet Debian, vous permettant facilement de modifier un composant existant ou dans écrire de nouveaux. Ainsi, pour modifier un fichier CSS par exemple, il vous suffira de le placer dans le dossier `/etc/ldapsaisie/local/css/`.

Pour une installation à partir du code source, il vous faut cloner le dépôt Git du projet dans le dossier `/var/www/ldapsaisie`. Pour cela il vous faut avoir installés les outils de Git contenu, dans Debian, dans le paquet `git-core`. Le dépôt Git doit ensuite être récupéré anonymement en utilisant la commande suivante :

```
git clone https://gitlab.easter-eggs.com/ee/ldapsaisie.git /var/www/ldapsaisie
```

Note : Pour que cette commande se déroule correctement, vous devez avoir accès au port TCP 443 du serveur `gitlab.easter-eggs.com`. En cas de problème vérifiez votre parefeu.

La suite des opérations se déroulera donc maintenant dans le dossier `/var/www/ldapsaisie`. Pour avoir plus de détails sur les éléments qu'on retrouve dans ce dossier, vous pouvez consulter la section concernée. Nous allons nous intéresser plus particulièrement :

- au script `upgradeFromGit.sh` permettant la mise à jour de votre repos tout en conservant les adaptations que nous ferons pour l'usage d'LdapSaisie adapté à notre annuaire ;
- au dossier `config.local` dans lequel seront stockés vos fichiers et vos adaptations de l'application ;
- au dossier `src/public_html` qui correspond à la futur racine du site web de l'application.

Le principe de l'adaptation est ici de mettre vos fichiers personnalisés dans le dossier `config.local`, de les déclarer dans votre fichier `config.local/local.sh` contenant la liste des fichiers devant être installés. Le fichier `local.sh` est la source de configuration du script `upgradeFromGit.sh`. Il faut donc dans un premier temps créer votre fichier `local.sh` en copiant

le fichier d'exemple `local.sh.example`. Ce fichier est un script bash déclarant les variables de configurations suivantes :

LOCAL_FILES

La liste des chemins des fichiers à installer dans l'arborescence du site. Cette élément doivent être séparés par des espaces ou des retour à la liste. Exemple :

```
conf/config.inc.php
lang/fr_FR.UTF8/lang.php
```

LOG_FILE

Nom du fichier de log des mises à jour.

THEME

Le nom du theme à installer (facultatif et non traité dans ce tutoriel).

BUILD_DOC

Variable booléenne définissant si la documentation doit être compiler en utilisant le script `buildDocExports.sh`. Ceci ne sera pas expliqué dans ce tutoriel et nous partirons donc du principe que cette variable est à 0.

Note : D'autres variables sont présentes dans ce fichier et concerne uniquement la compilation de la documentation. Elle peuvent être ignorée à partir du moment ou la variable `BUILD_DOC` vaut 0.

Note : Il est possible d'utiliser dans ce fichier de configuration la variable bash `$ROOT_DIR` correspondant au chemin du dossier d'installation, c'est à dire dans notre exemple `/var/www/ldapsaisie`.

2. La deuxième étape concerne la configuration globale de l'application : Cette partie est principalement contenue dans le fichier `src/conf/config.inc.php` (ou `/etc/ldapsaisie/local/conf/config.inc.php` en cas d'installation à partir du paquet Debian). En cas d'installation à partir du code source, il faut donc dans un premier temps copier ce fichier dans le dossier `config.local` et le déclarer dans la liste des fichiers à déployer lors des mises à jour (variable `LOCAL_FILES` dans le fichier `local.sh`). Il s'agit en particulier dans ce fichier de configurer la connexion à votre annuaire. Vous pouvez vous inspirer du fichier d'exemple fourni et pour plus de détails, reportez-vous à la section concernée.

Note : Notez qu'il est possible de passer l'application en mode *debug* ce qui peut être utile par la suite.

3. La troisième étape concerne la configuration des types de LSubjects : Chaque type d'objet manipulé par LdapSaisie doit correspondre avec un type de LSubject.

- a. Création du fichier de classe (*optional*) : Ce fichier contient la déclaration de la classe PHP correspondant au type de LSubject. Cette classe étend la classe *LSldapObject* qui contient pour ainsi dire toute les méthodes et propriétés nécessaires pour les types de LSubject simples. Si votre type de LSubject nécessite des méthodes ou propriétés particulières, vous pouvez implémenter cette classe. À défaut, une classe vierge d'adaptation sera automatiquement déclarée.

Les fichiers des classes sont contenus dans le dossier */includes/class/* et portent les noms composés de la manière suivante :

```
class.LSubjects.[nom du type d'LSubject].php
```

- b. Configurer vos LSubject : Cette partie est certainement la plus longue et consiste à déclarer l'ensemble des informations relatives aux types des objets LDAP manipulés. Les fichiers d'exemples fournis vous seront alors d'une aide précieuse. basé vous sur l'un de pour créer le votre. Pour cela le fichier de configuration du type d'LSubject *LSpeople* est le plus complet et est un bon point de départ. Pour plus de détails sur les éléments de configuration de ce fichier, reportez-vous à la section concernée.
- c. Configurer si nécessaire les relations entre les objets appelés LSrelations. Les relations les plus simples (via un attribut de liaison) pourront être implémentées à l'aide d'un simple paramétrage. Pour des relations, plus complexes, il sera possible d'implémenter des méthodes personnalisées pour les gérer. Pour plus de détails, reportez-vous à la section concernée.

Note : Pour avoir un exemple de fichier de classe PHP implémentant des méthodes de gestion de LSrelations complexes, vous pouvez consulter le fichier de classe *LSgroup*.

Important : En cas d'installation à partir du code source, pensez à déclarer les fichiers que vous venez de créer dans la variable `LOCAL_FILES` du fichier `local.sh`. Exemple pour le type d'LSubject portant comme nom `LSexample` :

```
src/conf/LSubjects/config.LSubjects.LSexample.php
src/includes/class/class.LSubjects.LSexample.php
```

Note : Vous pouvez également personnaliser l'interface : Il est possible de personnaliser à votre goût l'interface en écrivant votre template ou en modifiant simplement les fichiers CSS. Une partie de cette documentation concernera bientôt cette problématique. Patience...

4. En cas d'installation à partir du code source, une dernière étape à ce niveau consiste à lancer le script `upgradeFromGit.sh` pour qu'il installe les fichiers que vous venez de créer. Ce script est conçu pour dire tout ce qu'il fait donc en cas de problème vous devriez rapidement comprendre où cela

coince. Dans tout les cas, n'hésitez pas à poser vos questions à la communauté sur la liste `<ldapsaisie-users@lists.labs.libre-entreprise.org>`.

Chapitre 3. Mise à jour

Cette section de la documentation détaille la procédure de mise à jour d'une installation existante et regroupe des informations pratiques et utiles pour des montées de versions spécifiques entraînant par exemple une perte de rétrocompatibilité de la configuration.

3.1. Procédure de mise à jour

3.1.1. Installation via paquet Debian

Lors d'une installation par paquet Debian, la mise à jour est grandement facilité par le packaging: Il vous suffit de mettre à jour le paquet `ldapsaisie` :

```
apt update
apt install ldapsaisie
```

Une fois l'application mise à jour, prêtez attention aux nouveautés et point de vigilances décrite dans la section suivante.

3.1.2. Installation à partir des sources

Lors d'une installation par à partir des sources, le script `upgradeFromGit.sh` permet d'automatiser la mise à jour, à condition que vous ayez suivi la procédure d'installation à ce sujet.

Ce script s'occupera alors de :

- Nettoyer `working-tree` Git des liens symboliques des fichiers locaux (et éventuellement du thème) mis en place lors d'une précédente exécution ;
- Vider le cache des templates ;
- Mettre à jour le `working-tree` Git via un `git pull` de la mise à jour ;
- Installer des liens symboliques pour les fichiers locaux. En cas de fichier remplaçant un fichier livré avec l'application, le script vous notifiera en cas de changement intervenu dans le fichier fourni avec l'application et vous permettra de le mettre à jour simplement votre fichier local (via un `vim -d`) ;
- Détecter des changements dans les fichiers `MO` (traduction) et de déclencher dans ce cas un rechargement du serveur web pour prise en compte ;
- Option : de compiler une version locale à jour de la documentation ;

Une fois l'application mise à jour, prêtez attention aux nouveautés et points de vigilance décrite dans la section suivante.

3.2. Mise à jour 2.4.1 -> 3.0.0

Cette mise à jour majeure apporte de nombreuses nouveautés auxquelles il est important de prêter attention. Cette section ne parlera pas particulièrement de ces nouveautés, mais vous pouvez consulter le fichier `debian/ldapsaisie.NEWS`

(<https://gitlab.easter-eggs.com/ee/ldapsaisie/-/raw/master/debian/ldapsaisie.NEWS>) pour cela. Cette section listera en outre les points de vigilance à avoir et les adaptations à apporter sur votre configuration et votre code personnalisé.

3.2.1. Fichier `config.inc.php`

- ajout du paramètre `ConsoleTable` avec pour valeur par défaut sous Debian `/usr/share/php/Console/Table.php`
- ajout du paramètre `public_root_url` avec pour valeur par défaut sous Debian `/ldapsaisie`
- paramètre `$GLOBALS['defaultCSSfiles']` : il est nécessaire de modifier les URLs des fichiers listés : seul le nom du fichier doit rester, sa localisation sera détectée automatiquement. Par exemple, `$GLOBALS['defaultCSSfiles']=array('../light-blue.css');` devient `$GLOBALS['defaultCSSfiles']=array('light-blue.css');`
- les paramètres `authObjectType`, `authObjectFilter` et `authObjectTypeAttrPwd` sont remplacés par le tableau `LSobjects` dans le paramètre `LSauth`.

Par exemple:

```
'authObjectType' => 'LSpeople',
'authObjectFilter' => '(|(uid=%{user})(mail=%{user}))',
'authObjectTypeAttrPwd' => 'userPassword',
```

Devient:

```
'LSauth' => array (
  'LSobjects' => array(
    'LSpeople' => array(
      'filter' => '(|(uid=%{user})(mail=%{user}))',
      'password_attribute' => 'userPassword',
    ),
  ),
  [...]
),
```

- Une erreur de frappe historique a été corrigé dans le nom de la variable `$GLOBALS['defaultJSScripts']`, à savoir un "R" manquant.

- Les fichiers Javascript utilisés par défaut par l'application ne sont désormais plus listés dans la variable `$GLOBALS['defaultJSscripts']`. Seul doit y demeurer vos propres fichiers. Voici la liste des fichiers concernés et qui n'ont plus à être inclus via ce paramètre :

- `mootools-core.js`
- `mootools-more.js`
- `functions.js`
- `LSdefault.js`
- `LSinfosBox.js`

3.2.2. Fichiers CSS

Note : Les fichiers `light-*.css` ont été retravaillés pour tous *hériter* du fichier `light-blue.css` qui définit les couleurs de l'interface au travers des variables. Ainsi, il est très simple d'ajuster ce thème à vos couleurs. Si cela vous intéresse, vous pouvez prendre exemple sur les autres fichiers `light-*.css`.

Au passage, ce thème a été retravaillé pour prendre en compte la mise en forme d'un maximum de composants de l'application tout en profitant du côté responsive de l'interface apporter par cette mise à jour. Si vous avez un thème personnalisé, il est conseillé de regarder si celui-ci ne pourrait pas tirer partie du fichier `light-blue.css` en le surchargeant. À minima, vous pouvez analyser les évolutions de ce fichier pour identifier les modifications intéressantes à reporter sur votre thème personnel.

- Si vous utilisez un des fichiers `light-*.css` autre que le fichier `light-blue.css`, vous devez désormais également charger ce dernier en premier.
- corriger les URL des images : `url(../../images/default/find.png)` devient `url(../image/find)`. Pour identifier les fichiers CSS concernés, vous pouvez utiliser les commandes suivantes :


```
grep -Er 'url\(.+images' /etc/ldapsaisie/local/css
grep -Er 'url\(.+\. (png|gif|jpg)' /etc/ldapsaisie/local/css
```
- modification CSS page `fatal_error` (fichier `base.css`) : `#fatal_error` devient `#error`

3.2.3. Fichiers PHP

- `LSsession :: redirect()` devient `LSurl :: redirect()`. Pour identifier les fichiers CSS concernés, vous pouvez utiliser la commande suivante :


```
grep -Er 'LSsession *:: *redirect *\(' /etc/ldapsaisie/local/
```
- Les méthodes de gestion des Javascript et CSS additionnels ont été migrées de la classe `LSsession` vers la classe `LStemplate` :

- `LSsession :: addJSscript()` **devient** `LStemplate :: addJSscript()`.

Par ailleurs le paramètre `$path` disparaît et la méthode `addLibJSscript` à été ajoutée pour permettre spécifiquement l'inclusion des fichiers Javascript des librairies. Voici quelques exemples d'utilisation et leur équivalent à présent:

- `LSsession ::`
`addJSscript('../..../local/includes/js/LSformElement_eetelephone.js');`
devient `LStemplate :: addJSscript('LSformElement_eetelephone.js');`
- `LSsession ::`
`addJSscript('../..../local/includes/js/LSformElement_eetelephone.js');`
devient `LStemplate :: addJSscript('LSformElement_eetelephone.js');`
- `LSsession :: addJSscript('click-to-dial_view.js', 'local/includes/js/');`
devient `LStemplate :: addJSscript('click-to-dial_view.js');`
- `LSsession ::`
`addJSscript('Picker.js', LS_LIB_DIR.'arian-mootools-datepicker/');` **devient**
`LStemplate :: addLibJSscript('arian-mootools-datepicker/Picker.js');`
- `LSsession :: addJSconfigParam()` **devient** `LStemplate :: addJSconfigParam()`.
- `LSsession :: addHelpInfos()` **devient** `LStemplate :: addHelpInfo()`.
- `LSsession :: addCssFile()` **devient** `LStemplate :: addCssFile()`.

Par ailleurs le paramètre `$path` disparaît et la méthode `addLibCssFile` à été ajoutée pour permettre spécifiquement l'inclusion des fichiers CSS des librairies. Voici quelques exemples d'utilisation et leur équivalent à présent:

- `LSsession :: addCssFile('test.css', '../..../local/css/');` **devient**
`LStemplate :: addCssFile('test.css');` ; **Doit donc être conservé, que le nom du fichier CSS, pas de chemin vers celui-ci.**
- `LSsession ::`
`addCssFile('datepicker_vista.css', LS_LIB_DIR.'arian-mootools-datepicker/datepicker_vista');`
devient `LStemplate ::`
`addLibCssFile('arian-mootools-datepicker/datepicker_vista/datepicker_vista.css');`

Pour identifier les fichiers concernés, vous pouvez utiliser les commandes suivantes :

```
grep -Er 'LSsession *:: *(addJSscript|addLibJSscript|addJSconfigParam|addHelpInfos|addCssFile|addLibCssFile|addHelpInfo|addJSscript)' /etc/ldapsaisie/local/
grep -Er '(LSsession|LStemplate) *:: *addJSscript\(.*/local/ /etc/ldapsaisie/local/'
grep -Er '(LSsession|LStemplate) *:: *addJSscript\(.*/\.\.\./ /etc/ldapsaisie/local/'
grep -Er '(LSsession|LStemplate) *:: *addCssFile\(.*/local/ /etc/ldapsaisie/local/'
grep -Er '(LSsession|LStemplate) *:: *addCssFile\(.*/\.\.\./ /etc/ldapsaisie/local/'
```

- `LSlog` vs `LSdebug` : L'utilisation de `LSdebug` est dépriorisée en faveur de `LSlog`. Ce dernier ajoute désormais la notion de *logger*, permettant d'identifier la source des logs. Ce mécanisme permet la configuration d'un niveau de log spécifique pour un *logger* donné, ainsi que la mise en place de filtres au niveau des *handlers* pour ne logger par exemple que certains *loggers*, ou à l'inverse en exclure d'autres.
- Pour vos classes personnalisées : si celles-ci héritent d'une classe standard, il est fort probable qu'il soit possible d'utiliser des méthodes fournies par cette classe pour logger au travers un *logger* dédié (voir les méthodes `log_debug`, `log_info`, ...). À défaut, il est possible d'utiliser la classe `LSlog_staticLoggerClass` qui facilite l'implémentation.
- Pour vos `LSaddons` : il est conseillé d'utiliser un *logger* `LSaddon_[addon]` dédié. Le *logger* peut facilement être récupéré de la manière suivante :

```
LSlog :: get_logger("LSaddon_[addon]")
```

Cette méthode retourne une référence au *logger* et il est possible d'appeler directement une méthode de log, par exemple :

```
LSlog :: get_logger("LSaddon_[addon]") -> debug("message");
```

3.2.4. Fichiers templates :

3.2.4.1. Changement de l'inclusion des templates

- Le cas des fichiers `top.tpl` et `bottom.tpl`

```
{include file='ls:top.tpl' }
```

```
[...]
```

```
{include file='ls:bottom.tpl' }
```

devient :

```
{extends file='ls:base_connected.tpl' }
```

```
{block name="content" }
```

```
[...]
```

```
{/block}
```

Note : Pages à l'état connecté uniquement (incluant le menu, l'entête...).

- Fichiers avec entête HTML :

```
<html>
  <head>
    [...]
  </head>
```

```

<body>
  [...]
</body>
</html>

```

devient :

```

{extends file='ls:base.tpl'}
{block name="body"}
[...]
{/block}

```

Au besoin, si vous avez besoin :

- de remplacer les fichiers CSS chargés par défaut (`base.css` par exemple) : ajouter le block `css` :

```

{block name="css"}
  <link rel="stylesheet" type="text/css" href="{css name='custom.css'}" media="screen" t
  {include file='ls:css.tpl'}
{/block}

```

Note : Ce block contient tous les CSS, y compris ceux gérés par `LSsession :: addCssFile()`. Pensez à ajouter `{include file='ls:css.tpl'}` pour conserver ces derniers.

- d'ajouter des infos dans `<head>` : ajouter le block `head` (vide par défaut) :

```

{block name="head"}
[...]
{/block}

```

- d'ajouter des fichiers Javascript personnalisés : ajouter le block `js` (vide par défaut):

```

{block name="js"}
[...]
{/block}

```

Note : Ce block sera ajouté *APRÈS* les autres fichiers Javascript chargés (ceux par défaut et ceux ajoutés via `LSsession :: addJScript()`).

- Autres fichiers remplacés :
 - `blank.tpl` remplacé par `base.tpl`
 - `empty.tpl` remplacé par `base_connected.tpl`
 - `accueil.tpl` remplacé par `homepage.tpl`

Pour identifier les fichiers concernés, vous pouvez utiliser la commande suivante :

```
grep -Er '(accueil|blank|empty|top|bottom)\.tpl' /etc/ldapsaisie/local/
```

3.2.4.2. Fichiers templates fournis par défaut :

Vérifier les modifications des fichiers templates fournis avec l'application et que vous auriez personnalisés. Pour cela, vous pouvez utiliser la commande suivante :

```
for i in $( ls /etc/ldapsaisie/local/templates/* )
do
  default_file="/usr/share/ldapsaisie/templates/default/${ basename "$i" }"
  [ ! -e "$default_file" ] && continue
  vi -d $default_file $i
done
```

Note : Une attention particulière doit être portée aux fichiers `login.tpl` et `recoverpassword.tpl` qui ont particulièrement changé.

3.2.4.3. Corriger les URL des images :

```
../../images/default/find.png devient ../image/find
```

Pour identifier les fichiers concernés, vous pouvez utiliser les commandes suivantes :

```
grep -Er 'images' /etc/ldapsaisie/local/templates
grep -Er '\.(png|gif|jpg)' /etc/ldapsaisie/local/templates
```

3.2.4.4. Le cas de variable de template `{LSsession_css}` et `{LSsession_js}` :

Note : Ceci est déjà géré si vous étendez bien vos templates du fichier `base.tpl` (pour les pages non-connectées) ou `base_connected.tpl` (pour les pages connectées).

- `{LSsession_css}` doit être remplacé par `{include file='ls:css.tpl'}`

- `{LSsession_js}` doit être remplacé par `{include file='ls:js.tpl'}`

3.2.5. Tous les fichiers : Modification des URLs

- `view.php` :
 - **page recherche** : `view.php?LSobject=LSpeople` **devient** `object/LSpeople`
 - **page d'un objet** : `view.php?LSobject=LSpeople&dn=$dn` **devient** `object/LSpeople/$dn`
- `addon_view.php` : `addon_view.php?LSaddon=ee&view=copyContract` **devient** `addon/ee/copyContract`
- `index_ajax.php` :

- **Pour les méthodes Ajax de classes :**

```
var data = {
  template: 'LSformElement_eetelephone',
  action: 'make_call',
  telephoneNumber: tel,
  name: name,
};
new Request({url: 'index_ajax.php', data: data, onSuccess: ...});
```

Devient :

```
var data = {
  telephoneNumber: tel,
  name: name,
};
new Request({url: 'ajax/class/LSformElement_eetelephone/make_call', data: data, onSuccess: ...});
```

- **Pour les méthodes Ajax d'addon :**

```
var data = {
  addon: 'asterisk',
  action: 'LSasterisk_make_call',
  telephoneNumber: tel,
  name: name,
  nocache: new Date().getTime()
};
new Request({url: 'index_ajax.php', data: data, onSuccess: ...});
```

Devient :

```
var data = {
  telephoneNumber: tel,
  name: name,
  nocache: new Date().getTime()
};
```

```
new Request({url: 'ajax/addon/asterisk/LSasterisk_make_call', data: data, onSuccess: ..
```

- `global_search.php` : `global_search.php?refresh` **devient** `search?refresh`
- `index.php` : `index.php?LSsession_recoverPassword` **devient**
`index?LSsession_recoverPassword`
- `create.php` : `create.php?LSobject=LSpeople` **devient** `object/LSpeople/create`
- `modify.php` : `modify.php?LSobject=LSpeople&dn=$dn` **devient**
`object/LSpeople/$dn/modify`
- `import.php` : `import.php?LSobject=LSpeople` **devient** `object/LSpeople/import`
- `remove.php` : `remove.php?LSobject=LSpeople&dn=$dn` **devient**
`object/LSpeople/$dn/remove`

Avec validation : `remove.php?LSobject=LSpeople&dn=$dn&valid` **devient**
`object/LSpeople/$dn/remove?valid`

- `select.php` : `select.php?LSobject=LSpeople` **devient** `object/LSpeople/select`
- `custom_action.php` :
`custom_action.php?LSobject=LSpeople&dn=$dn&customAction=$customAction` **devient**
`object/LSpeople/$dn/customAction/$customAction`

Avec validation :
`custom_action.php?LSobject=LSpeople&dn=$dn&customAction=$customAction&valid`
devient `object/LSpeople/$dn/customAction/$customAction?valid`

- `custom_search_action.php` :
`custom_search_action.php?LSobject=LSpeople&customAction=$customAction` **devient**
`object/LSpeople/customAction/$customAction`

Avec validation :
`custom_search_action.php?LSobject=LSpeople&customAction=$customAction&valid`
devient `object/LSpeople/customAction/$customAction?valid`

Pour identifier les fichiers concernés, vous pouvez utiliser la commande suivante :

```
grep -Er '(index|global_search|view|select|create|modify|import|remove|index_ajax|custom_ac
```

Chapitre 4. Configuration

La configuration du projet est située principalement dans le dossier 'conf'. Les exceptions seront détaillées par la suite.

Avertissement

Toute la configuration du projet se fait par l'intermédiaire de fichiers définissant des variables PHP (<http://www.php.net/>) dont les valeurs sont utilisées par le programme. Ceci signifie que la syntaxe de ces fichiers doit être valide avec l'interpréteur PHP (<http://www.php.net/>) utilisé.

4.1. Configuration globale

La plus grande partie de la configuration globale se trouve dans le fichier `config.inc.php`.

```
Structure
// Variables globales
$GLOBALS['LSconfig'] = array(
    // Variables globales
);

// Variables et constantes indépendantes
$var1 = 'val1'
$var2 = 'val2'
...
define('CONST1', 'val1')
define('CONST2', 'val2')
...
?>
```

4.1.1. Variables globales

NetLDAP2

Chemin vers la librairie PEAR Net_LDAP2 (http://pear.php.net/package/Net_LDAP2).

```
/usr/share/php/Net/LDAP2.php
```

Smarty

Chemin vers le moteur de template Smarty (<http://www.smarty.net/>).

```
/usr/share/php/smarty/libs/Smarty.class.php
```

public_root_url

URL publique de la racine web de l'application. Il peut s'agir d'une URL relative bien qu'une URL absolue soit préférable, notamment pour éviter l'auto-détection de celle-ci lorsque nécessaire (lien dans un e-mail par exemple. Par défaut : /.)

Important : Il est indispensable que ce paramètre soit configuré en adéquation avec votre environnement pour que l'application fonctionne correctement (notamment en cas de déploiement dans un sous-dossier ou encore dans le cadre d'un accès à l'application au travers un *reverse-proxy*).

lang

Paramètre utilisé pour l'internationalisation : code de la langue.

fr_FR
en_US

encoding

Encodage de caractère.

UTF8

ldap_servers

Configuration des serveurs LDAP. Voir section concernée.

4.1.1.1. Préférences globales

Important : Les variables globales suivantes ont une action globale, mais non-prioritaire sur le comportement de l'application. Il peut être redéfini pour chacun des serveurs LDAP.

cacheLSprofiles

Activation/Désactivation de la mise en cache des profils des utilisateurs connectés (LSprofiles).

Valeurs possibles : True ou False

Valeur recommandée : True

Valeur par défaut : `False`

`cacheSubDn`

Activation/Désactivation de la mise en cache des niveaux de connexion (`subDn`) dans l'annuaire.

Valeurs possibles : `True` ou `False`

Valeur recommandée : `True`

Valeur par défaut : `False`

`cacheSearch`

Activation/Désactivation de la mise en cache du résultat des recherches dans l'annuaire.

Valeurs possibles : `True` ou `False`

Valeur recommandée : `True`

Valeur par défaut : `False`

`globalSearch`

Activation/Désactivation de la recherche globale dans l'annuaire.

Valeurs possibles : `True` ou `False`

Valeur par défaut : `True`

`keepLSsessionActive`

Activation/Désactivation du maintien de la `LSsession` active.

Valeurs possibles : `True` ou `False`

Valeur par défaut : `False`

4.1.1.2. Configuration des serveurs LDAP

Cette section décrit le tableau de configuration des différents serveurs LDAP utilisés par l'application. Ce tableau contient lui-même un tableau par serveur LDAP.

```

Structure...
$GLOBALS['LSconfig'] = array(
    ...
    'ldap_servers' => array(
        array (
            'name' => [nom de l'annuaire],
            'ldap_config'=> array(
                // Définition des paramètres de connexion à l'annuaire
            ),
            'useUserCredentials' => [boolean],
            'LSauth' => array (
                'method' => [LSauth method],
                'api_method' => [LSauth method],
                'LSubjects' => array(
                    '[object type 1]',
                    '[object type 2]' => array(
                        'filter' => '[LDAP filter]',
                        'password_attribute' => '[attribute name]',
                        'web_access' => [booléen],
                        'api_access' => [booléen],
                    )
                )
            ),
            'LSprofiles' => array (
                // Définition des LSprofiles
            ),
            'cacheLSprofiles' => [boolean],
            'cacheSearch' => [boolean],
            'globalSearch' => [boolean],
            'LSaccess' => array (
                [Type LSubject 1],
                [Type LSubject 2],
                ...
            ),
            'subDn' => array(
                // Définition des sous-niveaux de l'annuaire
            ),
            'subDnLabel' => [nom des sous-niveaux],
            'recoverPassword' => array(
                // Définition des paramètres de configuration de la récupération de mot de passe
            ),
            'defaultView' => [view],
            'emailSender' => [email],
            'keepLSsessionActive' => [booléen]
        )
        ...
    );

```

...

Paramètres de configuration

name

Le nom d’affichage de ce serveur Ldap (utilisé lorsque plusieurs serveur LDAP sont déclarés).

ldap_config

Informations de connexion au serveur LDAP. Ces informations sont structurées selon les attentes de la librairie Net_LDAP2 (http://pear.php.net/package/Net_LDAP2). Plus d’informations (<http://pear.php.net/manual/fr/package.networking.net-ldap.connecting.php>)

useUserCredentials

Booléen définissant si il faut utiliser les identifiants de l’utilisateur pour se connecter à l’annuaire (*false* par défaut). Si cette option est activée, la connexion à l’annuaire LDAP sera établie avec la configuration fournie dans le paramètre *ldap_config* en écrasant les informations de connexion (*binddn* et *bindpwd*) par ceux de l’utilisateur. Si l’utilisateur n’est pas encore connecté, la connexion sera établie sans modifier la configuration fournie.

LSprofiles

Définition des profils d’utilisateurs se connectant à l’annuaire. Voir la section concernée.

LSauth

Ce tableau définit les paramètres d’authentification à l’application.

Paramètres de configuration de l’authentification

method

Nom de la méthode d’authentification LSauthMethod. Exemple : pour utiliser la classe LSauthMethod_HTTP, la valeur de ce paramètre sera HTTP. *Paramètre facultatif, méthode par défaut : basic.*

api_method

Nom de la méthode d’authentification LSauthMethod à utilisée lors d’une connexion à l’API. Exemple : pour utiliser la classe LSauthMethod_HTTP, la valeur de ce paramètre sera HTTP. *Paramètre facultatif, méthode par défaut : HTTP.*

Avertissement

Toutes les LSauthMethod ne supportent pas forcément le mode API.

LSobjects

Tableau listant les types LSobjects pouvant se connecter à l'application. Les valeurs de ce tableau peuvent être un nom de type d'objet ou bien tableau détaillant les paramètres de connexion de ce type d'objet.

Paramètres de configuration d'un type d'objet

filter

LSformat du filtre de recherche de l'utilisateur à sa connexion. Ce format sera composé avec l'identifiant fourni par l'utilisateur. Cela peut par exemple permettre à l'utilisateur de se connecter en fournissant son login ou son email comme identifiant. Exemple de valeur : `(|(uid=%{user})(mail=%{user}))`. *Paramètre facultatif, filtre par défaut composé à l'aide de l'attribut RDN.*

password_attribute

Nom de l'attribut stockant le mot de passe de ce type d'LSobject. *Paramètre facultatif, valeur par défaut : `userPassword`.*

Note : C'est cet attribut de l'utilisateur qui sera modifié par la fonctionnalité de récupération de mot de passe.

web_access

Permet de définir si ce type d'objet à le droit d'utiliser l'interface web (facultatif, par défaut : `True`).

api_access

Permet de définir si ce type d'objet à le droit d'utiliser l'API (facultatif, par défaut : `False`).

allow_multi_match

Booléen permettant de définir si un doublon d'identifiant utilisateur est autorisé. Si c'est le cas et lorsqu'un identifiant fourni par l'utilisateur a sa connexion a permis de trouver plus d'un utilisateur possible correspondant, l'application tentera de déterminer lequel de ces utilisateurs correspond à la tentative d'authentification. La méthodologie employée dépendra de la LSauthMethod configurée. Par exemple, la LSauthMethod `basic` tentera de s'identifier avec le mot de passe. Dans tous les cas, si cette méthode n'a pas permis d'identifier un seul utilisateur, l'authentification échouera. *Paramètre facultatif, valeur par défaut : `Faux`.*

cacheLSprofiles

Activation/Désactivation de la mise en cache des LSprofiles des utilisateurs connectés à ce serveur.

Valeur par défaut : *valeur de la variable globale du même nom*

cacheSearch

Activation/Désactivation de la mise en cache du résultat des recherches sur ce serveur.

Valeur par défaut : *valeur de la variable globale du même nom*

globalSearch

Activation/Désactivation de la recherche globale sur ce serveur en particulier. Par défaut, la valeur du paramètre `global globalSearch` est utilisée.

Valeur par défaut : *valeur de la variable globale du même nom*

LSaccess

Définition des types d'LSobjects devant apparaître dans le menu de l'interface.

Important : Ce paramètre n'est utilisé que pour les annuaires n'ayant pas de sous-niveaux (subDn).

subDn

Définition des sous-niveaux de connexion à l'annuaire. Voir section concernée.

Important : Ce paramètre remplace le paramètre LSaccess dans le cas d'un annuaire multi-niveaux.

subDnLabel

Définition du label utilisé pour qualifier les sous-niveaux de connexion.

Important : Ce paramètre est utile uniquement dans le cas d'un annuaire multi-niveaux.

recoverPassword

Définition des paramètres de la récupération de mot de passe. Voir la section concernée.

defaultView

Définition de la vue par défaut de l'application. Par défaut, une page blanche est affichée et il est possible de définir à l'aide de ce paramètre la vue qui s'affichera. Ce paramètre peut prendre comme valeur :

- `SELF` pour la vue *Mon compte*
- Le nom d'un `LSubject` pour afficher la liste de ce type d'objet
- Le nom d'une vue d'un `LSaddon` au format `[addon]::[viewId]` pour afficher cette vue

emailSender

Adresse mail utilisée par `LdapSaisie` pour envoyer des e-mails en relation avec cet annuaire. Cette adresse est celle utilisée par défaut. L'adresse utilisée peut également être configurée dans le contexte de configuration du module devant envoyer des e-mails.

keepLSsessionActive

Activation/Désactivation du maintien de la `LSsession` active.

Valeurs possibles : *True* ou *False*

Valeur par défaut : *valeur de la variable globale du même nom*

4.1.1.2.1. Profils d'utilisateurs

Cette section décrit la manière dont sont définis les profils d'utilisateurs se connectant à l'interface appelés *LSprofile*. Il est possible d'attribuer un profil à l'utilisateur connecté sur tout ou partie de l'annuaire LDAP.

4.1.1.2.1.1. Profils d'utilisateurs par défaut

Il existe des profils d'utilisateurs par défaut, non liée à la configuration de l'application:

user

Tous les utilisateurs connectés à l'utilisateur. Ce `LSprofile` est valide sur l'ensemble de l'annuaire.

self

L'utilisateur connecté sur son objet correspondant dans l'annuaire. Ce `LSprofile` est utile pour donner des droits à l'utilisateur sur lui-même.

nom du type de l'objet connecté

Un `LSprofile` du nom du type d'objet utilisateur connecté est automatiquement ajouté à l'utilisateur. Ainsi, si l'utilisateur connecté est un `LSubject` `LSpeople` par exemple, il aura le `LSprofile`

LSpeople sur tous l'annuaire. Ce LSprofile est utile pour donner des droits à tous un type d'objets pouvant se connecter à l'application (par exemple, tous les utilisateurs applicatifs).

4.1.1.2.1.2. Profils d'utilisateurs personnalisés

Il est possible de définir autant de profils d'utilisateurs que l'on souhaite. Pour chaque profil d'utilisateur personnalisé, il faudra définir dans quelles parties de l'annuaire ce profil existe (Exemple : les administrateurs de groupes existent uniquement dans la branche de l'annuaire stockant les groupes). Enfin pour chaque partie de l'annuaire, il faudra définir la manière d'identifier si l'utilisateur qui se connecte appartient à ce profil.

Structure...

```
'LSprofile' => array (
  [nom d'un LSprofile] => array (
    [label] => [label du LSprofile],
    [basedn] => [dn utilisateur],
    [autre basedn] => array (
      [dn d'un utilisateur] => NULL,
      [autre dn] => array ( // via un listage de l'attribut d'un objet
        'attr' => [nom de l'attribut clé de l'objet],
        'attr_value' => [format de la valeur de l'attribut clé],
        'LSubject' => [nom du type LSubject de l'objet]
      )
    ),
  'LSubjects' => array ( // via une liste d'objet sur lequel l'utilisateur a des pouvoirs
    [nom du LSubject] => array (
      'attr' => [nom de l'attribut clé],
      'attr_value' => [format de la valeur de l'attribut clé],
      // ou
      'filter' => [format du filtre de recherche],

      'basedn' => [basedn de recherche],
      'params' => [configuration de la recherche]
    ),
    [nom quelconque] => array (
      'filters' => array(
        array(
          'LSubject' => [nom du LSubject],
          'attr' => [nom de l'attribut clé],
          'attr_value' => [format de la valeur de l'attribut clé],
          // ou
          'filter' => [format du filtre de recherche],

          'basedn' => [basedn de recherche],
          'params' => [configuration de la recherche]
        )
      )
    ),
  ),
  ...
)
```

```

    )
  ),
  ...
),
...

```

Le paramètre `LProfiles` est un tableau associatif contenant, en valeur clé, le nom d'un *LProfile* et en valeur associée, la configuration nécessaire pour déterminer si l'utilisateur connecté appartient à ce *LProfile* pour tout ou partie de l'annuaire.

Dans chaque configuration de *LProfile*, il est possible d'identifier l'appartenance ou non de l'utilisateur connecté de deux manières :

- Pour une branche de l'annuaire donnée (*basedn*) : en listant les utilisateurs appartenant à ce *LProfile* pour tous les objets de la branche. Il sera possible de lister les utilisateurs dont on connaît le *DN* ou de lister les utilisateurs appartenant à une liste stockée dans l'annuaire (par exemple la liste des membres d'un groupe).
- Liste des *DNs* d'utilisateurs :

```

Structure...
'LProfile' => array (
  [nom du LProfile] => array (
    [basedn] => [dn utilisateur],
    // ou si plusieurs DNs
    [autre basedn] => array (
      [dn d'un utilisateur] => NULL,
      [dn d'un utilisateur 2] => NULL
    ),
    ...
  ),
  ...
),
...

```

Explication : Pour un *LProfile* et un *basedn* donnés, on définit l'utilisateur appartenant au *LProfile* en donnant son *DN*. Si on souhaite lister plusieurs utilisateurs, on utilise un tableau associatif dans lequel les clés sont les *DNs* des utilisateurs et les valeurs associées sont toutes *NULL*.

- Liste d'utilisateurs stockée dans l'annuaire :

```

Structure...
'LProfile' => array (
  [nom du LProfile] => array (
    [basedn] => array (
      [DN d'un objet] => array (
        'attr' => [nom de l'attribut clé de l'objet],
        'attr_value' => [format de la valeur de l'attribut clé],
        'LSubject' => [nom du type LSubject de l'objet]
      )
    ),
  ),

```

```

    ...
  ),
  ...

```

Explication : Pour un *LSprofile* et un *basedn* donnés, on liste les utilisateurs du *LSprofile* référencés dans l'attribut `attr` de l'objet de type `LSubject` et selon le format de valeur décrit dans `attr_value`.

- Pour un type de *LSubject* donné : en listant les objets pour lesquels l'utilisateur aura les droits du *LSprofile*. Il sera possible, à travers une recherche paramétrable dans l'annuaire, de lister les objets pour lesquels l'utilisateur appartiendra au *LSprofile*.

```

Structure...
'LSprofile' => array (
  [nom d'un LSprofile] => array (
    'LSubjects' => array ( // via un liste d'objet pour lequel l'utilisateur
                        // appartient au LSprofile
    [nom du LSubject] => array (
      'attr' => [nom de l'attribut clé],
      'attr_value' => [format de la valeur de l'attribut clé],
      // or
      'filter' => [format du filtre de recherche],

      'basedn' => [basedn de recherche],
      'params' => [configuration de la recherche]
    ),
    array (
      'filters' => array(
        array(
          'LSubject' => [nom du LSubject],
          'attr' => [nom de l'attribut clé],
          'attr_value' => [format de la valeur de l'attribut clé],
          // ou
          'filter' => [format du filtre de recherche],

          'basedn' => [basedn de recherche],
          'params' => [configuration de la recherche]
        ),
      ),
    ),
  ),
  ...
)
),
...
),
...

```

Explications : Dans la configuration d'un *LSprofile*, la valeur clé *LSubjects* signifie qu'on est dans un cas de la délégation de droits sur des types d'*LSubject*. Dans ce tableau associatif, il est possible de

définir un ou plusieurs types de LSubject pour lesquels on délègue des droits via des recherches simples ou enchaînées. Le fonctionnement simple consiste à partir de l'objet de l'utilisateur et à générer un filtre de recherche sur un type de LSubject. Le fonctionnement enchaîné consiste à faire une première recherche à partir de l'objet de l'utilisateur puis à recommencer à partir des objets trouvés en construisant une liste de filtres de recherche pour chaque objet qui seront combinés via l'opérateur booléen *ou*.

Pour configurer une délégation de type simple on mettra le nom du LSubject dans la clé du tableau et dans la valeur un tableau définissant la recherche. Il est possible de ne pas utiliser la clé du tableau comme nom du LSubject grâce à la clé de configuration *LSubject*.

Pour configurer une délégation de type enchaîné on pourra utiliser n'importe quelle valeur unique pour la clé du tableau et pour la valeur un tableau contenant une unique clé *filters*. La valeur associée à cette clé est celle d'une délégation de type simple où la clé *LSubject* est devenue obligatoire.

Cette configuration contient les paramètres d'une ou plusieurs recherches dans l'annuaire en considérant que l'utilisateur connecté aura les droits du LSprofile sur les objets retournés. Les paramètres de la recherche sont :

LSubject

C'est le nom du LSubject recherché. (*Paramètre facultatif pour une délégation de type simple*)

attr

Nom de l'attribut des LSubjects contenant une valeur clé qui permettra d'identifier l'utilisateur comme ayant droit.

attr_value

Le format de la valeur clé prise par l'attribut *attr*. Ce format est composé à partir des données de l'objet de l'utilisateur connecté. Voir le paragraphe Format paramétrable pour plus d'informations sur l'écriture du format.

filter

Ce paramètre remplace les paramètres *attr* et *attr_value*. Il est possible ici d'écrire directement le format paramétrable du filtre recherche dans l'annuaire. Ce filtre sera automatiquement agrémenté des conditions sur l'attribut *objectclass*. Voir le paragraphe Format paramétrable pour plus d'informations sur l'écriture du format.

basedn

C'est le *basedn* de la recherche. Il est possible ainsi de la limiter sur les LSubjects d'une branche précise de l'annuaire. (*Paramètre facultatif*)

params

C'est un tableau associatif contenant les paramètres étendus de la recherche. Voir le paragraphe Paramètres étendus des recherches dans l'annuaire pour plus de détails. (*Paramètre facultatif*)

Par ailleurs, il est possible d'attribuer un label plus explicite à chaque *LSprofile* à l'aide de la clé `label`. Ce label sera utilisé pour faire référence au *LSprofile* lorsque nécessaire. (*Paramètre facultatif*)

4.1.1.2.2. Sous-niveaux de connexion

Cette section décrit la manière de définir des sous-niveaux de connexion à l'annuaire (*subDn*). Le concept de sous-niveau de connexion sert à déclarer les niveaux logiques de l'annuaire. Par exemple, dans un annuaire dans lequel sont stockés des objets concernant plusieurs organisations et que celles-ci se distinguent grâce à la présence d'une séparation dans l'arbre, il sera alors possible de définir des sous-niveaux de connexion pour chacune des organisations.

Exemple d'arborescence d'annuaire utilisant le concept de sous-niveaux correspondant à des sociétés

```
| - o=ls
| | - ou=companies
| | | - ou=company1
| | | | - ou=people
| | | | - ou=groups
| | | - ou=company2
| | | | - ou=people
| | | | - ou=groups
| | - ou=people
| | - ou=groups
```

Explications : Il est possible dans cet exemple de définir des sous-niveaux de connexion correspondants aux sociétés. Dans chacune de ces sociétés, on retrouve les *OU* correspondant au type d'*LSobjets*. Lors de la connexion à l'interface, l'utilisateur devra choisir dans quel sous-niveau de l'annuaire il souhaite se connecter. Une fois connecté, l'utilisateur manipulera uniquement les objets du sous-niveau de l'annuaire dans lequel il se trouve. Il lui sera également possible de changer de sous-niveau de connexion à travers l'interface : une liste déroulante est disponible pour cela dans le menu.

Il existe deux manières de déclarer des sous-niveaux de connexion à l'annuaire :

- En déclarant manuellement un *subDn* de l'annuaire et en lui donnant un nom.
- En listant les *LSobjets* d'un type précis et en utilisant leurs données pour constituer le nom des sous-niveaux. Cette liste est constituée en effectuant une recherche dans l'annuaire. Il est possible de définir un *basedn* particulier pour cette recherche.

Pour chacune de ces méthodes on définira également les types d'*LSobjets* qui sont présents dans cette branche de l'annuaire.

```
Structure...
'subDn' => array(
```

```

// Déclaration manuelle
'[Nom du sous-niveau]' => array(
  'dn' => '[basedn du sous-niveau]',
  'nologin' => true, // Désactive la connection dans ce subDn
  'LSubjects' => array( // Liste des types d'LSobjets présents dans le sous-niveau
    [LSubject1],
    [LSubject2],
    ...
  )
),
// Liste de LSobjets
'LSubject' => array(
  '[type d'LSobjet]' => array( // le type d'LSobjet à lister
    'basedn' => '[basedn]', // Le basedn de la recherche
    'displayValue' => '[format]', // Format du nom des sous-niveaux
    'nologin' => true, // Désactive la connection dans ces subDn
    'onlyAccessible' => True, // Pour que seul les LSobjet accessible à l'utilisateur soit
    'LSubjects' => array( // Liste des types d'LSobjets présents dans les sous-niveaux
      [LSubject1],
      [LSubject2],
      ...
    )
  )
)
),
...

```

4.1.1.2.3. Récupération de mot de passe

Cette section décrit la manière de configurer la récupération de mot de passe par les utilisateurs. Le mécanisme de récupération de mot de passe fonctionne en deux parties :

- Dans un premier lieu, l'utilisateur ayant perdu son mot de passe accède à l'interface de récupération à partir de la page de connexion. L'interface lui demande de saisir son identifiant et éventuellement de sélectionner le serveur LDAP concerné. Une fois ces informations saisies, une recherche de l'utilisateur est effectuée dans l'annuaire et si celui-ci est trouvé, la valeur de l'attribut `recoveryHashAttr` de l'objet est alors redéfinie avec une valeur aléatoire.

Un mail est ensuite envoyé à l'utilisateur en utilisant la première valeur de l'attribut `mailAttr` comme adresse. Ce mail est formé à partir des paramètres du tableau associatif `recoveryHashMail`. Celui-ci doit contenir le sujet du mail dans `subject` et le corps du message dans `msg`. Ces deux informations sont des *formats paramétrables* composés avec, comme valeur clé, l'URL de retour à laquelle l'utilisateur devra se rendre pour accéder à la seconde étape de la récupération de son mot de passe.

- L'utilisateur doit donc se rendre sur l'interface par l'intermédiaire de l'URL qui lui aura été fournie dans le mail de l'étape précédente. Cette URL contient la valeur de l'attribut `recoveryHashAttr` précédemment définie. A partir de cette information, une recherche est effectuée dans l'annuaire pour retrouver l'utilisateur correspondant.

Si l'utilisateur est retrouvé, un nouveau mot de passe lui est généré en utilisant les paramètres de configuration éventuellement définis dans la configuration HTML de l'attribut "mot de passe". Pour avoir plus d'information sur ces paramètres, consulter la documentation du type d'attribut HTML *LSattr_html_password*. L'attribut `recoveryHashAttr` est quant à lui supprimé.

Ensuite, un mail est composé à partir des paramètres du tableau associatif `newPasswordMail` et est envoyé à l'utilisateur. Ce tableau doit contenir le sujet du mail dans `subject` et le corps du message dans `msg`. Ces deux informations sont des *formats paramétrables* composés avec, comme valeur clé, le nouveau mot de passe de l'utilisateur.

```
Structure...
'recoverPassword' => array(
  'mailAttr' => '[attribut mail]',
  'recoveryHashAttr' => '[attribut hash]',
  'recoveryEmailSender' => '[adresse mail utilisée par LdapSaisie pour l'envoi des mails]',
  'recoveryHashMail' => array( // 1er mail : avec l'URL pour l'accès à la 2nde partie
    'subject' => '[sujet du mail]',
    'msg' => "[message contenant le mot clé %{url}]"
  ),
  'newPasswordMail' => array( // 2nd mail : avec le mot de passe
    'subject' => '[sujet du mail]',
    'msg' => "[message contenant le mot clé %{mdp}]"
  )
),
...
```

4.1.2. Variables et constantes indépendantes

LS_THEME

Constante déterminant le nom du theme utilisé.

Valeur par défaut : *default*

LS_TEMPLATES_DIR

Constante déterminant le chemin du dossier des templates.

Valeur par défaut : *templates*

LS_IMAGES_DIR

Constante déterminant le chemin du dossier des images.

Valeur par défaut : *images*

LS_CSS_DIR

Constante déterminant le chemin du dossier des CSS.

Valeur par défaut : *css*

LSdebug

Variable booléenne déterminant si le débogage à l'écran est activé.

\$GLOBALS['LSlog']

Variable permettant de configurer la journalisation de l'application. Voir section concernée.

NB_LSOBJECT_LIST

Constante déterminant le nombre d'objet affichés par page de résultat de recherche.

NB_LSOBJECT_LIST_SELECT

Constante déterminant le nombre d'objet affichés par page de résultat de recherche dans une fenêtre *LSelect*.

\$GLOBALS['NB_LSOBJECT_LIST_CHOICES']

Variable permettant de configurer la liste des choix proposés à l'utilisateur pour le nombre maximum d'objets affichés par page de résultat de recherche.

MAX_SEND_FILE_SIZE

Constante déterminant la taille maximale d'un fichier envoyé à travers les formulaires.

\$GLOBALS['defaultJSscripts']

Tableau déterminant les fichiers Javascript à charger sur toute les pages.

\$GLOBALS['defaultCSSfiles']

Tableau déterminant les fichiers CSS à charger sur toute les pages. Ces fichiers seront chargés dans l'ordre et en dernier permettant de surcharger tous paramètres de style.

4.1.2.1. Configuration de la journalisation

Cette section décrit le tableau de configuration de la journalisation de l'application.

Structure...

```

$GLOBALS['LSlog'] = array(
    'enable' => [booléen],
    'level' => '[niveau]',
    'handlers' => array(
        '[handler 1]',
        array (
            'handler' => [handler 2],
            'enabled' => [booléen],
            'level' => '[niveau]',
            'loggers' => array('logger1', [...]),
            'excluded_loggers' => array('logger2', [...]),
            'format' => '[LSformat]',
            'cli_format' => '[LSformat]',
            'datetime_prefix' => [booléen],
            'datetime_format' => '[format date()]',
            // Autres paramètres propre à ce handler
            [...]
        ),
        [...]
    ),
    'loggers' => array (
        'logger1' => array (
            'level' => 'DEBUG',
        ),
        'logger2' => array (
            'enabled' => false,
        ),
        [...]
    );
);
...

```

Paramètres de configuration

enable

Booléen permettant d'activer ou désactiver complètement la journalisation. Par défaut : `False`

level

Ce paramètre définit le niveau minimum de la journalisation : tous les messages des niveaux inférieurs ne seront pas inclus dans le journal de l'application. Les niveaux de journalisation gérés par l'application sont (dans l'ordre du plus petit au plus grand) :

- TRACE
- DEBUG
- INFO
- WARNING
- ERROR
- FATAL

handlers

Tableau permettant de configurer les *handlers* de la journalisation. Chaque *handler* gère les messages journalisés d'une manière qui lui est propre.

Plusieurs *handlers* peuvent être configurés en même temps (y compris plusieurs *handlers* du même type).

Ce tableau peut contenir simplement le nom du type de *handler* à utiliser ou bien des tableaux configurant un à un chacun des *handlers*. Dans ce second cas, la structure de la configuration d'un *handler* est la suivante :

```
Structure...
array(
  'handler' => [type],
  'level' => '[niveau]',
  'loggers' => array('logger1', [...]),
  'excluded_loggers' => array('logger2', [...]),
  'format' => '[LSformat]',
  'cli_format' => '[LSformat]',
  'datetime_prefix' => [booléen],
  'datetime_format' => '[format date()]',
  // Autres paramètres propre à ce handler
  [...]
)
...
```

Paramètres de configuration d'un handler

handler

Type du *handler* (voir ci-dessous).

level

Ce paramètre définit le niveau minimum de la journalisation spécifique à cet *handler*. Si ce paramètre est omis, le niveau global sera utilisé. Les valeurs possibles de ce paramètre sont les mêmes que pour le paramètre `$GLOBALS['LSlog']['level']` .

enabled

Booléen permettant d'activer ou désactiver cet *handler* (paramètre facultatif, par défaut : `True`).

loggers

Liste exhaustive des composants dont les messages doivent être traités par ce handler (paramètre facultatif, par défaut : tous les composants).

excluded_loggers

Liste exhaustive des composants dont les messages ne doivent pas être traités par ce handler (paramètre facultatif, par défaut : aucun composant).

format

LSformat des messages de cet journalisé par ce handler. Ce format est composé à partir des informations décrivent ci-dessous. Par défaut : `%{requesturi} - %{remoteaddr} - %{ldapserversname} - %{authuser} - %{logger} - %{level} - %{message}`

level

Le niveau du message.

message

Le message.

logger

Le composant ayant déclenché cette journalisation.

clibinpath

Le nom du script ayant déclenché cette journalisation (uniquement en cas d'exécution en ligne de commande).

requesturi

L'URL de la page courante (uniquement dans un contexte Web).

remoteaddr

L'adresse IP du client (uniquement dans un contexte Web).

ldapserversname

Le nom du serveur LDAP courant.

authuser

Le DN de l'utilisateur connecté (uniquement dans un contexte Web).

cli_format

LSformat des messages de cet journalisé par ce handler dans le cas d'une exécution en ligne de commande. Ce format est composé à partir des mêmes informations que le paramètre `format` (voir ci-dessus). Par défaut : `%{clibinpath} - %{logger} - %{level} - %{message}`

datetime_format

Booléen permettant de définir si le message doit être préfixé de la date et heure courante. La valeur par défaut dépend de l'handler (en règle générale, toujours actif sauf lorsque le canal de journalisation l'ajoute déjà).

datetime_format

Format de la date et heure lorsque celle-ci est ajoutée en préfixe du message (voir paramètre `datetime_format`). Le format correspond à celui attendu par la fonction `date()` de PHP (<http://www.php.net/>). Consultez la documentation officielle (<http://php.net/date>) pour plus de détails (Par défaut : `Y/m/d H:i:s`).

Il existe plusieurs types d'*handlers* gérés par l'application :

file

Journalisation dans un simple fichier texte. Le chemin du fichier peut être configuré via le paramètre `path`. Si ce paramètre est omis, le chemin du fichier par défaut est soit la valeur de la variable `$GLOBALS['LSlog']['filename']` (pour la rétro-compatibilité avec les anciennes versions d'LdapSaisie) ou à défaut : `tmp/LS.log`.

syslog

Journalisation via le service *syslog*. Il est possible de configurer une priorité systématique pour les messages journalisés. À défaut, la priorité sera déterminée automatiquement en fonction du niveau du message. Les valeurs possibles de ce paramètre sont : `EMERG`, `ALERT`, `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, `DEBUG`

system

Journalisation via le gestionnaire d'erreurs PHP. Cet *handler* utilise la fonction PHP `error_log`. Pour plus d'informations sur comment configurer le gestionnaire d'erreurs PHP, consulter la documentation officielle (https://www.php.net/error_log).

email

Journalisation via l'envoi d'un email : chaque message journalisé déclenchera l'envoi d'un email au destinataire configuré. L'adresse email du destinataire peut-être configurée via le paramètre `recipient`.

Note : Il est conseillé d'utiliser ce type d'*handler* avec un niveau minimum de journalisation important (`FATAL` recommandé) pour ne pas déclencher un nombre trop important d'envois d'emails.

loggers

Tableau permettant de configurer la journalisation composant par composant. Chaque composant peut avoir son propre *logger* ce qui permet alors, par exemple, de configurer le niveau de log spécifiquement pour ce composant.

Le nom des composant correspond en général au nom de la classe PHP (<http://www.php.net/>) correspondante, ou bien encore le nom d'une commande (lors d'une exécution en ligne de commande).

Note : Par défaut, le nom du composant ayant déclenché un message journalisé est affiché juste avant le niveau de log.

Paramètres de configuration d'un logger

enabled

Booléen permettant de désactiver complètement les logs du composant (par défaut: `True`).

level

Niveau de log spécifique pour ce composant (par défaut: le niveau de log global).

4.1.3. Format paramétrable

Un *format paramétrable* est une chaîne de caractères contenant des mots clés formés comme dans l'exemple suivant :

```
%{[nom du mot clé][:A][:B][! ou _][~][%]}
```

Le nom du mot clé peut contenir des lettres de "a" à "z", de "A" à "Z" et des chiffres de 0 à 9. Ces mots clés seront remplacés par les valeurs passées en paramètres et liées au contexte d'utilisation. Les paramètres `:A` et `:B` permettent d'extraire une partie de la chaîne complète avant la substitution.

Le paramètre `A` correspond, lorsque `B` n'est pas défini, au nombre maximum de caractères à extraire de la chaîne de substitution. `A` doit être un entier dont le signe influ, comme expliqué ci-dessous :

- Si `A` est positif, les `A` premiers caractères de la chaîne de substitution seront extraits.
- Si `A` est négatif, les `|A|` derniers caractères de la chaîne de substitution seront extraits.

Lorsque le paramètre `B` est défini, `A` correspond au rang du premier caractère à partir duquel la chaîne de substitution sera découpée et `B` le nombre maximum de caractères à extraire. Le signe de `B` influera comme expliqué dans le premier cas. Si `B` vaut zéro, la totalité de la longueur de la chaîne sera retournée en tenant compte de `A` pour le rang du premier caractère.

Il existe par ailleurs des paramètres permettant de modifier la valeur de substitution avant son utilisation :

- Les paramètres `!` ou `_` permettent respectivement de forcer la mise en majuscule ou en minuscule ;

- Le paramètre `~` permet de forcer la suppression des accents ;
- Le paramètre `%` permet de protéger les caractères éligibles en entités HTML.

Important : Lorsque qu'une seule valeur clé est disponible pour la substitution, le nom du mot clé n'importe pas. Tous les mots clés trouvés dans le format seront remplacés par cette seule valeur.

4.1.4. Paramètres étendus des recherches dans l'annuaire

Les paramètres des recherches sont ceux supportés par Net_LDAP2 (http://pear.php.net/package/Net_LDAP2). Ces paramètres sont passés sous la forme d'un tableau associatif. Les paramètres supportés sont détaillés ci-dessous :

Nom	Description	Valeur par défaut
<code>scope</code>	Définition de l'étendue de la recherche : <ul style="list-style-type: none"> • <code>base</code> - Sur une entrée seulement • <code>one</code> - Sur les entrées immédiatement contenu par le <code>basedn</code> de la recherche • <code>sub</code> - Sur l'arbre entier 	<code>sub</code>
<code>sizelimit</code>	Le nombre maximum d'entrées retournées par la recherche.	0 (illimité)
<code>timelimit</code>	Le délai d'attente maximum de la réponse du serveur en secondes.	0 (illimité)
<code>attrsonly</code>	Si <i>vrai</i> , seuls les noms des attributs seront retournés.	<code>false</code>
<code>attributes</code>	Tableau contenant les noms des attributs que les entrées retournées peuvent contenir et que l'on souhaite récupérer.	<code>array()</code> (tous)

* Pour plus d'information sur le sujet, vous pouvez consulter la documentation officiel du projet Net_LDAP2 (http://pear.php.net/package/Net_LDAP2).

4.2. Configuration LSubject

Cette partie décrit la manière de configurer les différents types de LSubjects manipulés par LdapSaisie.

La configuration des LSubjects est stockée dans le dossier `/conf/LSubjects`. Dans ce dossier, on retrouve un fichier par type d'LSubject, nommé de la manière suivante :

```
config.LSubjects.[nom du type d'LSubject].php
```

Ce fichier contient la déclaration de la configuration du type d'LSubject qui est stocké dans la variable globale `$GLOBALS['LSubjects']`['[nom du type d'LSubject]'].

```
Structure...
$GLOBALS['LSubjects'] ['[nom du type d'LSubject]'] = array (
    'objectclass' => array(
        'objetclass1',
        'objetclass2',
        ...
    ),
    'filter' => '[filtre LDAP]',

    'rdn' => 'attr1',

    'LSaddons' => [LSaddon(s)],

    'container_dn' => 'ou=people',
    'generate_container_dn' => '[callable]',
    'container_auto_create' => array(
        // Information des configurations pour la création du conteneur du type d'LSubject
        // lors de la création nouveau subDn
    ),

    'disable_creation' => [boolean]',

    'before_modify' => 'function1',
    'after_modify' => 'function2',
    'after_create' => 'function3',
    'after_delete' => 'function4',

    'label' => 'objet1',
    'display_name_format' => '[format]',
    'displayAttrName' => '[boolean]',

    //Custom Actions
    'customActions' => array (
        // Configuration des customActions pour ce type d'objet
    ),

    // LSrelation
    'LSrelation' => array(
        // Configuration des LSrelations entre ce type d'objet et les autres
```

```

),

// LSform
'LSform' => array (
    // Configuration des formulaires de l'objet
), // fin LSform

// LSsearch
'LSsearch' => array (
    // Configuration des recherches de l'objet
), // fin LSsearch

'globalSearch' => [booleen],

// ioFormat
'ioFormat' => array (
    // Configuration des formats d'import/export de l'objet
),

// Attributs
'attrs' => array (
    // Configuration des attributs du type d'LSobjet
)
);
...

```

Paramètres de configuration

objectclass

La liste des *objectclass* des objets.

filter

Filtre de recherche LDAP applicable à tout les objets de ce type et qui sera utilisé lors de chaque recherche de ce type d'objet.

rdn

Nom de l'attribut correspondant au *RDN* des objets LDAP.

LSaddons

LSaddon(s) dont le type d'objet dépend. Ce peut être un tableau de chaînes de caractères ou une simple chaîne de caractères correspondant au(x) nom(s) du/des LSaddon(s) en dépendance.

container_dn

Élément pour construire le *basedn* de stockage de ce type d'objet. Par exemple, si le *basedn* de l'annuaire est `o=ls` et que les objets *utilisateurs* sont stockés dans la branche de l'annuaire `ou=people,o=ls`, alors *container_dn* devra valoir `ou=people`.

* Lorsque l'annuaire possède des *subDn*, les objets seront cherchés dans le *basedn* résultant de la concaténation du paramètre *container_dn*, d'une virgule et du *basedn* correspondant au *subDn* courant.

generate_container_dn

Callable (au sens PHP), utilisé pour générer la valeur du paramètre `container_dn` dynamiquement. Ce *callable* prend en paramètre l'objet `LSubject` à créer et retourne la valeur du paramètre `container_dn`.

container_auto_create

Tableau associatif contenant les paramètres de configuration nécessaires à la création des `container_dn` dans les nouveaux objets utilisés comme `subDn`. Voir la section concernée.

disable_creation

Booléen permettant de désactiver la création de ce type d'objet de manière globale.

before_modify

Chaîne de caractères (ou tableau de chaîne de caractères) correspondant au nom d'une ou plusieurs fonctions qui seront exécutées avant la modification d'un objet. Voir la section concernée.

after_modify

Chaîne de caractères (ou tableau de chaîne de caractères) correspondant au nom d'une ou plusieurs fonctions qui seront exécutées après la modification d'un objet. Voir la section concernée.

after_create

Chaîne de caractères (ou tableau de chaîne de caractères) correspondant au nom d'une ou plusieurs fonctions qui seront exécutées après la création d'un objet. Voir la section concernée.

after_delete

Chaîne de caractères (ou tableau de chaîne de caractères) correspondant au nom d'une ou plusieurs fonctions qui seront exécutées après la suppression d'un objet. Voir la section concernée.

label

Nom générique au pluriel qualifiant le type d'objet. Exemple : *Utilisateurs*.

display_name_format

Format paramétrable du nom des objets composés à partir des valeurs d'affichage des attributs de l'objet.

displayAttrName

Booléen définissant si le nom des attributs doit être affiché en préfixe de leur message d'aide (paramètre *help_info*).

customActions

Tableau associatif contenant les paramètres de configuration des `customActions`. Voir la section concernée.

LSrelation

Tableau associatif contenant les paramètres de configuration des `LSrelations`. Voir la section concernée.

LSform

Tableau associatif contenant les paramètres de configuration des LSforms des LSubjects. Voir la section concernée.

LSsearch

Tableau associatif contenant les paramètres de configuration des recherches de LSubject de ce type dans l'annuaire. Voir la section concernée.

globalSearch

Inclure ou non ce type d'objet dans le résultat des recherches globales (Par défaut : `True`).

ioFormat

Tableau associatif contenant les paramètres de configuration des formats de fichiers d'import/export de ce type d'LSubject. Voir la section concernée.

attrs

Tableau associatif contenant les paramètres de configuration des attributs des objets. Voir la section concernée.

4.2.1. Configuration des attributs

Cette section décrit les options de configuration des attributs des LSubjects. Les attributs sont définis dans le tableau associatif `attrs` de la configuration des LSubjects. Dans ce tableau, les clés les noms des attributs et les valeurs liés sont la configuration des attributs.

Avertissement

Contrairement à ce qui existe dans le standard LDAP, les noms des attributs sont sensibles à la casse. Il faut que le nom des attributs dans `LdapSaisie` soient scrupuleusement les mêmes que ceux retourné par `Net_LDAP2` (http://pear.php.net/package/Net_LDAP2)

Structure...

```
'attrs' => array (
  /* ----- start -----*/
  'attr1' => array (
    'label' => '[label de l'attr1]',
    'displayAttrName' => '[booleen]',
    'help_info' => '[Message d'aide sur l'attribut attr1]',
    'help_info_in_view' => '[booleen]',
    'ldap_type' => 'ldaptypel',
    'ldap_options' => array(
      // Options LDAP liées au type LDAP de l'attribut
    ),
    'html_type' => 'htmltypel',
    'html_options' => array(
```

```

    // Options HTML liées au type HTML de l'attribut
  ),
  'no_value_label' => '[No set value label]',
  'multiple' => 0,
  'required' => 1,
  'generate_function' => 'fonction1',
  'generate_value_format' => '[LSformat]',
  'default_value' => 'valeur1',
  'check_data' => array (
    // Règle de vérification syntaxique des données saisies
  ),
  'validation' => array (
    // Règle de vérification d'intégrité des données saisies
  ),
  'rights' => array(
    'LSprofile1' => 'droit1',
    'LSprofile2' => 'droit2',
    ...
  ),
  'view' => 1,
  'form' => array (
    'create' => 1,
    'modify' => 0,
    ...
  ),
  'dependAttrs' => array(
    // Attributs en dépendance
  ),
  'onDisplay' => 'fonction2'

  'before_modify' => 'fonction1',
  'after_modify' => 'fonction2'
),
/* ----- end -----*/
...
);
...

```

Paramètres de configuration

label

Le label de l'attribut.

displayAttrName

Booléen définissant si le nom de l'attribut doit être affiché en préfixe du message d'aide (paramètre *help_info*).

help_info

Message d'aide qui sera affiché dans une bulle d'aide à côté du nom de l'attribut dans les formulaires.

`help_info_in_view`

Booléen définissant si le message d'aide doit être affiché sur la vue de visualisation de l'objet.

Valeurs possibles : *0* ou *1*

Valeur par défaut : *0*

`ldap_type`

Le type LDAP de l'attribut (facultatif, par défaut: `LSattr_ldap_ascii`). Voir la section concernée.

`ldap_options`

Tableau associatif contenant les paramètres de configuration du type LDAP de l'attribut. Voir la section concernée.

`html_type`

Le type HTML de l'attribut (facultatif, par défaut: `LSattr_html_text`). Voir la section concernée.

`html_options`

Tableau associatif contenant les paramètres de configuration du type HTML de l'attribut. Voir la section concernée.

`no_value_label`

Label affiché lorsque l'attribut n'a pas de valeur (facultatif).

`multiple`

Booléen définissant si cet attribut peut stocker plusieurs valeurs.

Valeurs possibles : *0* ou *1*

Valeur par défaut : *0*

`required`

Booléen définissant si cet attribut doit obligatoirement être défini.

Valeurs possibles : *0* ou *1*

Valeur par défaut : *0*

`generate_function`

Nom de la fonction permettant de générer la valeur de l'attribut. Cette fonction sera exécutée, en passant en premier paramètre, l'objet LSobjet courant.

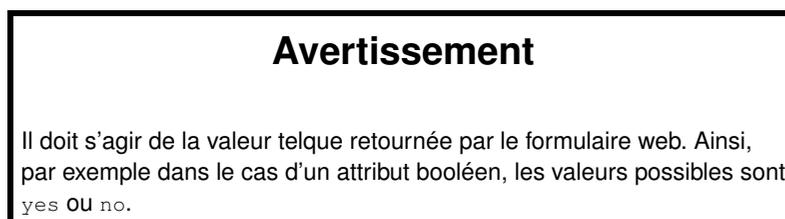
`generate_value_format`

LSformat permettant la génération de l'attribut.

Note : Cette méthode de génération est utilisée uniquement si aucune fonction de génération de la valeur n'est définie (paramètre `generate_function`).

`default_value`

Valeur par défaut de l'attribut.



Note : Cette valeur est également utilisée dans le cadre de la génération automatique de la valeur de l'attribut si aucune autre méthode n'est disponible (via une fonction ou un LSformat).

`check_data`

Tableau associatif contenant les règles de vérification syntaxique des données de l'attribut. Voir la section concernée.

`validation`

Tableau associatif contenant les règles de vérification d'intégrité des données de l'attribut. Voir la section concernée.

`rights`

Tableau associatif dont les clés sont les noms des LSprofiles ayant des droits sur cet attribut et les valeurs associées sont les droits correspondants. La valeur des droits d'un LSprofile peut être `r` pour le droit de lecture ou `w` pour le droit de lecture-écriture. Par défaut, un LSprofile n'a aucun droit.

`view`

Booléen définissant si l'attribut est, ou non, affiché lors de la visualisation des objets du type courant.

Valeurs possibles : 0 ou 1

Valeur par défaut : 0

form

Tableau associatif dont les clés sont les noms des LSforms et les valeurs associées la définition de l’affichage dans ce LSform. Si cette valeur vaut 0, alors l’attribut sera lecture-seule et si cette valeur vaut 1, cet attribut sera affiché en lecture-écriture.

dependAttrs

Tableau associatif listant les attributs dépendants de celui-ci. Les attributs listés ici seront régénérés lors de chaque modification de l’attribut courant. Cette génération sera effectué avec la fonction définie dans le paramètre `generate_function` de l’attribut.

onDisplay

Nom ou liste de nom des fonctions retournant les valeurs d’affichages de l’attribut. Si c’est une liste, chacune des fonctions seront exécutée les unes après les autres. Ces fonctions seront exécutées, en passant en premier paramètre, le tableau des valeurs de l’objet.

before_modify

Chaîne de caractères (ou tableau de chaîne de caractères) correspondant au nom d’une ou plusieurs fonctions qui seront exécutées avant toutes modifications de la valeur de l’attribut. Voir la section concernée

after_modify

Chaîne de caractères (ou tableau de chaîne de caractères) correspondant au nom d’une ou plusieurs fonctions qui seront exécutées après toutes modifications de la valeur de l’attribut. Voir la section concernée

4.2.1.1. Configuration des attributs LDAP

Cette section décrit les options propres à chacun des types d’attributs LDAP supportés par LdapSaisie.

4.2.1.1.1. LSattr_ldap_ascii

Ce type est utilisé pour la gestion des attributs dont la valeur est une chaîne de caractère. Ce type est le type par défaut.

4.2.1.1.2. LSattr_ldap_boolean

Ce type est utilisé pour la gestion des attributs dont la valeur est un booléen. On attend ici par booléen, tout attribut ne pouvant prendre que deux valeurs pré-définies correspond pour l’un à *Oui* et l’autre à *Non*

```
Structure...
'ldap_options' => array (
  'true_value' => '[valeur correspondant à Vrai]',
  'false_value' => '[valeur correspondant à Faux]'
),
...
```

Paramètres de configuration

true_value

La valeur de l'attribut correspondant à *Vrai*. (Par défaut : TRUE)

false_value

La valeur de l'attribut correspondant à *Faux*. (Par défaut : FALSE)

Important : Les valeurs possibles pour le paramètre *default_value* sont *yes* et *no*.

4.2.1.1.3. LSattr_ldap_compositeValueToJson

Ce type est utilisé pour la gestion des attributs composites dont les valeurs respectent le format suivant :
[key1=value1] [key2=value2] [...]

Ce type d'attribut LDAP sera utilisé pour convertir la valeur en son équivalent JSON pour pouvoir être traité à l'aide du type d'attribut HTML LSattr_html_jsonCompositeAttribute.

4.2.1.1.4. LSattr_ldap_date

Ce type est utilisé pour la gestion des attributs dont la valeur est une date.

Note : Au sein d'LdapSaisie, les dates manipulées au travers ce type d'attribut LDAP, sont au format *timestamp*. Il s'agit donc de nombres entiers correspondant au nombre de secondes depuis le 1 janvier 1970.

Le type d'attribut HTML utilisé conjointement avec ce type d'attribut LDAP devra être prévu pour recevoir et fournir des dates au format *timestamp*, comme c'est le cas pour le type d'attribut HTML *date*.

```
Structure...
'ldap_options' => array (
  'timestamp' => [Booléen], // Si la date est stockée au format timestamp
  'format' => '[Format de stockage]', // Default : "YmdHisO"
  'timezone' => '[Fuseau horaire]', // Default : "UTC"
),
```

...

Paramètres de configuration

timestamp

Booléen définissant si la date est stockée sous la forme d'un timestamp Unix (nombre de secondes depuis le 1er janvier 1970 à 00:00:00 UTC).

* Si `timestamp` est vrai, `LdapSaisie` ne tient pas compte du paramètre `format`.

format

Format de stockage de la date dans l'annuaire. Ce format est composé à partir des motifs clés gérés par la fonction `date()` de PHP (<http://www.php.net/>). Pour plus d'information, consulter la documentation officielle (<http://www.php.net/date>).

Note : La valeur par défaut est `YmdHisO`, correspondant à la syntaxe `Generalized Time` (sans les micro-secondes) telle que définie dans la RFC4517 (<https://tools.ietf.org/html/rfc4517>).

Exemples : `20091206230506Z (=2009/12/06 23:05:66 UTC)` ou `20190613143537+0200 (=2019/06/13 14:35:37 UTC+0200)`.

Avertissement

Si vous exploitez un attribut stockant une date incluant les micro-secondes, ce type d'attribut LDAP sera capable de gérer l'interprétation des valeurs stockées en configurant le format `YmdHis.uO`. En outre, le type d'attribut `LSattr_html_date`, s'appuyant sur les méthodes standards `strftime()` et `strptime()`, ne permettra pas aujourd'hui la saisie et l'affichage des millisecondes.

timezone

Fuseau horaire de stockage des dates dans l'annuaire LDAP. Les valeurs possibles sont documentées dans la documentation officielle de PHP (<https://www.php.net/timezones>). (Par défaut : UTC)

4.2.1.1.5. `LSattr_ldap_image`

Ce type est utilisé pour la gestion des attributs dont la valeur est une image. Pour le moment, aucun traitement particulier n'est appliqué pour le stockage.

4.2.1.1.6. `LSattr_ldap_naiveDate`

Ce type est utilisé pour la gestion des attributs dont la valeur est une date dont la `timezone` doit être ignorée. Côté LDAP, les dates seront stockées au format UTC étant donnée que la syntaxe LDAP exige

une *timezone*, cependant celle-ci sera complètement ignorée. Ce type peut-être utilisé à la place du type `LSattr_ldap_date`.

4.2.1.1.7. `LSattr_ldap_numeric`

Ce type est utilisé pour la gestion des attributs dont la valeur est un nombre. Pour le moment, aucun traitement particulier est n'appliqué pour le stockage.

4.2.1.1.8. `LSattr_ldap_password`

Ce type est utilisé pour la gestion des attributs dont la valeur est un mot de passe.

Structure...

```
'ldap_options' => array (
  'encode' => '[Type d'encodage du mot de passe]',
  'encode_function' => '[Nom de la fonction d'encodage]',
  'verify_function' => '[Nom de la fonction de vérification]',
  'no_random_crypt_salt' => '[Booléen]', // Désactivation de l'utilisation d'une salt aléat
  'wildcardPassword' => '[mot de passe(s) en clair]',
  'encodedWildcardPassword' => '[mot de passe(s) encodé(s)]'
),
...
```

Paramètres de configuration

`encode`

Nom du type d'encodage du mot de passe utilisé. Les types d'encodages supportés sont les suivants :

- `md5crypt`
- `crypt`
- `ext_des`
- `blowfish`
- `sha`
- `sha256`
- `sha512`
- `ssha`
- `ssha256`
- `ssha512`
- `smd5`
- `md5`
- `clear`

Note : Valeur par défaut : `md5crypt`

Important : Si le type d'encodage est inconnu, ou qu'il n'est pas supporté par le serveur web, un message d'erreur alertera l'utilisateur et le mot de passe sera stocké en clair.

`encode_function`

Nom d'une fonction qui sera utilisée afin d'encoder le mot de passe. Cette fonction recevra deux paramètres : le `LSldapObject` et le mot de passe en clair.

`verify_function`

Nom d'une fonction qui sera utilisée afin de valider un mot de passe soumis par l'utilisateur par rapport à celui stocké dans l'annuaire. Cette fonction recevra trois paramètres : le `LSldapObject`, le mot de passe en clair et le mot de passe hashé. Si ce paramètre est omis et que le paramètre `encode_function` est défini, le mot de passe à tester sera encodé à nouveau à l'aide de la fonction `encode_function` et le résultat sera comparé avec le mot de passe stocké dans l'annuaire.

`no_random_crypt_salt`

Désactivation de l'utilisation d'une salt générée aléatoirement au profit de l'utilisation des deux premiers caractères du mot de passe. Ce paramètre impacte uniquement le type de cryptage `crypt`.

`wildcardPassword`

Mot de passe (ou tableau de mot de passe) qui sera ajouté systématiquement, en plus du mot de passe choisi. Il sera encodé de la même manière que pour le mot de passe choisi avant enregistrement.

`encodedWildcardPassword`

Mot de passe (ou tableau de mot de passe) qui sera ajouté systématiquement, en plus du mot de passe choisi. Contrairement à la directive `wildcardPassword`, le mot de passe ne sera pas encodé avant enregistrement.

Note : Cette directive peut cohabiter avec sa cousine `wildcardPassword`. Les mot de passes contenus dans les deux directives seront alors ajoutés.

4.2.1.1.9. `LSattr_ldap_postaladdress`

Ce type est utilisé pour la gestion des attributs dont la valeur est construite sur le modèle de l'attribut standard `postalAddress`, c'est à dire dont les lignes sont séparées à l'aide du caractère de délimiteur `§`.

Lors de la lecture des valeurs de ce type d'attribut dans l'annuaire, les caractères \$ seront remplacés par des caractères \n et, à l'inverse, lors de l'écriture des valeurs de ce type d'attribut dans l'annuaire, les caractères \n seront remplacés par des caractères \$.

4.2.1.1.10. LSattr_ldap_pwdHistory

Ce type est utilisé pour la gestion de l'attribut standard *pwdHistory*. Cet attribut, accessible en lecture uniquement, stocke dans un format prédéfini l'historique des mots de passe d'une utilisateur avec pour chaque entrée :

- la date et heure de l'ajout du mot de passe dans l'historique
- l'OID de la syntaxe du mot de passe
- la longueur du mot de passe
- le mot de passe (hâché)

Ce type d'attribut LDAP permettra de convertir la valeur en son équivalent JSON pour pouvoir être traité à l'aide du type d'attribut HTML LSattr_html_jsonCompositeAttribute.

Exemple de valeur de l'attribut pwdHistory

```
20201202144718Z#1.3.6.1.4.1.1466.115.121.1.40#105#{SSHA512}XDSiR6Sh6W7gyVIk6Rr2OUv8rNPr+0rH
```

Exemple de valeur transformée

```
{"time":1606920438,"syntaxOID":"1.3.6.1.4.1.1466.115.121.1.40","length":105,"hashed_password":
```

Exemple de configuration complète de l'attribut

```
'pwdHistory' => array (
  'label' => 'Passwords in history',
  'ldap_type' => 'pwdHistory',
  'html_type' => 'jsonCompositeAttribute',
  'html_options' => array (
    'components' => array (
      'time' => array (
        'label' => 'Date added to history',
        'type' => 'text',
        'required' => true,
        'multiple' => false,
      ),
      'syntaxOID' => array (
        'label' => 'Syntax OID',
        'type' => 'text',
        'required' => true,
        'multiple' => false,
      ),
      'length' => array (
        'label' => 'Length',
        'type' => 'text',
        'required' => true,
      ),
    ),
  ),
)
```

```

        'multiple' => false,
    ),
    'hashed_password' => array (
        'label' => 'Hashed password',
        'type' => 'text',
        'required' => true,
        'multiple' => false,
    ),
),
),
),
'no_value_label' => 'History is empty.',
'multiple' => 1,
'rights' => array(
    'admin' => 'r',
),
'view' => 1,
),
),

```

La date et heure de l'ajout du mot de passe dans l'historique est convertie dans un format lisible. Par défaut, ce format est AAAA/MM/JJ HH:MM:SS, mais il peut aussi être personnalisé via le paramètre `date_format`. Ce format est composé à partir des motifs clés gérés par la fonction `date()` de PHP (<http://www.php.net/>). Pour plus d'information, consulter la documentation officielle (<http://www.php.net/date>).

Note : La valeur par défaut est `YmdHisO`, correspondant à la syntaxe `Generalized Time` telle que définie dans la RFC4517 (<https://tools.ietf.org/html/rfc4517>) et prévu par le `Draft-behera-ldap-password-policy` (<http://tools.ietf.org/id/draft-behera-ldap-password-policy-10.txt>) spécifiant cet attribut standard.

4.2.1.1.11. `LSattr_ldap_sambaAcctFlags`

Ce type est prévu pour gérer l'attribut `sambaAcctFlags` du schéma Samba, qui au travers d'une seule et unique valeur, respectant un format prévu, liste l'ensemble des drapeaux actifs d'un compte Samba. Il transforme l'unique valeur de l'attribut LDAP en une liste de drapeaux actuellement activés sur le compte. Il est conçu pour être utilisé conjointement avec le type d'attribut HTML `LSattr_html_sambaAcctFlags`.

4.2.1.1.12. `LSattr_ldap_shadowExpire`

Ce type est prévu pour gérer l'attribut `shadowExpire` du schéma POSIX, qui stocke une date sous la forme d'un entier correspondant au nombre de jours depuis le premier 1er 1970. Il est prévu pour être utilisé conjointement avec le type d'attribut HTML `LSattr_html_date`.

Note : Malgré son nom, ce type d'attribut LDAP peut être utilisé pour d'autres attributs stockant ce même format de date, tel que l'attribut `shadowLastChange`.

4.2.1.2. Configuration des attributs HTML

Cette section décrit les options propres à chacun des types d'attributs HTML supportés par LdapSaisie.

4.2.1.2.1. LSattr_html_boolean

Ce type est utilisé pour la gestion des attributs dont la valeur est un booléen.

La valeur retournée est l'une des chaînes de caractères suivantes :

- *yes* pour *Vrai*
- *no* pour *Faux*

```
Structure...  
'html_options' => array (  
  'true_label' => '[label]',  
  'false_label' => '[label]',  
)  
...  
...
```

Paramètres de configuration

true_label

Label affiché pour désigner la valeur *Vrai*.

false_label

Label affiché pour désigner la valeur *Faux*.

Note : Pour le moment, les attributs à valeurs multiples ne sont pas gérés.

Note : Pour maîtriser les valeurs stockées dans l'annuaire, il faut coupler ce type d'attribut HTML avec le type d'attribut LDAP boolean

Important : La définition de la valeur par défaut d'un attribut utilisant ce type HTML (paramètre `default_value`), doit se faire à l'aide des valeurs *yes* ou *no*.

4.2.1.2.2. *LSattr_html_date*

Ce type est utilisé pour la gestion des attributs dont la valeur est une date. L'outil de sélection de date MooTools-DatePicker (http://mootools.net/forge/p/mootools_datepicker) est utilisé pour la sélection graphique de la date et de l'heure.

Structure...

```
'html_options' => array (
  'format' => '[Format d'affichage de la date]',
  'time' => '[Booleen pour le choix ou non de l heure]',
  'manual' => '[Booleen pour l edition manuelle ou non]',
  'showNowButton' => '[Booleen]',
  'showTodayButton' => '[Booleen]',
  'style' => '[Nom du style utilise]',
  'special_values' => array (
    '[value]' => '[label]',
    [...]
  ),
),
...
```

Paramètres de configuration

format

Format d'affichage de la date dans le champ de saisie. Ce format est composé à partir des motifs clés suivants :

Mot clé	Valeur de substitution	Exemple de va
%a	Nom abrégé du jour de la semaine	De Sun à Sat
%A	Nom complet du jour de la semaine	De Sunday à Sat
%b	Nom du mois, abrégé, suivant la locale	De Jan à Dec
%B	Nom complet du mois, suivant la locale	De January à Dec
%c	Date et heure préférées, basées sur la locale	Exemple : Tue Feb 2009 pour le 5 Feb 12:45:10 AM
%d	Jour du mois en numérique, sur 2 chiffres (avec le zéro initial)	De 01 à 31
%e	Jour du mois, avec un espace précédant le premier chiffre. L'implémentation Windows est différente, voyez après pour plus d'informations.	De 1 à 31

Mot clé	Valeur de substitution	Exemple de va
%H	L'heure, sur 2 chiffres, au format 24 heures	De 00 à 23
%I	Heure, sur 2 chiffres, au format 12 heures	De 01 à 12
%j	Jour de l'année, sur 3 chiffres avec un zéro initial	001 à 366
%m	Mois, sur 2 chiffres	De 01 (pour Janvier) à 12 (pour Décembre)
%M	Minute, sur 2 chiffres	De 00 à 59
%p	'AM' ou 'PM', en majuscule, basé sur l'heure fournie	Exemple : AM pour 12:00, PM pour 22:23
%s	Timestamp de l'époque Unix (identique à la fonction time())	Exemple : 305810000 10 Septembre 1999 08:00 AM
%S	Seconde, sur 2 chiffres	De 00 à 59
%T	Identique à "%H:%M:%S" Exemple : 21:34:17 pour 21:34:17 PM	
%U	Numéro de la semaine de l'année donnée, en commençant par le premier Lundi comme première semaine	13 (pour la 13ème semaine pleine de l'année)
%w	Représentation numérique du jour de la semaine	De 0 (pour Dimanche) à 6 (pour Samedi)
%y	L'année, sur 2 chiffres	Exemple : 09 pour 2009 pour 1979
%Y	L'année, sur 4 chiffres	Exemple : 2038
%z	Soit le décalage horaire depuis UTC, ou son abréviation (suivant le système d'exploitation)	Exemple : -0500 pour l'heure de l'Est
%Z	Le décalage horaire ou son abréviation NON fournie par %z (suivant le système d'exploitation)	Exemple : -0500 pour l'heure de l'Est
%%	Le caractère de pourcentage ("%")	---

Note : La valeur par défaut est %d/%m/%Y, %T. Exemple : 23/04/2009, 23:03:04

time

Booléen définissant si l'outil de sélection permettra ou non le choix de l'heure en plus de la date

manual

Booléen autorisant ou non l'édition manuelle du champs. Si ce paramètre vaut `False`, la sélection se fera uniquement à l'aide de l'outil graphique

showNowButton

Booléen définissant si le bouton *Maintenant* est affiché ou non. Par défaut, il est affiché.

showTodayButton

Booléen définissant si le bouton *Aujourd'hui* est affiché ou non. Par défaut, il est affiché.

style

Nom du style d'affichage de l'outil de sélection. Les valeurs possibles sont par défaut :

- default
- dashboard
- vista
- jquery

Note : La création de nouveau thème est possible. Pour plus d'information, consulter l'aide de l'outil de sélection de date (http://mootools.net/forge/p/mootools_datepicker).

special_values

Tableau listant les valeurs spéciales que peut prendre l'attribut. Dans ce tableau associatif, la clé doit correspondre à la valeur de l'attribut (telle que fournie par l'attribut LDAP) et la valeur associée au label associé.

Ces valeurs spéciales seront proposées à l'utilisateur sous la forme de cases à cocher dans le formulaire. Elles peuvent permettre par exemple de donner une signification particulière au zéro pour un attribut LDAP stockant un *timestamp*.

4.2.1.2.3. LSattr_html_image

Ce type est utilisé pour la gestion des attributs dont la valeur est une image. Pour le moment, les attributs à valeurs multiples ne sont pas gérés.

4.2.1.2.4. LSattr_html_jsonCompositeAttribute

Ce type est utilisé pour la gestion des attributs dont les valeurs sont des dictionnaires de valeurs encodées aux formats *JSON*.

Exemple de valeur gérée

```
{"component1": "value1", "component2": "value2", "component3": "value3"}
```

Le principe est que ces dictionnaires contiennent plusieurs composants référencés par leur clé et stockant une valeur dont le type peut être un texte libre ou bien être issue d'une liste déroulante configurable selon le même principe que le type d'attribut `LSattr_html_select_list`.

Structure...

```
'html_options' => array (
  'components' => array (
    '[clé composant 1]' => array (
      'label' => '[Label du composant]',
      'help_info' => '[Message d'aide sur le composant]',
      'type' => '[Type de la valeur stocké]',
      'required' => [Booléen],
      'multiple' => [Booléen],
      'check_data' => => array (
        // Règle de vérification syntaxique des données saisies
      ),
    ),
    '[clé composant 2]' => array (
      'label' => '[Label du composant 2]',
      'type' => 'select_list',
      'required' => [Booléen],
      'options' => array (
        [Configuration équivalente à un attribut LSattr_html_select_list]
      )
    ),
    [...]
  ),
  'fullWidth' => [booléen],
),
...
```

Paramètres de configuration

components

Tableau associatif obligatoire contenant en valeur clé, l'identifiant des composants, correspondant à la clé dans le dictionnaire *JSON*, et en valeurs associés, la configuration du composant.

label

Le label du composant.

help_info

Message d'aide sur le composant (affiché uniquement en mode édition).

type

Le type de valeur du composant. Les types possibles sont `text` ou `select_list` pour respectivement soit une valeur saisie librement, soit une valeur sélectionnée parmi une liste déroulante.

options

Dans le cadre d'un composant de type `select_list`, cela correspond à la configuration de la liste déroulante. Cette configuration utilise la même syntaxe de configuration que celle du type d'attribut `LSattr_html_select_list` et son paramètre `html_options`.

multiple

Booléen définissant si ce composant peut stocker plusieurs valeurs (Défaut : *Faux*).

required

Booléen définissant si ce composant doit obligatoirement être défini (Défaut : *Faux*).

check_data

Tableau associatif contenant les règles de vérification syntaxique des données du composant. Ces règles sont configurables de la même manière que les celles des valeurs attributs. Voir la section concernée.

fullWidth

Booléen permettant de définir si l'affichage dans le formulaire doit se faire sur toute la largeur disponible de la page (Défaut : *Faux*).

4.2.1.2.5. LSattr_html_labeledValue

Ce type est utilisé pour la gestion des attributs dont la valeur est préfixé d'un `label` et qui respecte le format suivant : `[label]valeur`.

Structure...

```
'html_options' => array(
  'labels' => array ( // Liste des labels possible
    'label1' => 'Libellé label1',
    'label2' => 'Libellé label2',
    [...]
  ),
  'translate_labels' => [booléen],
),
...
```

Paramètres de configuration

labels

Tableau associatif obligatoire contenant en valeur clé, le `label` utilisé dans la valeur stockée et en valeur associée, le valeur d'affichage du `label`.

translate_labels

Booléen permettant d'activer/désactiver la traduction des labels (Par défaut : `Vrai`).

4.2.1.2.6. *LSattr_html_mail*

Ce type est utilisé pour la gestion des attributs dont la valeur est une adresse e-mail. Il offre les fonctionnalités suivantes :

- la possibilité d'envoyer des mails directement depuis l'interface de l'application ;
- l'autocomplétion lors de la saisie d'une adresse.

Structure...

```
'html_options' => array(
  'disableMailSending' => [booléen],

  // Autocomplétion pour un type d'LSobject donné
  'autocomplete' => array (
    'object_type' => '[Type d'LSobject]',
    'mail_attributes' => array (
      'mail',
      'mailAlternateAddress',
      [...]
    ),
    'filter' => '[filtre LDAP]',
    'basedn' => '[base DN spécifique]',
    'scope' => '[scope de recherche]',
    'displayFormat' => '[LSformat]',
    'onlyAccessible' => [booléen],
  ),

  // Autocomplétion sur la base d'une recherche LDAP brute
  'autocomplete' => array (
    'mail_attributes' => array (
      'mail',
      'mailAlternateAddress',
      [...]
    ),
    'filter' => '[filtre LDAP]',
    'basedn' => '[base DN spécifique]',
    'scope' => '[scope de recherche]',
    'displayFormat' => '[LSformat]',
    'onlyAccessible' => [booléen],
```

```

),
// Autocomplétion (par défaut)
'autocomplete' => true,
),
...

```

Paramètres de configuration

disableMailSending

Désactive l'envoi de mail depuis l'interface pour cet attribut.

Note : Ceci ne désactive pas pour autant le lien HTML de type *mailto:*. Pour cela, utilisez plutôt le type d'attribut HTML `text`.

autocomplete

Paramétrage de l'autocomplétion des adresses mails saisies : Il peut s'agir d'un tableau configurant les paramètres de l'autocomplétion ou simplement `true` pour activer l'autocomplétion par défaut, c'est à dire la recherche brute dans l'annuaire de n'importe quel objet ayant l'attribut `mail`.

En cas de configuration avancée, il est possible de faire une recherche :

- Sur la base d'un type d'LSobject donné : l'autocomplétion se fera alors comme n'importe quelle recherche d'un type d'objet donné.
- Sur la base d'une recherche brute dans l'annuaire : l'autocomplétion se fera alors sur la valeur de l'adresse mail recherchée et au travers une recherche brute dans l'annuaire sur n'importe quels objets ayant une adresse email correspondant.

Les paramètres associés à ces deux cas de figure sont décrits ci-dessous :

object_type

Le type d'LSobject recherché.

mail_attributes

Le(s) nom de l'attribut stockant les adresses emails recherchées. Il peut s'agir d'une chaîne de caractères ou d'un tableau s'il y a plusieurs attributs.

filter

Un filtre de recherche facultatif venant en plus de celui calculé automatiquement à partir du mot clé de recherche.

basedn

Le *basedn* de la recherche. *Paramètre facultatif*.

scope

Le *scope* de la recherche. *Paramètre facultatif, par défaut : sub*.

displayFormat

Le LSformat d'affichage des objets trouvés. Ce paramètre est facultatif et par défaut, il s'agira du format d'affichage propre au type d'LSobject (si défini) et à défaut, l'adresse mail trouvée sera affichée.

onlyAccessible

Booléen facultatif définissant si seul les LSobjects auxquels l'utilisateur connecté à accès doivent être considérés comme sélectionnables (Faux par défaut). Ce paramètre n'est appliqué que dans le cas d'une recherche pour un type d'LSobject donné.

Important : Ce type d'attribut HTML est dérivé du type text. Il profite donc de toutes les fonctionnalités d'un champ de ce type (autogénération, ...).

4.2.1.2.7. LSattr_html_maildir

Ce type est utilisé pour la gestion des attributs dont la valeur est le chemin d'une maildir. Typiquement, ce type attribut HTML est utile dans le cas de l'attribut *mailbox* utilisé par maildrop pour stocker le chemin des boîtes mails. Ce type d'attribut offre la possibilité de gérer un niveau de l'attribut et à travers les déclencheurs gérés par LdapSaisie la création, la modification et ou la suppression de la boîte mails. Le LSaddon boolean est utilisé pour manipuler la boîte mail à distance.

Note : Actuellement, cet LSaddon ne gérant que l'accès via FTP au serveur distant, l'API d'accès via FTP est attaquée directement.

```
Structure...
'html_options' => array (
  'LSform' => array (
    '[LSform1]' => [booléen],
    '[LSform2]' => [booléen],
    ...
  ),
  'remoteRootPathRegex' => "[Expression régulière pour matcher le dossier à créer]",
  'archiveNameFormat' => "[LSformat du chemin/nom du fichier une fois archiver]"
),
...
```

Paramètres de configuration

LSform

Tableau associatif obligatoire contenant en valeur clé le nom des LSforms dans lesquels la fonctionnalité de modification de la boîte mail sera présente. Les valeurs attachées sont des booléens définissant si la modification est active par défaut.

remoteRootPathRegex

Expression régulière (compatible Perl) facultative dont le but est de *matcher* dans la valeur complète du chemin distant de la *maildir*, le chemin de la *maildir* à créer une fois connecté sur le serveur.

Exemple : Si le chemin complet de la *maildir* est */home/vmail/user*, mais que l'utilisateur FTP lorsqu'il se connecte arrive directement dans */home/vmail*, et faut définir le paramètre *remoteRootPathRegex* de la manière suivante :

```
/^\home\/vmail\/([^\/]*)\/+$/
```

archiveNameFormat

LSformat du nom du dossier de la *maildir* une fois archivée. Si ce format est défini, le dossier ne sera pas supprimé mais déplacé ou renommé. Le format sera construit avec pour seul mot clé, le nom de l'ancien dossier. Exemple : Si le dossier de la *maildir* est */home/vmail/user* et le paramètre *archiveNameFormat* vaut *%{old}.bckp*, le dossier sera renommé en */home/vmail/user.bckp*.

Important : Ce format est interprété après application de la routine liée au paramètre *remoteRootPathRegex*. Ainsi, dans l'exemple précédent, si le paramètre *remoteRootPathRegex* tronquait uniquement le nom du dossier final, c'est à dire *user*, le format une fois interprété donnerai *user.bckp*.

4.2.1.2.8. LSattr_html_mailQuota

Ce type est utilisé pour la gestion des attributs dont la valeur est le quota d'une boîte mail. Le format de la valeur générée correspondant au format attendu par le serveur de mail Courier (<http://www.courier-mta.org/>) par défaut. Exemple : *50000000S* correspond à un quota de 50Mio.

```
Structure...
'html_options' => array(
  'suffix' => '[suffix]',
)
),
...
```

Paramètres de configuration

suffix

Chaîne de caractères suffixant la valeur du quota (Par défaut : s).

4.2.1.2.9. LSattr_html_password

Ce type est utilisé pour la gestion des attributs dont la valeur est un mot de passe.

```
Structure...
'html_options' => array(
  'isLoginPassword' => [booléen],
  'generationTool' => [booléen],
  'autoGenerate' => [booléen],
  'length' => [nombre de caractères],
  'chars' => array ( // Caractères que peut contenir le mot de passe
    array( // Liste caractère avec un nombre minimum d'apparition supérieur à 1
      'nb' => [nb caractères],
      'chars' => '[liste de caractères possibles]'
    ),
    '[autre liste de caractères possibles]', // Liste caractère avec un nombre
                                          // d'apparitions égal à 1
  ),
  ...
),
'use_pwgen' => [booléen], // Utiliser pwgen pour la génération du mot de passe
'pwgen_path' => "/path/to/pwgen",
'pwgen_opts' => "[options à passer à pwgen]",
'verify' => [booléen], // Activation de l'outil de vérification du mot de passe
'viewHash' => [booléen], // Activation de l'outil de visualisation du mot de passe haché
'confirmChange' => [booléen], // Activation de la confirmation en cas de changement du m
'confirmChangeQuestion' => "[LSformat]", // LSformat de la question de confirmation du c
'mail' => array( // Configuration de l'envoi du mot de passe par mail
  'subject' => "[LSformat du sujet du mail]",
  'msg' => "[LSformat du message du mail]",
  'mail_attr' => 'mail', // Attribut mail de l'objet
  'get_mail_attr_fonction' => '[fonction]', // Fonction retournant l'attribut mail de l'o
  'send' => 1, // Activation par défaut de l'envoi du mot de passe
  'ask' => 1, // Laisser le choix à l'utilisateur
  'canEdit' => 1, // Activation de l'édition du LSformat du message par l'utilisateur
  'checkDomain' => false, // Désactivation de la vérification du domaine de l'adresse ema
  'domain' => '[nom de domaine]', // Nom de domaine obligatoire lors de la validation de
)
),
...
```

Paramètres de configuration

isLoginPassword

Booléen définissant si le mot de passe est celui utilisé par l'utilisateur pour se connecter à l'annuaire LDAP. Si c'est le cas, pour vérifier si le mot de passe correspond avec un autre, une tentative de connexion de l'utilisateur à l'annuaire sera faite. (Par défaut : `Faux`)

generationTool

Booléen définissant si l'outil de génération de mot de passe est activé.

autoGenerate

Active la génération automatique du mot de passe lorsque l'attribut n'a encore aucune valeur de définie. Il faut également que l'outil de génération soit activé (`generationTool`).

length

Nombre de caractères que devront contenir les mots de passe générés.

chars

Tableau contenant une liste de listes de caractères possibles pour composer le mot de passe. Dans chacune de ces listes, au moins un caractère sera utilisé dans le nouveau mot de passe. Il est possible de définir un nombre supérieur de caractères d'une liste devant apparaître dans les mots de passe générés en spécifiant un tableau associatif dont la clé `nb` associera le nombre entier de caractères et la clé `chars` la liste de caractères. Une liste de caractères est un chaîne.

use_pwgen

Booléen définissant si la commande `pwgen` doit être utilisé pour générer le mot de passe.

pwgen_path

Chemin d'accès au binaire `pwgen`. (Par défaut : `pwgen`).

pwgen_opts

Options à passer à la commande `pwgen`.

verify

Booléen définissant si l'outil de vérification du mot de passe est activé. Si celui-ci est activé, l'utilisateur pourra entrer un mot de passe dans le champ et cliquer sur un bouton qui lancera une procédure de vérification du mot de passe via un test de connexion à l'annuaire.

viewHash

Booléen définissant si l'utilisateur aura accès à la fonctionnalité de visualisation du mot de passe haché.

confirmInput

Booléen définissant si un second champ mot de passe sera affiché dans le formulaire pour que l'utilisateur confirme la saisie du nouveau mot de passe.

`confirmInputError`

LSformat du message d'erreur affiché à l'utilisateur si le mot de passe saisi dans le champs de confirmation ne correspond pas au nouveau mot de passe. *Paramètre facultatif.*

`confirmChange`

Booléen définissant si l'utilisateur devra confirmer le changement de ce mot de passe. Lorsque cette fonctionnalité est activée, l'utilisateur verra apparaître une popup de confirmation à la validation du formulaire s'il a saisi un nouveau mot de passe.

`confirmChangeQuestion`

LSformat de la question posée à l'utilisateur en cas de changement du mot de passe et si la fonctionnalité est activée. Il sera composé à l'aide du *label* de l'attribut. *Paramètre facultatif.*

`clearView`

Booléen définissant si l'utilisateur pourra voir le mot de passe en clair par défaut (y compris en mode visualisation uniquement).

`clearEdit`

Booléen définissant si l'utilisateur éditera le mot de passe au travers un champs HTML de type *text* et donc lisible ou au travers un champs HTML de type *password*.

`mail`

Paramètres de configuration de l'envoi par mail du mot de passe à l'utilisateur. Lorsque cet outil est activé, lors de la modification/création du mot de passe, l'utilisateur pourra recevoir un mail lui spécifiant son nouveau mot de passe.

Paramètres de configuration

`send`

Booléen définissant si l'envoi du mot de passe est activé par défaut.

`ask`

Booléen définissant si on laisse le choix à l'utilisateur d'activer ou non l'envoi du mot de passe par mail.

`canEdit`

Booléen définissant si on laisse la possibilité à l'utilisateur d'éditer le LSformat du message et du sujet.

`subject`

LSformat du sujet du mail. Ce format sera composé avec la valeur du nouveau mot de passe de l'utilisateur.

msg

LSformat du message du mail. Ce format sera composé avec les informations de l'objet LDAP, y compris le mot clé *%{password}* correspondant à la valeur du nouveau mot de passe de l'utilisateur.

mail_attr

Le nom de l'attribut listant les mails possibles de l'utilisateur. Par défaut, la première valeur de l'attribut sera utilisée comme adresse mail destinatrice. Cet attribut peut également être un tableau de plusieurs noms d'attributs. Dans ce cas, la première valeur correcte sera retenue. Si *canEdit* est activé, l'utilisateur pourra choisir l'adresse mail destinatrice parmi la liste des valeurs de l'attribut.

get_mail_attr_function

Nom de la fonction (ou *callable* au sens PHP) qui sera utilisé pour récupérer le nom de l'attribut listant les mails possibles de l'utilisateur. Cette fonction prendra en paramètre, l'objet *LSformElement* courant et devra retourner une valeur équivalente au paramètre de configuration *mail_attr*. Si ce paramètre est défini, il prévalera toujours sur le paramètre *mail_attr*.

bcc

Mettre en *BCC* un mail systématiquement (ou plusieurs en les séparant par des virgules).

headers

Un tableau de type clé/valeur ou la clé est le nom d'un header à ajouter au mail et la valeur est la valeur de l'header en question.

checkDomain

Booléen définissant si le domaine de l'adresse mail doit être validée. *Paramètre facultatif, par défaut: TRUE*

domain

Nom de domaine obligatoire lors de la validation de l'adresse mail. Ce paramètre peut être une simple chaîne correspondant au domaine ou un tableau listant plusieurs domaines valides. *Paramètre facultatif, par défaut tous les domaines sont acceptés.*

4.2.1.2.10. LSattr_html_postaladdress

Ce type est utilisé pour la gestion des attributs du type de l'attribut standard *postalAddress*. Ce type d'attribut permet d'afficher, en plus de l'adresse, un lien composé à partir d'informations de l'objet permettant par exemple d'afficher un lien vers une carte géocalisant l'adresse postale.

Par défaut, le lien ajouté sera un lien de recherche de l'adresse postale générée à partir de la valeur de l'attribut (en remplaçant les retours à la ligne ($\backslash n$) par des espaces) via le service Nominatim d'OpenStreetMap (<http://nominatim.openstreetmap.org/>).

Note : Dans le cadre du fonctionnement par défaut et pour maîtriser les valeurs stockées dans l'annuaire, il faut coupler ce type d'attribut HTML avec le type d'attribut LDAP postaladdress

```
Structure...
'html_options' => array(
  'map_url_pattern_format' => '[LSformat]',
  'map_url_pattern_generate_function' => '[callable]',
  'map_url_format' => '[LSformat]',
),
...
```

Paramètres de configuration

map_url_pattern_format

Ce LSformat doit permettre de générer la valeur de l'adresse postale qui sera insérée dans l'URL du lien ajouté dans l'interface.

map_url_pattern_generate_function

Ce paramètre permet de définir une fonction qui sera utilisée à la place du paramètre `map_url_pattern_format` pour générer la valeur de l'adresse postale qui sera insérée dans l'URL du lien ajouté dans l'interface. Cette fonction prendra en paramètre l'objet *LSformElement* courant et devra retourner une chaîne de caractères correspondant à l'adresse postale à insérer dans le lien de l'interface. Par défaut, la fonction `LSformElement_postaladdress__generate_pattern` est utilisée.

map_url_format

Ce LSformat doit permettre de générer l'URL du lien ajouté dans l'interface. Il sera composé avec les informations de l'objet LDAP, y compris le mot clé `%{pattern}` correspondant à la valeur de l'adresse postale générée à l'aide des paramètres précédents. Par défaut, le format suivant sera utilisé : `http://nominatim.openstreetmap.org/search.php?q=%{pattern}`

4.2.1.2.11. LSattr_html_pre

Ce type est dérivé du type `LSattr_html_textarea` et permet simplement que lors de l'affichage de la valeur, celle-ci soit affichée en respectant les retours à la ligne et en utilisant une police de caractères `monospace`. Cela reproduit l'affichage d'une balise HTML `pre`.

4.2.1.2.12. LSattr_html_rss

Ce type est utilisé pour la gestion des attributs dont la valeur est l'URL d'un flux RSS. Il propose directement dans l'interface, la possibilité d'accéder au flux RSS.

Important : Ce type d'attribut HTML est dérivé du type `text`. Il profite donc de toutes les fonctionnalités d'un champ de ce type (autogénération, ...).

4.2.1.2.13. *LSattr_html_sambaAcctFlags*

Ce type est prévu pour gérer l'attribut *sambaAcctFlags* du schéma Samba, qui au travers d'une seule et unique valeur, respectant un format prévu, liste l'ensemble des drapeaux actifs d'un compte Samba. Il est conçu pour être utilisé conjointement avec le type d'attribut LDAP *LSattr_ldap_sambaAcctFlags*.

Pour définir la valeur par défaut de cet attribut, il faut définir paramètre `default_value` comme un tableau des drapeaux telque prévu par Samba :

Listes des drapeaux Samba

U

Compte utilisateur standard

W

Compte de poste de travail approuvé

S

Compte de serveur approuvé

I

Compte de domaine approuvé

M

Compte de connexion Majority Node Set (MNS)

H

Dossier personnel requis

N

Compte sans mot de passe

X

Le mot de passe n'expire jamais

D

Compte désactivé

T

Copie temporaire d'un autre compte

L

Compte automatiquement bloqué

```

Exemple de valeur par défaut...
'default_value' => array('U', 'X'),
...

```

Note : Ce type d'attribut est implémenté en dérivant le type *LSattr_html_select_box* dont les valeurs possibles sont pré-configurées (paramètre `possible_values`). Même si cela n'est pas forcément utiles, les autres paramètres du type parent restent utilisables.

4.2.1.2.14. *LSattr_html_select_box*

Ce type est identique au type *LSattr_html_select_list* excepté qu'il utilise en lieu et place d'une balise HTML `select`, plusieurs balises HTML `input` de type `checkbox` en cas de valeurs multiples ou de type `radio` en cas de valeur unique. Les paramètres de configuration sont entièrement hérités de la classe *LSattr_html_select_list* et sont donc exactement les mêmes.

4.2.1.2.15. *LSattr_html_select_list*

Ce type est utilisé pour la gestion des attributs dont les valeurs font partie d'une liste statique ou dynamique. Il est possible de lister des valeurs statiques et également des références à d'autres *LSobjects*. La référence à un objet correspond à une valeur clé, référente à un objet précis, qui peut être soit la valeur d'un de ses attributs, soit son *DN*.

```

Structure...
'html_options' => array (
  'possible_values' => array (
    '[LSformat de la valeur clé]' => '[LSformat du nom d'affichage]',
    ...
  'OTHER_OBJECT' => array (
    'object_type' => '[Type d'LSobject]',
    'display_name_format' => '[LSformat du nom d'affichage des LSobjects]',
    'value_attribute' => '[Nom de l'attribut clé]',
    'values_attribute' => '[Nom de l'attribut clé multi-valeur]',
    'filter' => '[Filtre de recherche des LSobject]',
    'scope' => '[Scope de la recherche]',
    'basedn' => '[Basedn de la recherche]',
    'onlyAccessible' => '[Booléen]'
  ),
  'OTHER_ATTRIBUTE' => '[attr]',
  // Or :
  'OTHER_ATTRIBUTE' => array(
    '[attr1]' => '[label1]',

```

```

    '[attr2]' => '[label2]',
    [...]
  ),
  // Or :
  'OTHER_ATTRIBUTE' => array(
    'attr' => [attr],
    'json_component_key' => '[Composant JSON clé]',
    'json_component_label' => '[Composant JSON label]',
  ),
  array (
    'label' => '[LSformat du nom du groupe de valeurs]',
    'possible_values' => array (
      '[LSformat de la valeur clé]' => '[LSformat du nom d'affichage]',
      ...
      'OTHER_OBJECT' => array (
        ...
      )
    )
  )
),
'get_possible_values' => [callable],
'translate_labels' => [booléen],
'sort' => [Booléen],
'sortDirection' => '[ASC|DESC]'
),
...

```

Paramètres de configuration

possible_values

Tableau associatif obligatoire contenant en valeur clé le LSformat des valeurs clés présent par l'attribut et en valeurs associées, le LSformat des noms d'affichage de ces valeurs. Ces LSformats sont composés à partir des valeurs de l'objet courant (attributs, dn, ...).

Si la valeur clé est égale à `OTHER_OBJECT`, une liste d'LSubject sera insérée dans la liste des valeurs possibles. La valeur associée est alors un tableau associatif dont les valeurs clés sont les noms des paramètres de configuration de la recherche de ces LSubjects et les valeurs associées, les valeurs des paramètres.

Il est possible de regrouper des valeurs de l'attribut en plaçant leur déclaration dans un sous-tableau. Ce sous-tableau devra contenir la clé `label` dont la valeur associée sera le LSformat du nom du groupe de valeurs. Ce LSformat est composé à partir des valeurs de l'objet courant (attributs, dn, ...). Une seconde clé `possible_values` regroupera les valeurs possibles du groupe. Comme pour le tableau principal, la clé `OTHER_OBJECT` permet d'incorporer une liste d'LSubject.

object_type

Nom du type d'LSubject en référence.

`display_name_format`

LSformat du nom d'affichage des objets lors de leur sélection.

`value_attribute`

Nom de l'attribut des LSobjets en référence servant de valeur clé et permettant de les identifier (Exemple : *dn* ou *uid*).

`values_attribute`

Nom de l'attribut des LSobjets en référence servant de catalogue de valeurs. Dans ce mode, la valeur n'a pas de label et est affichée directement dans l'interface. Ce paramètre peut-être utilisé en complément ou non du paramètre *value_attribute*.

`filter`

Filtre facultatif de la recherche des LSobjets. Il sera dans tous les cas agrémenté des valeurs des *objectclass* du type d'LSobjet.

`scope`

Scope facultatif de la recherche des LSobjets.

`basedn`

Basedn facultatif de la recherche des LSobjets.

`onlyAccessible`

Booléen facultatif définissant si seul les LSobjets auxquels l'utilisateur connecté a accès doivent être considérés comme sélectionnables (Faux par défaut).

Si la valeur clé est égale à `OTHER_ATTRIBUTE`, une liste de valeur possible sera composée à l'aide des valeurs d'un (ou plusieurs) autre attribut de l'objet courant. La valeur associée peut être alors :

- soit le nom d'un attribut dont les valeurs seront utilisées comme valeurs possibles (la valeur affichée est égale à la valeur stockée).
- soit un tableau associatif dont les valeurs clés sont les noms des attributs dont les valeurs seront utilisés comme valeurs possibles et dont les valeurs associés seront les labels sous lesquels ces valeurs seront regroupées (la valeur affichée est égale à la valeur stockée).
- soit un tableau associatif référençant un attribut sous la clé *attr* dont les valeurs seront utilisées comme valeurs possibles. Cet attribut peut-être du type `LSattr_html_jsonCompositeAttribute`. Il sera alors possible d'utiliser les valeurs d'un composant en particulier en le référençant à l'aide de la clé *json_component_key*. Il est également possible de référencer un autre composant à l'aide de la clé *json_component_label* et dont les valeurs seront utilisées comme valeurs affichées lors de la sélection. À défaut, les valeurs affichées seront identiques à celles stockées.

`get_possible_values`

Paramètre permettant de spécifier un *callable* qui sera utilisé pour lister les valeurs possibles de l'attribut. Il recevra en paramètres les informations suivantes:

`$options`

Les options HTML de l'attribut.

`$name`

Le nom de l'attribut.

`&$ldapObject`

Une référence à l'objet `LSldapObject`.

La valeur de retour attendue est un tableau associatif des valeurs possibles de l'attribut avec la valeur que prendra l'attribut en tant que clé et le label correspondant en tant que valeur. Tout autre retour sera considéré comme un échec et déclenchera une erreur.

Il est également possible de regrouper des valeurs possibles de l'attribut: pour cela, le tableau retourné devra lui-même contenir un tableau associatif contenant la label traduit du groupe sous la clé `label` et les valeurs possibles du groupe sous la clé `possible_values`.

Les valeurs retournées pourront être combinées avec les autres valeurs possibles configurées de l'attribut. La prise en charge du tri des valeurs possibles est assurée par la fonction appelante sauf dans le cas des sous-groupes de valeurs possibles. Dans ce cas, la méthode `LSattr_html_select_list :: _sort()` pourra être utilisée pour trier les valeurs du sous-groupe: cette méthode accepte en paramètre une référence du tableau des valeurs possibles ainsi que les options HTML de l'attribut.

Si la traduction des labels des valeurs possibles de l'attribut est activées (voir ci-dessous), celle-ci doit être prise en charge par la fonction configurée.

`translate_labels`

Booléen permettant d'activer/désactiver la traduction des labels (Par défaut : `Vrai`).

`sort`

Booléen définissant si les valeurs possibles doivent être triées ou non (`Vrai` par défaut). Le tri est effectué sur les libellés des valeurs possibles.

`sortDirection`

Mot clé déterminant le sens du tri des valeurs possibles.

Valeurs possibles : `ASC` ou `DESC` (`ASC` par défaut).

4.2.1.2.16. LSattr_html_select_object

Ce type est utilisé pour la gestion des attributs dont les valeurs sont des références à d'autres LSubjects. Chaque référence à un objet correspond à une valeur prise par l'attribut. Les valeurs clés référant à un LSubject sont soit la valeur d'un de leurs attributs, soit leur *DN*.

```
Structure...
'html_options' => array (
  selectable_object => array (
    array (
      'object_type' => '[Type d'LSubject sélectionnable]',
      'display_name_format' => '[LSformat du nom d'affichage des LSubjects]',
      'value_attribute' => '[Nom de l'attribut clé des LSubjects]',
      'filter' => '[Filtre de recherche]',
      'onlyAccessible' => '[Booléen]'
    ),
    [...]
  ),
  'ordered' => [Booléen],
  'sort' => [Booléen],
  'sortDirection' => '[ASC|DESC]'
),
...
```

Paramètres de configuration

selectable_object

Tableau dont chaque valeur correspond à un tableau associatif spécifiant un type d'LSubject sélectionnable. Pour chaque type d'objet sélectionnable, les paramètres suivants doivent être renseignés :

object_type

Nom du type d'LSubject en référence (*Paramètre obligatoire*).

display_name_format

LSformat du nom d'affichage des objets lors de leur sélection (*Paramètre facultatif*).

value_attribute

Nom de l'attribut des LSubjects en référence servant de valeur clé et permettant de les identifier (*Paramètre obligatoire, exemples : dn ou uid*).

filter

Filtre de recherche qui sera ajouter au filtre par défaut lors de la sélection des objets (*Paramètre facultatif*).

onlyAccessible

Booléen définissant si seul les LSobjets auxquels l'utilisateur connecté à accès doivent être considérés comme sélectionnables (*Paramètre facultatif, par défaut: False*).

ordered

Booléen définissant si la liste des objets choisis doit être ordonnable ou non (*Paramètre facultatif, par défaut: False*). Cela aura pour effet d'activer une fonctionnalité dynamique de l'interface permettant de remonter ou descendre dans la liste les objets choisis.

Note : Cette fonctionnalité désactive automatiquement le trie des objets à l'affichage.

sort

Booléen définissant si la liste des objets choisis doit être triée ou non (*Paramètre facultatif, par défaut: True*). Le trie est effectué sur les libellés des objets choisis.

sortDirection

Mot clé déterminant le sens du trie des objets choisis.

Valeurs possibles : ASC ou DESC (ASC par défaut).

4.2.1.2.17. LSattr_html_ssh_key

Ce type est utilisé pour la gestion des attributs dont la valeur est une clef publique SSH. Il permet dans l'interface, d'avoir un affichage adapté à ce type de donnée.

4.2.1.2.18. LSattr_html_tel

Ce type est utilisé pour la gestion des attributs dont la valeur est un numéro de téléphone. Lors de l'affichage, un lien hypertexte avec une URI de type `tel:~~` est affiché.

Important : Ce type d'attribut HTML est dérivé du type text. Il profite donc de toutes les fonctionnalités d'un champ de ce type (autogénération, ...).

4.2.1.2.19. *LSattr_html_text*

Ce type est utilisé pour la gestion des attributs dont la valeur est une chaîne de caractères devant être affichée dans un champ *input* HTML de type *text*.

```
Structure...
'html_options' => array(
  'generate_value_format' => '[LSformat pour la génération de la valeur]',
  'autoGenerateOnCreate' => [booléen],
  'autoGenerateOnModify' => [booléen],
  'withoutAccent' => [booléen],
  'replaceSpaces' => "[chaîne de remplacement]",
  'upperCase' => [booléen],
  'lowerCase' => [booléen]
),
...
```

Paramètres de configuration

generate_value_format

LSformat de la valeur utilisée pour la génération automatique de celle-ci à partir des informations saisies dans le formulaire. Les valeurs clefs du format sont les noms des attributs de l'objet. Seuls les attributs affichés au moins en lecture seule dans le formulaire peuvent être utilisés dans le format. Une seule valeur par attribut sera utilisée pour la génération : celle du premier champ (dans l'ordre d'apparition dans le formulaire).

Important : Seuls les éléments du formulaire de type HTML *input*, *select* ou *textarea* peuvent être utilisés.

autoGenerateOnCreate

Activation de la génération automatique lorsque celui-ci est vide au moment du chargement du formulaire.

Note : La valeur par défaut est *False*.

autoGenerateOnModify

Activation de la génération automatique lors de chaque modification de la valeur des champs du formulaire lié.

Note : La valeur par défaut est *False*.

withoutAccent

Activation de la suppression des accents dans la chaîne de caractères générée automatiquement.

Note : La valeur par défaut est *False*.

withoutAccent

Activation du remplacement des accents dans la chaîne de caractères générée automatiquement. La valeur de remplacement est celle du paramètre.

Note : La valeur par défaut est *False*.

upperCase

Activation de la mise en majuscule de la valeur générée automatiquement.

Note : La valeur par défaut est *False*.

lowerCase

Activation de la mise en minuscule de la valeur générée automatiquement.

Note : La valeur par défaut est *False*.

4.2.1.2.20. LSattr_html_textarea

Ce type est utilisé pour la gestion des attributs dont la valeur est une chaîne de caractères trop longue pour être saisie dans un champ HTML *input* de type *text* et est plus adapté à un champ HTML *textarea*.

4.2.1.2.21. *LSattr_html_url*

Ce type est utilisé pour la gestion des attributs dont la valeur est une URL. Il propose directement dans l'interface, la possibilité d'accéder au site ou encore de l'ajouter dans ses favoris (lorsque le navigateur le supporte).

Important : Ce type d'attribut HTML est dérivé du type text. Il profite donc de toutes les fonctionnalités d'un champ de ce type (autogénération, ...).

4.2.1.2.22. *LSattr_html_valueWithUnit*

Ce type est utilisé pour la gestion des attributs dont la valeur est un entier auquel un facteur peut s'appliquer (par exemple : Kilo, Méga, ...).

```
Structure...
'html_options' => array(
  'units' => array (
    '[facteur1]' => '[label unit1]',
    '[facteur2]' => '[label unit2]',
    [...]
  ),
  'translate_labels' => [booléen],
  'nb_decimals' => [number of decimals],
  'dec_point' => '[decimals point]',
  'thousands_sep' => '[thousands separator]',
  'store_integer' => [booléen],
  'round_down' => [booléen],
)
),
...
```

Paramètres de configuration

units

Tableau associatif dont la clé est un entier correspondant au facteur et la valeur est le label de l'unité. (Par exemple : 1 => Octet, 1024 => Kilo-octet, ...).

translate_labels

Booléen permettant d'activer/désactiver la traduction des labels (Par défaut : Vrai).

nb_decimals

Le nombre de décimales à afficher en cas de nombre non-entier (Par défaut : 2).

`dec_point`

Le caractère à utiliser comme séparateur de décimal (Par défaut, une virgule).

`thousands_sep`

Le caractère à utiliser comme séparateur de milliers (Par défaut, un espace).

`store_integer`

Booléen permettant d'activer/désactiver le stockage de valeurs entières (Par défaut : `Vrai`).

`round_down`

Booléen permettant d'arrondir à l'entier inférieur (et non à l'entier supérieur par défaut) en cas de stockage de valeurs entières.

4.2.1.2.23. `LSattr_html_wysiwyg`

Ce type est utilisé pour la gestion des attributs dont la valeur est du code HTML et dont l'édition doit être faite à l'aide d'un éditeur WYSIWYG . La librairie TinyMCE (<https://www.tinymce.com>) est utilisée pour cela.

```
Structure...
'html_options' => array(
  'extra_options' => array (
    [Options à passer à TinyMCE]
  ),
),
...
```

Paramètres de configuration

`extra_options`

Ce paramètre permet de passer des options à TinyMCE (<https://www.tinymce.com>) pour personnaliser son comportement. Par exemple, il est possible d'utiliser le paramètre `valid_styles` pour définir quels styles CSS sont autorisés. Pour plus d'informations, consultez la documentation de TinyMCE (<https://www.tinymce.com>).

4.2.1.2.24. `LSattr_html_xmpp`

Ce type est utilisé pour la gestion des attributs dont la valeur est une adresse XMPP. Il propose directement dans l'interface, la possibilité de lancer une fenêtre de dialogue avec l'interlocuteur de son client XMPP préféré .

Note : Cette fonctionnalité n'est supporté uniquement par les navigateurs web supportant les URI de type `xmpp://~`.

Important : Ce type d'attribut HTML est dérivé du type text. Il profite donc de toutes les fonctionnalités d'un champ de ce type (autogénération, ...).

4.2.1.3. Configuration des règles de vérification syntaxique

Cette section décrit la manière de configurer des règles de vérification syntaxique sur les données des attributs. Ces règles seront utilisées pour vérifier que les valeurs saisies par un utilisateur dans un formulaire sont correctes.

```
Structure...
'check_data' => array (
  '[regle1]' => array(
    'msg' => "[Message d'erreur]",
    'params' => array(
      // Paramètres de la règle
    )
  ),
  ...
),
...
```

Le paramètre *check_data* est un tableau associatif dont les clés sont les noms des règles de vérification syntaxique actives et les valeurs associées sont des tableaux associatifs contenant les paramètres des règles.

Paramètres de configuration

msg

Le message d'erreur à afficher lors que la règle n'est pas respectée.

params

Tableau associatif contenant les paramètres de la règle. Les paramètres possibles sont propres à chaque type de règle. Les clés sont les noms des paramètres et les valeurs associés, les valeurs des paramètres.

4.2.1.3.1. *alphanumeric*

Cette règle vérifie que la valeur est une chaîne de caractères composée uniquement de lettres non-accuénées, en minuscule ou en majuscule et/ou de chiffres.

Paramètres de configuration

`withAccents`

Si le paramètre est à `true`, les lettres accentuées seront acceptées.

4.2.1.3.2. *callable*

Cette règle vérifie que la valeur saisie est correcte en utilisant une fonction personnalisée. Cette fonction devra prendre en paramètres la valeur à valider, le tableau des paramètres de la règle ainsi qu'un pointeur sur l'objet `LSformElement`. Sur la base de ses informations, elle devra valider la valeur et retourner `True` si la valeur est valide et `False` sinon.

Paramètres de configuration

`callable`

Le nom de la fonction (ou tout autre `callable` au sens PHP) de validation.

4.2.1.3.3. *date*

Cette règle vérifie que la valeur saisie est bien une date et qu'elle respecte un format précis.

Paramètres de configuration

`format`

Format de la date à respecter. Ce format doit être compatible avec la fonction `strtotime()` de PHP (<http://www.php.net/>). Voir la documentation de la fonction (<http://www.php.net/strtotime>)

`special_values`

Tableau listant les valeurs spéciales que peut prendre l'attribut. Dans ce tableau, seules les valeurs sont utilisées et les clés n'ont pas d'importance. Ces valeurs spéciales n'auront pas forcément besoin de respecter le format attendu.

4.2.1.3.4. *differentPassword*

Cette règle vérifie que la valeur saisie ne correspond pas à un des mots de passe stockés dans d'autres attributs du même objet.

Important : Les autres attributs doivent utiliser le type d'attribut `LDAP LSattr_ldap_password`.

Paramètres de configuration

otherPasswordAttributes

La liste des autres attributs dont les mots de passe doivent être différents.

4.2.1.3.5. email

Cette règle vérifie que la valeur saisie est bien une adresse e-mail. Il est possible de vérifier si elle appartient bien à un domaine en particulier ou encore de vérifier si le domaine existe et qu'il possède un serveur de mail(MX).

Paramètres de configuration

domain

Nom de domaine obligatoire. Ce paramètre peut être une simple chaîne correspondant au domaine ou un tableau listant plusieurs domaines possibles.

checkDomain

Booléen définissant si le domaine de l'adresse mail doit être validée.

4.2.1.3.6. filesize

Cette règle vérifie que la valeur est un fichier dont la taille en octets respecte les limites passées en paramètre.

Paramètres de configuration

minSize

Taille minimum.

maxSize

Taille maximum.

4.2.1.3.7. imagefile

Cette règle vérifie que la valeur est bien un fichier et que le type mime de celui-ci est bien une image. Cette règle utilise la règle mimetype en spécifiant si l'utilisateur ne le fait pas que le type mime doit respecter la regex suivante : `/image\/.*\/`

Important : Cette règle est une simple interface à la règle mimetype, il est donc possible de passer d'autres paramètres propres à ce type.

4.2.1.3.8. *imagesize*

Cette règle vérifie que la valeur est une image dont la taille en pixels respecte les limites passées en paramètre.

Paramètres de configuration

`minWidth`

Largeur minimum.

`maxWidth`

Largeur maximum.

`minHeight`

Hauteur minimum.

`maxHeight`

Hauteur maximum.

4.2.1.3.9. *inarray*

Cette règle vérifie que la valeur saisie fait partie d'une liste de valeurs autorisées.

Paramètres de configuration

`possible_values`

Tableau listant les valeurs autorisées.

4.2.1.3.10. *integer*

Cette règle vérifie que la valeur saisie est un entier. Les paramètres permettent de spécifier éventuellement si la valeur doit être positive ou négative et également de borner les valeurs valides.

Paramètres de configuration

`positive`

Booléen définissant si la valeur doit être positive.

`negative`

Booléen définissant si la valeur doit être négative.

min

Valeur minimale (supérieur ou égale).

max

Valeur maximale (inférieur ou égale).

4.2.1.3.11. *lettersonly*

Cette règle vérifie que la valeur est une chaîne de caractères composée uniquement de lettres non-accuées, en minuscule ou en majuscule.

4.2.1.3.12. *maxlength*

Cette règle vérifie que la valeur saisie est une chaîne de caractères dont la longueur est inférieur ou égale à la valeur passées en paramètre.

Paramètres de configuration

limit

Limite supérieur (ou égale) de la longueur de la chaîne de caractères.

4.2.1.3.13. *minlength*

Cette règle vérifie que la valeur saisie est une chaîne de caractères dont la longueur est supérieur ou égale à la valeur passée en paramètre.

Paramètres de configuration

limit

Limite inférieure (ou égale) de la longueur de la chaîne de caractères.

4.2.1.3.14. *mimetype*

Cette règle vérifie que la valeur est bien un fichier et que le type mime de celui-ci est correct. Il est possible de vérifier si le type mime fait partie d'une liste ou encore s'il valide une expression régulière.

Paramètres de configuration

contentType

Type mime obligatoire. Ce paramètre peut être une simple chaîne correspondant au type mime ou un tableau listant plusieurs possibilités.

contentTypeRegEx

Expression régulière que doit respecter le type mime.

4.2.1.3.15. nonzero

Cette règle vérifie que la valeur est une valeur numérique non nulle.

4.2.1.3.16. nopunctuation

Cette règle vérifie que la valeur est une chaîne de caractères ne contenant pas de signe de ponctuation. Les caractères suivants sont actuellement exclus : () . \ / \ * \ ^ \ ? # ! @ \$ % + = , " ' > < ~ [] { }

4.2.1.3.17. numeric

Cette règle vérifie que la valeur est une valeur numérique.

4.2.1.3.18. password

Cette règle vérifie que la valeur est un mot de passe respectant la politique de sécurité définie par les paramètres de la règle.

Paramètres de configuration

minlength

Longueur minimale du mot de passe.

maxlength

Longueur maximale du mot de passe.

prohibitedValues

Tableau de valeurs interdites.

regex

Expression(s) régulière(s) que doit respecter le mot de passe. Ce paramètre peut être une expression régulière au format PCRE (<http://php.net/pcre.pattern>) ou un tableau d'expressions régulières.

minValidRegex

Le nombre minimum d'expression régulière qui doivent être validées pour que le mot de passe soit considéré comme correct. Ce paramètre est optionnel, par défaut, toutes les expressions régulières doivent être validées.

4.2.1.3.19. *rangelength*

Cette règle vérifie que la valeur saisie est une chaîne de caractères dont la longueur est comprise entre deux valeurs passées en paramètre.

Paramètre de configuration

limits

Tableau contenant deux valeurs, la première étant la limite inférieure ou égale et la seconde la limite supérieure ou égale.

4.2.1.3.20. *regex*

Cette règle vérifie que la valeur saisie respecte bien l'expression régulière passée en paramètre.

Paramètres de configuration

regex

L'expression régulière devant être respectée. Cette expression régulière doit être au format PCRE (<http://php.net/pcre.pattern>).

4.2.1.3.21. *required*

Cette règle vérifie que la valeur n'est pas une chaîne de caractères de longueur nulle.

4.2.1.3.22. *ssh_pub_key*

Cette règle vérifie que la valeur est une clé publique SSH.

Cette vérification utilise tout d'abord une expression régulière pour valider la forme syntaxique de la clé publique (`ssh-[type] [clé au format base64] [commentaire]`) puis tente de décoder la partie en base64 de la clé pour vérifier qu'il s'agit bien d'une chaîne de caractères.

4.2.1.3.23. telephonenumber

Cette règle vérifie que la valeur est un numéro de téléphone français. Celui-ci doit respecter l'expression régulière suivante : `/^(01|02|03|04|05|06|08|09)[0-9]{8}$/`

4.2.1.4. Configuration des règles de vérification d'intégrité

Cette section décrit la manière de configurer des règles de vérification d'intégrité sur les données des attributs. Il est possible de valider la valeur de l'attribut par l'intermédiaire de la vérification de résultat d'une recherche paramétrable dans l'annuaire ou encore d'appeler une fonction de votre choix pour effectuer la vérification voulue.

4.2.1.4.1. Validation par l'analyse du résultat d'une recherche dans l'annuaire

Une telle règle permet de vérifier si les valeurs des attributs n'entrent pas en conflit avec d'autres objets de l'annuaire. Ce test peut également permettre de vérifier si les valeurs devant faire référence à d'autres objets de l'annuaire sont correctes.

```
Structure...
'validation' => array (
  ...
  array(
    'msg' => "[LSformat du message d'erreur]",
    'filter' => '[LSformat du filtre de la recherche]',
    'object_type' => '[Type d'LSobject recherché]',
    'basedn' => '[BaseDn de la recherche]',
    'scope' => '[Scope de la recherche]',
    'result' => '[Résultat positif de la recherche]',
    'except_current_object' => '[Exclure l'objet courant]'
  ),
  ...
),
...
```

Paramètres de configuration

msg

LSformat du message d'erreur à afficher lorsque la validation échoue. Ce format est construit avec les valeurs du LSobject.

filter

LSformat du filtre de la recherche. Ce format peut être construit avec toutes les valeurs du LSobject (attributs, DN, ...) et également avec la valeur à valider en utilisant pour mot clé `%{val}`.

object_type

Le nom du type d'LSobject recherché. Si un type est spécifié, le filtre de la recherche sera une combinaison de celui du paramètre *filter* et du filtre composé à partir des *objectClass* du type d'LSobject. *Paramètre facultatif.*

basedn

Le *basedn* de la recherche. *Paramètre facultatif.*

scope

Le *scope* de la recherche. *Paramètre facultatif.*

result

Le résultat de la recherche : si *result* vaut zéro, la recherche ne devra retourner aucun objet pour que la validation soit réussie. Sinon, la recherche devra retourner au moins un objet.

except_current_object

Booléen définissant si l'objet courant doit être exclu du résultat de la recherche. Ce paramètre n'est évalué quand cas de création (formulaire *create*).

4.2.1.4.2. Validation par l'exécution d'une fonction

Il est possible d'effectuer la validation de l'attribut par l'exécution d'une fonction de votre choix. Il lui sera passé en paramètre une référence à l'objet `LSldapObject` courant. Si la fonction ne retourne pas *true*, la validation échouera.

```
Structure...
'validation' => array (
  ..
  array(
    'msg' => "[LSformat du message d'erreur]",
    'function' => '[Nom de la fonction de validation]'
  ),
  ...
),
...
```

Paramètres de configuration**msg**

LSformat du message d'erreur à afficher lorsque la validation échoue. Ce format est construit avec les valeurs du LSobject.

function

Le nom de la fonction à exécuter. Si cette fonction n'existe pas, un message d'erreur sera affiché et la validation échouera.

4.2.1.5. Déclencheurs

Cette section décrit la manière de paramétrer des déclencheurs afin que LdapSaisie exécute durant ses processus, et à des moments bien précis des traitements d'un LSattribute, des fonctions que vous pourrez développer vous même. De plus, le résultat de l'exécution de vos fonctions pourra influencer sur le déroulement des processus.

Actuellement, les évènements suivant sont gérés :

Nom	Description	Bloquant
before_create	Avant la création du LSubject, lorsque l'attribut a au moins une valeur.	Oui
after_create	Après la création du LSubject, lorsque l'attribut a au moins une valeur.	Non
before_modify	Avant la modification de la valeur de l'attribut.	Oui
after_modify	Après la modification de la valeur de l'attribut.	Non
before_delete	Avant la suppression du LSubject contenant l'attribut.	Oui
after_delete	Après la suppression du LSubject contenant l'attribut.	Non

Note : Si un événement est dit *bloquant*, lors de l'exécution des actions liées, si une des fonctions retourne `false`, le processus s'arrêtera.

4.2.1.5.1. Configuration

La configuration des déclencheurs se fait dans la définition des LSattributes. Par exemple, pour définir les fonctions à exécuter après la modification de la valeur de l'attribut *mail* du type LSubject *LSpople*, c'est à dire lors de leur événement `after_modify`, il faut définir la variable suivante :

```
$GLOBALS['LSubjects']['LSpople']['attrs']['mail']['after_modify']
```

Cette variable peut contenir soit une chaîne de caractères correspondant au nom de la fonction à exécuter, soit un tableau de chaînes de caractères correspondant aux noms des fonctions à exécuter.

4.2.1.5.2. Ecriture d'une fonction

Une fonction exécuté par un déclencheur d'un LAttribute se déclare de la manière suivante :

```
/*
 * Ma fonction à exécuter lors de l'évènement [event]
 *
 * Paramètre :
 *   - $object : Le LSubject contenant le LAttribute sur lequel l'évènement
 *               survient
 *
 * Valeurs retournées :
 *   - True : Tout s'est bien passé
 *   - False : Une erreur est survenue ou la fonction souhaite bloquer le
 *             processus lors d'un évènement bloquant.
 */
function maFonction ($object) {

    // Actions

}
```

Cette fonction doit prendre pour seul paramètre, le LSubject contenant le LAttribute sur lequel l'évènement survient et doit retourner soit `True` si tout s'est bien passé, soit `False` en cas de problème. Dans le cas d'un évènement bloquant, si la fonction retourne `False`, le processus est arrêté.

4.2.2. Création automatique du conteneur des LSubjects dans un subDn

Cette section décrit la manière de configurer la création automatique des conteneurs des LSubjects. Si le *basedn* correspondant à la branche de stockage des LSubjects n'existe pas, LdapSaisie tentera de le créer à partir de la configuration de la variable `$GLOBALS['LSubjects']['[nom du type d'LSubject]']['container_auto_create']`.

```
Structure
$GLOBALS['LSubjects']['[nom du type d'LSubject]']['container_auto_create'] = array (
    'objectclass' => array(
        'objectclass1',
        'objectclass2',
        ...
    ),
    'attrs' => array(
        'attr1' => 'vall1',
        'attr2' => array(
            'val2',
            'val3',
```

```

        ...
    ),
    ...
)
);

```

Paramètres de configuration

`objectclass`

La liste des *objectclass* de l'objet conteneur.

`attrs`

Un tableau associatif dont les clés sont les noms des attributs de l'objet conteneur à définir et dont les valeurs associées sont la/les valeur(s) de ces attributs.

4.2.3. Déclencheurs

Cette section décrit la manière de paramétrer des déclencheurs afin que LdapSaisie exécute durant ses processus, et à des moments bien précis des traitements d'un LSubject, des fonctions que vous pourrez développer vous même. De plus, le résultat de l'exécution de vos fonctions pourra influencer sur le déroulement des processus.

Actuellement, les événements suivant sont gérés :

Nom	Description	Bloquant
<code>before_create</code>	Avant la création du LSubject.	Oui
<code>after_create</code>	Après la création du LSubject.	Non
<code>before_modify</code>	Avant la modification du LSubject	Oui
<code>after_modify</code>	Après la modification du LSubject	Non
<code>before_rename</code>	Avant de renommer le LSubject	Oui
<code>after_rename</code>	Après avoir renommé le LSubject	Non
<code>before_delete</code>	Avant la suppression du LSubject	Oui
<code>after_delete</code>	Après la suppression du LSubject	Non

Note : Si un événement est dit *bloquant*, lors de l'exécution des actions liées, si une des fonctions retourne `false`, le processus s'arrêtera.

4.2.3.1. Configuration

La configuration des déclencheurs se fait dans la définition des types d'LSobjects. Par exemple, pour définir les fonctions à exécuter après la modification des LSobjects de type *LSpeople*, c'est à dire lors de leur évènement `after_modify`, il faut définir la variable suivante :

```
$GLOBALS['LSobjects'][$nom du type d'LSobject]['after_modify']
```

Cette variable peut contenir soit une chaîne de caractères correspondant au nom de la fonction à exécuter, soit un tableau de chaînes de caractères correspondant aux noms des fonctions à exécuter.

4.2.3.2. Ecriture d'une fonction

Une fonction exécuté par un déclencheur d'un LSobject se déclare de la manière suivante :

```
/*
 * Ma fonction à exécuter lors de l'évènement [event]
 *
 * Paramètre :
 *   - $object : Le LSobject sur lequel l'évènement survient
 *
 * Valeurs retournées :
 *   - True : Tout s'est bien passé
 *   - False : Une erreur est survenue ou la fonction souhaite bloquer le
 *             processus lors d'un évènement bloquant.
 */
function maFonction ($object) {

    // Actions

}
```

Cette fonction doit prendre pour seul paramètre, le LSobject sur lequel l'évènement survient et doit retourner soit `True` si tout s'est bien passé, soit `False` en cas de problème. Dans le cas d'un évènement bloquant, si la fonction retourne `False`, le processus est arrêté.

4.2.4. customActions

Cette section décrit la manière de configurer les actions personnalisées exécutables sur les LSobjects appelées `customActions`.

```

Structure
$GLOBALS['LSubjects']['[nom du type d'LSubject]']['customActions'] = array (
  'action1' => array(
    'label' => '[label l'action]',
    'hideLabel' => '[booléen]',
    'helpInfo' => '[label d'aide]',
    'icon' => '[nom de l'icône de l'action]',
    'function' => '[fonction à exécuter]',
    'question_format' => '[LSformat de la question de confirmation]',
    'onSuccessMsgFormat' => '[LSformat du message à afficher en cas de succès de l'action]',
    'disableOnSuccessMsg' => '[booléen]',
    'noConfirmation' => '[booléen]',
    'redirectToObjectList' => '[booléen]',
    'noRedirect' => '[booléen]',
    'rights' => array(
      'LSprofile1',
      'LSprofile2',
      ...
    )
  )
);

```

Paramètres de configuration

label

Le label de l'action.

hideLabel

Cache le label dans le bouton de l'action.

helpInfo

Le label du message d'aide qui sera affiché au survole du bouton de l'action.

icon

Nom de l'icône à afficher dans le bouton de l'action. Ce nom correspond au nom du fichier de l'image (sans l'extention) qui devra se trouver dans le dossier *src/images/[nom du theme d'images]/* ou dans le dossier *src/local/images*.

function

Le nom de la fonction à exécuter qui implémente l'action personnalisée Cette fonction prendra en seule paramètre le LSubject sur lequel l'action devra être exécutée et retournera *True* en cas de succès ou *False* en cas d'échec d'exécution de la fonction.

question_format

Le LSformat de la question de confirmation d'exécution de l'action. Ce LSformat sera composé à l'aide du nom de l'objet.

onSuccessMsgFormat

Le LSformat du message à afficher en cas de succès d'exécution de l'action. Ce LSformat sera composé à l'aide du nom de l'objet.

disableOnSuccessMsg

Booléen permettant de désactiver le message afficher en cas de succès d'exécution de l'action.

noConfirmation

Booléen permettant de désactiver la confirmation de l'exécution de l'action.

redirectToObjectList

Booléen permettant de rediriger l'utilisateur vers la liste des objets plutôt que sur la fiche de l'objet après l'exécution de l'action.

noRedirect

Booléen permettant de désactiver la redirection de l'utilisateur après l'exécution de l'action. Cela permet à la fonction de définir son propre fichier de template de retour et donc d'afficher une page personnalisable.

rights

Tableau contenant la liste des noms des LSprofiles ayant le droit d'exécuter cette action.

4.2.4.1. Ecriture d'une fonction implémentant une *customAction*

Une fonction implémentant une *customAction* se déclare de la manière suivante :

```
/*
 * Ma fonction implémentant ma customAction
 *
 * Paramètre :
 *   - $object : Le LSubject sur lequel mon action doit être exécutée
 *
 * Valeurs retournées :
 *   - True : Tout s'est bien passé
 *   - False : Une erreur est survenue
 */
function maFonction ($object) {

    // Actions

}
```

Cette fonction doit prendre pour seul paramètre, le LSubject sur lequel l'action personnalisée doit être exécutée et doit retourner soit `True` si tout s'est bien passé, soit `False` en cas de problème.

Note : Ces fonctions sont le plus couramment définies au sein d'LSaddon.

4.2.5. LSrelation

Cette section décrit la manière de configurer les relations entre les LSubjects appelées LSrelation.

Dans le cadre d'une liaison dite *simple*, c'est à dire une liaison au travers la valeur d'un attribut qui fera directement référence à un autre objet (*DN* ou la première valeur d'un attribut de référence), pourra être configurée simplement en spécifiant l'attribut de liaison et le type de valeur qu'il contient. Dans le cas d'une liaison plus complexe, il sera possible de développer vous même des méthodes de mise en relation.

Structure

```

$GLOBALS['LSubjects']['[nom du type d'LSubject]']['LSrelation'] = array (
  'relation1' => array(
    'label' => '[label de la relation]',
    'emptyText' => "[texte affiché si aucune relation avec d'autres objets
                    n'existe pour l'objet courant]",
    'LSubject' => '[le type d'LSubject en relation]',

    // Liaison simple
    'linkAttribute' => '[attribut de liaison]',
    'linkAttributeValue' => '[valeur de l'attribut de liaison]',
    'linkAttributeOtherValues' => array('[autres valeurs possible de l'attribut de liaison]

    // Liaison complexe
    'list_function' => '[methode1]',
    'getKeyvalue_function' => '[methode2]',
    'update_function' => '[methode3]',
    'remove_function' => '[methode4]',
    'rename_function' => '[methode5]',
    'canEdit_function' => '[methode6]',
    'canEdit_attribute' => '[nom d'attribut]',

    'rights' => array(
      'LSprofile1' => 'r',
      'LSprofile2' => 'w',
      ...
    )
  )
);

```

Paramètres de configuration

label

Le label de la relation.

emptyText

Le texte à afficher pour décrire le fait que l'objet courant n'a aucune relation d'établie avec d'autres LSubjects. Exemple (au sujet d'un utilisateur) : *N'appartient à aucun groupe.*

LSubject

Le type d'LSubject en relation avec le type courant. *(Facultatif en cas de liaison complexe)*

linkAttribute

Dans le cadre d'une relation simple, il s'agit de l'attribut de liaison du type d'LSubject en relation avec le type courant, c'est à dire l'attribut dans lequel on retrouve une valeur en relation avec l'objet courant. *(Facultatif en cas de liaison complexe)*

linkAttributeValue

Dans le cadre d'une relation simple, il s'agit du type de valeur present par l'attribut de liaison du type d'LSubject en relation avec le type courant. Il peut s'agir du mot clé `dn` si l'attribut de liaison contient le *DN* de l'objet courant ou bien le nom d'un attribut du type d'objet courant dont la première valeur sera stockée par l'attribut de liaison. *(Facultatif en cas de liaison complexe)*

linkAttributeOtherValues

Dans le cadre d'une relation simple, il s'agit d'autres types de valeur possiblement present par l'attribut en plus de celui défini par le paramètre `linkAttributeValue`. Ce paramètre ne sert qu'à détecter des liaisons établies à l'aide de valeurs autres que celle relative au paramètre `linkAttributeValue` : en cas de nouvelle liaison, c'est la valeur associée à ce dernier qui sera utilisée pour établir la liaison. *(Facultatif en cas de liaison complexe)*

list_function

La méthode de la classe du type d'LSubject en relation, permettant de lister les objets de ce type en relation avec l'objet courant. *(Facultatif en cas de liaison simple)*

getkeyvalue_function

La méthode de la classe du type d'LSubject en relation, permettant d'obtenir la valeur clé à stocker pour établir la relation entre l'objet courant et d'autres objets du type concerné. *(Facultatif en cas de liaison simple)*

update_function

La méthode de la classe du type d'LSubject en relation, permettant de mettre à jour les relations existantes entre l'objet courant et les objets du type concerné. Cette liste d'objets en relation est établie par l'utilisateur à travers l'interface. *(Facultatif en cas de liaison simple)*

remove_function

La méthode de la classe du type d'LSubject en relation permettant de supprimer une relation existante entre l'objet courant et un objet du type concerné. *(Facultatif en cas de liaison simple)*

rename_function

La méthode de la classe du type d'LSubject en relation permettant d'effectuer les actions nécessaires lorsque l'objet courant est renommé dans le but de maintenir les valeurs clés permettant d'établir les

relations entre l'objet courant et les objets en relation avec lui. (*Facultatif en cas de liaison simple*)

canEdit_function

La méthode de la classe du type d'LSobject en relation permettant de vérifier que l'utilisateur a le droit de modifier la relation avec un objet en particulier. (*Facultatif en cas de liaison simple*)

canEdit_attribute

Le nom de l'attribut du type d'LSobject en relation devant être éditable par l'utilisateur pour que celui-ci puisse modifier la relation. Dans le cadre d'une relation simple, celui-ci peut, si nécessaire, être différent du paramètre `linkAttribute`.

rights

Tableau associatif dont les clés sont les noms des LSprofiles ayant des droits sur cette relation et dont les valeurs associées sont les droits correspondants. La valeur des droits d'un LSprofile peut être `r` pour le droit de lecture ou `w` pour le droit de lecture-écriture. Par défaut, un LSprofile n'a aucun droit.

4.2.6. LSform

Cette section décrit la manière de paramétrer les formulaires d'LdapSaisie pour un type LSobject donné. Pour chaque type d'LSobject, il faut configurer plusieurs formulaires correspondant aux vues gérées par LdapSaisie (création, modification, ...). Les formulaires se configurent par plusieurs biais :

- Via la configuration des attributs : La configuration des attributs détermine la présence ou non des attributs dans les formulaires. Elle permet également de définir si on souhaite bloquer leur présence en lecture seulement.
- Via les droits de l'utilisateur connecté sur les attributs de l'objet à éditer : en fonction des droits de l'utilisateur sur un attribut, celui-ci apparaîtra en lecture-écriture ou en lecture uniquement voir pas du tout.
- Via la configuration au niveau de chaque type d'LSobject : il y est possible de définir le comportement globale du formulaire comme la validation via Ajax ou encore la disposition logique des attributs dans le formulaire.

Structure

```
$GLOBALS['LSobjects'][$nom du type d'LSobject]['LSform'] = array (
    'ajaxSubmit' => [booléen],
    'layout' => array (
        // Configuration de la disposition logique des attributs
    ),
    'dataEntryForm' => array (
        // Configuration des masques de saisie
    )
);
```

Paramètres de configuration

ajaxSubmit

Booléen définissant si le formulaire sera envoyé via une requête Ajax plutôt qu'à travers un rafraîchissement de la page. Par défaut : *VRAI*.

layout

Tableau contenant la configuration de l'affichage du formulaire : il est possible de définir la disposition des attributs dans le formulaire en les regroupant dans des onglets et en les faisant apparaître dans un ordre logique. Voir la section concernée.

dataEntryForm

Tableau contenant la configuration des masques de saisie : il est possible de définir des masques de saisie pour faire en sorte que lors de la création d'un objet, seul un certain nombre d'éléments soit demandé à l'utilisateur. Voir la section concernée.

4.2.6.1. Configuration de l'affichage

La configuration des *layout* se situe dans la configuration des LSubjects, dans la variable `layout` (`$GLOBALS['LSubjects']['[nom du type d'LSubject]']['LSform']['layout']`). Cette variable est un tableau associatif dont la clé est l'identifiant de l'onglet et dont la valeur associée est la configuration de l'onglet.

Structure

```
$GLOBALS['LSubjects']['[nom du type d'LSubject]']['LSform']['layout'] = array (
    'onglet1' => array(
        'label' => '[label de l'onglet]',
        'img' => 1, // Valeur possible 1 ou 0
        'args' => array (
            'arg1',
            'arg2',
            ...
        )
    ),
    ...
);
```

Paramètres de configuration

label

Le label de l'onglet.

img

Affiche ou non l'image d'un éventuel attribut de type HTML LSattr_html_image.

args

Tableau associatif contenant une liste ordonnée des attributs qui apparaîtront dans l'onglet.

Important : Lorsqu'un *layout* est défini, celui-ci est "suivi à la lettre" pour l'affichage du LSform. Ainsi, si un attribut est défini dans la configuration de l'objet comme présent dans le LSform courant, mais que celui-ci n'est pas présent dans le *layout*, il ne sera pas du tout affiché.

4.2.6.2. Configuration des masques de saisie

La configuration des masques de saisie (*dataEntryForm*) se situe dans la configuration des LSobjects, dans la variable `dataEntryForm` (`$GLOBALS['LSobjects']`)[*[nom du type d'LSobject]*][*'LSform'*][*'dataEntryForm'*]). Cette variable est un tableau associatif dont la clé est l'identifiant du masque de saisie et dont la valeur associée est sa configuration.

Structure

```
$GLOBALS['LSobjects'] [[nom du type d'LSobject]] ['LSform'] ['dataEntryForm'] = array (
  'masquel' => array(
    'label' => [[label du masque de saisie]],
    'disabledLayout' => [booleen],
    'displayedElements' => array (
      'arg1',
      'arg2',
      ...
    ),
    'defaultValues' => array (
      'arg3' => [value],
      'arg4' => [value],
      ...
    ),
    'requiredAllAttributes' => [booleen],
    'requiredAttributes' => array (
      'arg1',
      'arg2',
      ...
    )
  ),
  ...
);
```

Paramètres de configuration

label

Le label du masque de saisie.

disabledLayout

Active ou non les layouts pour ce masque de saisie.

displayedElements

Tableau contenant la liste des attributs qui devront être saisi dans le masque de saisie.

defaultValues

Tableau associatif contenant la liste des valeurs par défaut des attributs. Les valeurs multiples sont possibles en utilisant des tableaux.

Important : Les valeurs seront vue comme des valeurs retournées par le formulaire et non comme des valeurs des attribus LDAP eux-même. Ainsi et par exemple, un attribut traité comme un booléen dans un formulaire pourra prendre comme valeur par défaut *yes* ou *no*.

requiredAttributes

Tableau contenant la liste des attributs obligatoires du masque de saisie. Cette liste d'attributs obligatoires viendra en complément de la configuration des attributs. Il est ainsi possible de rendre des attributs obligatoires durant la saisie d'un masque tout en les laissant facultatif le reste du temps.

requiredAllAttributes

Si ce parametre vaut *True*, tout les attributs du masque de saisie seront tous obligatoires de la même manière qu'avec le paramètre *requiredAttributes*.

4.2.7. LSsearch

Cette section décrit la manière de paramétrer les recherches dans l'annuaire pour un type d'LSobject donné.

La configuration des *LSsearch* se situe dans la configuration des LSobjects, dans la variable *LSsearch* (*\$GLOBALS['LSobjects']['[nom du type d'LSobject]']['LSsearch']*).

Structure

```
$GLOBALS['LSobjects']['[nom du type d'LSobject]']['LSsearch'] = array (
  'attrs' => array(
    'attr1',
    'attr2',
    ...
    'attr3' => array(
      'searchLSformat' => '[LSformat]',
      'approxLSformat' => '[LSformat]',
    ),
  ),
),
```

```

...
),
'params' => array(
  // Paramètres de la recherche
  'pattern' => '[string]',
  'sizelimit' => [integer],
  'recursive' => [boolean],
  'approx' => [boolean],
  'withoutCache' => [boolean],
  'onlyAccessible' => [boolean],
  // Paramètres de tri
  'sortBy' => [displayName|subDn],
  'sortDirection' => [ASC|DESC],
  'sortlimit' => [integer],
  // Paramètre d'affichage
  'displayFormat' => [LSformat],
  'nbObjectsByPage' => [integer],
  'nbObjectsByPageChoices' => array([integer], [integer], ...),
  'validPatternRegex' => '[regex]'
),
'predefinedFilters' => array(
  'filter1' => 'label filter1',
  'filter2' => 'label filter2'
),
'extraDisplayedColumns' => array(
  'col1' => array(
    'label' => 'label column 1',
    'LSformat' => '[LSformat]'
  ),
  'col2' => array(
    'label' => 'label column 2',
    'generateFunction' => '[fonction de génération]',
    'additionalAttrs' => array('[attr1]', '[attr2]', ...),
    'escape' => [booléen],
  ),
  'col3' => array(
    'label' => 'label column 3',
    'LSformat' => '[LSformat]',
    'alternativeLSformats' => array (
      '[LSformat 1]',
      '[LSformat 2]'
    ),
    'formaterLSformat' => '[LSformat]',
    'formaterFunction' => '[fonction de formatage]',
    'cssStyle' => '[CSS style]',
    'visibleTo' => array (
      '[LSprofile 1]',
      '[LSprofile 2]'
    )
  ),
),
'customActions' => array (
  // Configuration des customActions pour les recherches de ce type d'objet

```

```
)
);
```

Paramètres de configuration

`attrs`

Tableau listant les attributs pouvant être utilisés dans les filtres de recherche LDAP employés par LdapSaisie. Lorsqu'un motif de recherche est passé par l'utilisateur, LdapSaisie composera un filtre LDAP à partir de cette liste.

Lors d'une recherche non-approximative, le filtre de recherche sera composé (par défaut) de la manière suivante :

```
(|(attr1=*motif*)(attr2=*motif*)...)
```

Lors d'une recherche approximative, le filtre de recherche sera composé (par défaut) de la manière suivante :

```
(|(attr1=~motif)(attr2~motif)...) )
```

Il est également possible de paramétrer la manière dont sera composé le filtre de recherche attribut par attribut à l'aide des paramètres *searchLSformat* et *approxLSformat*.

Important : Ces filtres, une fois composés, sont insérés dans un autre, filtrant en plus sur les *ObjectClass* du type d'*LSubject* de la manière suivante :

```
(& (&(objectclass=oc1)(objectclass=oc2)) (filtre) )
```

Paramètres des attributs

`searchLSformat`

Ce paramètre est un LSformat permettant de définir, attribut par attribut, comment le filtre de recherche LDAP est composé à partir d'un motif de recherche et en cas de recherche non-approximative.

Ce LSformat est composé à l'aide des éléments *name*, le nom de l'attribut et *pattern*, le motif de recherche.

Exemple

```
(%{name}=%{pattern})
```

Important : Le filtre déduit doit obligatoirement commencer par (et se terminer par).

approxLSformat

Ce paramètre est un LSformat permettant de définir, attribut par attribut, comment le filtre de recherche LDAP est composé à partir d'un motif de recherche et en cas de recherche approximative.

Ce LSformat est composé à l'aide des éléments *name*, le nom de l'attribut et *pattern*, le motif de recherche.

Exemple

```
(%{name} =~ %{pattern})
```

Important : Le filtre déduit doit obligatoirement commencer par (et se terminer par).

params

Tableau des paramètres par défaut d'une recherche. Ce tableau contient les paramètres qui seront utilisés pour initialiser une recherche. Ces paramètres pourront être redéfini par l'utilisateur ou par l'application en fonction du contexte dans lequel cette recherche est effectuée.

Paramètres de configuration

pattern

Mot clé de la recherche.

sizelimit

Entier déterminant le nombre maximum d'objet pouvant être retournés dans une recherche.

recursive

Booléen déterminant si la recherche récursive est activée.

approx

Booléen déterminant si la recherche approximative est activée.

withoutCache

Booléen déterminant si le cache de recherche doit être utilisé.

onlyAccessible

Booléen déterminant si seul les objets accessibles à l'utilisateur connecté doivent être retournés par la recherche.

sortBy

Mot clé déterminant sur quel valeur/colonne le résultat de recherche sera trié.

Valeurs possibles : `displayName`, `subDn` ou `NULL`.

sortDirection

Mot clé déterminant le sens du trie du résultat de la recherche.

Valeurs possibles : `ASC`, `DESC` ou `NULL`.

sortlimit

Entier déterminant le nombre maximum d'objet pouvant être triés dans le résultat d'une recherche.

displayFormat

LSformat d'affichage du nom de l'objet dans le résultat de la recherche.

nbObjectsByPage

Entier déterminant le nombre d'objet maximum affichés dans une page de résultat de la recherche.

nbObjectsByPageChoices

Tableau des choix proposés à l'utilisateur pour le nombre d'objets maximum affichés dans une page de résultat de la recherche.

validPatternRegex

Expression régulière de validation des mots clés de recherche pour ce type d'LSobject.

(Par défaut : `/^[\\w`

`\\-_\\ \\ \\ ' \\ \" ^ \\ [\\] \\ (\\) \\ { \\ } \\ = \\ + \\ £ \\ % \\ $ \\ € \\ . \\ : \\ ; \\ , \\ ? \\ / \\ @ \\] + \\ $ / i u`)

predefinedFilters

Tableau associatif contenant des filtres prédéfinis pour la recherche. Les clés sont les filtres au format LDAP et les valeurs sont les labels associés.

extraDisplayedColumns

Tableau associatif contenant des colonnes supplémentaires à afficher dans les résultats de recherche. Les clés sont les identifiants des colonnes supplémentaires et les valeurs sont leur configuration définie à partir des paramètres suivant :

label

Le label de la colonne.

LSformat

Le LSformat d’affichage de la colonne. Ce format est composé à partir des attributs des objets LDAP dans leur format brut.

alternativeLSformats

Tableau des LSformats alternatifs à utiliser si le résultat du format principal est vide. Les formats définis dans cette liste sont essayés les uns après les autres et le premier LSformat retournant une valeur non-vide est utilisé.

formaterLSformat

LSformat optionnel permettant de mettre en forme le résultat obtenu des LSformats précédents. Ce LSformat ne sera utilisé que si le résultat obtenu précédemment n’est pas vide. Il est ainsi possible d’utiliser les paramètres `LSformat` et `alternativeLSformats` afin de récupérer la valeur à afficher, puis de la mettre en forme grâce à ce LSformat. Ce format est composé à partir des attributs des objets LDAP dans leur format brut et de la valeur retournés précédemment accessible via la variable `val`.

formaterFunction

Le nom d’une fonction optionnelle à exécuter pour mettre en forme le résultat obtenu des LSformats précédents. Cette fonction ne sera appelée que si le résultat obtenu précédemment n’est pas vide. La fonction prendra en paramètre la valeur à mettre en forme et retournera la valeur mise en forme.

generateFunction

Le nom d’une fonction qui sera utilisée pour générer la valeur d’affichage de cette colonne. La fonction prendra en paramètre une référence de l’objet `LSsearchEntry` et retournera la valeur de la colonne.

additionalAttrs

Un tableau de nom d’attributs à inclure dans le resultat de la recherche LDAP. Ce tableau permet notamment d’inclure les attributs nécessaires au bon fonctionnement de la fonction `generateFunction`.

escape

Ce paramètre booléen permet de définir si, lors de l’affichage, le contenu de la colonne doit être transformé pour protéger les caractères éligibles en entités HTML. Par défaut, ce paramètre est `Vrai`.

Avertissement

Cette fonctionnalité existe pour des raisons de sécurité et notamment en protection des failles `xss`. Si vous désactivez cette fonctionnalité, il est important de gérer la problématique de sécurité par ailleurs.

cssStyle

Ce paramètre permet de définir un style CSS personnalisé pour la colonne. S'il est défini, le contenu de ce paramètre sera ajouté en tant qu'attribut `style` des balises `th` et `td` de la colonne.

visibleTo

Ce paramètre permet de restreindre la visibilité de cette colonne aux seuls `LSprofiles` spécifiés. S'il est omis, la colonne sera visible pour tous.

customActions

Tableau associatif contenant les paramètres de configuration des `customActions`. Voir la section concernée.

4.2.7.1. customActions

Cette section décrit la manière de configurer les actions personnalisées exécutables sur les recherches d'`LSobjects` appelées `customActions`.

Structure

```
$GLOBALS['LSobjects']['[nom du type d'LSobject]']['LSsearch']['customActions'] = array (
  'action1' => array(
    'label' => '[label l'action]',
    'hideLabel' => '[booléen]',
    'icon' => '[nom de l'icône de l'action]',
    'function' => '[fonction à exécuter]',
    'question_format' => '[LSformat de la question de confirmation]',
    'onSuccessMsgFormat' => '[LSformat du message à afficher en cas de succès de l'action]',
    'disableOnSuccessMsg' => '[booléen]',
    'noConfirmation' => '[booléen]',
    'redirectToObjectList' => '[booléen]',
    'rights' => array(
      'LSprofile1',
      'LSprofile2',
      ...
    )
  )
)
```

);

Paramètres de configuration

label

Le label de l'action.

hideLabel

Cache le label dans le bouton de l'action.

icon

Nom de l'icône à afficher dans le bouton de l'action. Ce nom correspond au nom du fichier de l'image (sans l'extension) qui devra se trouver dans le dossier */src/images/[nom du theme d'images]/* ou dans le dossier *src/local/images*.

function

Le nom de la fonction à exécuter qui implémente l'action personnalisée. Cette fonction prendra en seule paramètre l'objet *LSsearch* sur lequel l'action devra être exécutée et retournera *True* en cas de succès ou *False* en cas d'échec d'exécution de la fonction.

question_format

Le LSformat de la question de confirmation d'exécution de l'action. Ce LSformat sera composé à l'aide du label de l'action.

onSuccessMsgFormat

Le LSformat du message à afficher en cas de succès d'exécution de l'action. Ce LSformat sera composé à l'aide du label de l'action.

disableOnSuccessMsg

Booléen permettant de désactiver le message afficher en cas de succès d'exécution de l'action.

noConfirmation

Booléen permettant de désactiver la confirmation de l'exécution de l'action.

redirectToObjectList

Booléen permettant de rediriger ou non l'utilisateur vers la liste des objets (Vrai par défaut). Si l'utilisateur n'est redirigé, le template par défaut (ou celui défini durant l'exécution de la fonction) sera affiché.

rights

Tableau contenant la liste des noms des LSprofiles ayant le droit d'exécuter cette action.

4.2.7.1.1. Ecriture d'une fonction implémentant une customAction

Une fonction implémentant une *customAction* se déclare de la manière suivante :

```

/*
 * Ma fonction implémentant ma customAction
 *
 * Paramètre :
 *   - $search : L'objet LSsearch de la recherche sur lequel mon action doit être exécuté
 *
 * Valeurs retournées :
 *   - True : Tout s'est bien passé
 *   - False : Une erreur est survenue
 */
function maFonction ($search) {

    // Actions

}

```

Cette fonction doit prendre pour seul paramètre, l'objet LSsearch sur lequel l'action personnalisée doit être exécutée et doit retourner soit `True` si tout s'est bien passé, soit `False` en cas de problème.

Important : La recherche passée en paramètre n'a pas encore été exécutée. En conséquence, si vous avez besoin d'accéder au résultat de la recherche, il est nécessaire d'exécuter au préalable : `$search -> run();`. Cela permet en outre, de modifier les paramètres de la recherche avant de l'exécuter. Cela peut par exemple être utile, si vous avez besoin d'accéder aux valeurs d'attributs particuliers, d'ajouter des attributs au résultat de la recherche : `$search -> setParam('attributes', array('attr1', 'attr2'));`.

Note : Ces fonctions sont le plus couramment définies au sein d'LSaddon.

4.2.8. ioFormat

Cette section décrit la manière de paramétrer les formats d'import/export pour un type d'LSobject donné.

La configuration des *ioFormats* se situe dans la configuration des LSobjects, dans la variable `ioFormat` (`$GLOBALS['LSobjects'][$nom du type d'LSobject]['ioFormat']`). Cette variable est un tableau associatif dont la clé est l'identifiant du format et dont la valeur associée est la configuration du format.

Important : Le moteur d'importation simule la validation d'un formulaire de création du type d'LSobject. En conséquence :

- seul les attributs présent dans le formulaire de création peuvent être importés.
- tous les attributs obligatoires présents dans le formulaire de création doivent être fournis par le fichier source ou générer à partir des autres attributs.
- Les valeurs des attributs issus de l'importation seront vue comme des valeurs retournées par le formulaire et non comme des valeurs des attribus LDAP eux-même. Ainsi et par exemple, un attribut traité comme un booléen dans un formulaire pourra prendre comme valeur par défaut `yes` ou `no`.

```
Structure
$GLOBALS['LSubjects']['[nom du type d'LSubject]']['ioFormat'] = array (
  '[ioFormat ID]' => array (
    'label' => '[Label du type de fichier]',
    'driver' => '[Pilote d'ioFormat utilisé]',
    'driver_options' => array([Options du pilote d'ioFormat utilisé]),
    'fields => array (
      '[champ 1]' => '[attribut 1]',
      '[champ 2]' => '[attribut 2]',
      [...]
    ),
    'generated_fields' => array (
      '[attribute 3]' => '[LSformat]',
      '[attribute 4]' => '[LSformat]',
      [...]
    )
  ),
  [...]
);
```

Paramètres de configuration

label

Le label du format

driver

Le pilote a utilisé pour ce format. Le pilote permet de gérer la lecture et l'écriture dans un type de fichier d'import/export. Pour plus d'information sur les pilotes disponibles, Voir la section concernée.

driver_options

Tableau associatif des options du pilote utilisé pour ce format. Pour plus d'informations, consulter la documentation du pilote utilisé.

fields

Tableau associatif permettant d'associer un champ du fichier source (la clé) avec attribut de l'objet LDAP (la valeur).

generated_fields

Tableau associatif permettant de définir des LSformats pour générer des valeurs d'attributs automatiquement. Ce tableau contient en clé, le nom de l'attribut à généré, et en valeur associée, le LSformat à utilisé. Ce LSformat est composé à l'aide des valeurs des autres attributs de l'objet.

4.2.8.1. Pilote d'ioFormat

Cette section décrit la manière de configurer les pilotes d'ioFormat utilisés lors des imports/exports d'LSobjects.

4.2.8.1.1. Pilote de fichiers CSV

Ce pilote permet de gérer l'import/export de LSobject à partir d'un fichier CSV. Depuis la version 4 d'LdapSaisie, ce pilote utilise les fonctions standards `fgetcsv()` et `fputcsv` fournis par PHP. Avant cela, la classe PEAR File_CSV_DataSource (http://pear.php.net/package/File_CSV_DataSource) était utilisée. Par défaut, les paramètres de lecture et d'écriture des fichiers sont : la virgule sert de délimiteur, le caractère " peut être utilisé pour encadrer les valeurs des champs et la longueur maximale d'une ligne est infini. Ces paramètres peuvent être modifiés en configurant les options du pilote.

Structure

```
$GLOBALS['LSobjects']['[nom du type d'LSobject]']['ioFormat']['[ID ioFormat]']['driver_options'] = array(
    'delimiter' => '[délimiteur]',
    'enclosure' => '[caractère d'encadrement de texte]',
    'length' => [longueur maximale d'une ligne],
    'escape' => '[caractère d'échappement]',
    'multiple_value_delimiter' => '[délimiteur]',
);
```

Paramètres de configuration

delimiter

Le caractère utilisé pour délimiter les champs (Par défaut, une virgule).

length

La longueur maximale d'une ligne du fichier. Si zéro est spécifié, la longueur d'une ligne ne sera pas limité, mais la lecture du fichier sera ralentie. (Par défaut : 0)

enclosure

Le caractère utilisé pour encadrer les valeurs des champs (Par défaut : ").

escape

Le caractère d'échappement utilisé si un des champs d'une ligne de fichier contient le caractère utilisé pour encadrer les valeurs. (Par défaut : \).

Note : Selon la RFC4180, l'échappement du caractère utilisé pour encadrer les valeurs des champs doit se faire en le doublant. Le caractère défini ici est une alternative à ce comportement par défaut. Pour désactiver ce caractère d'échappement alternatif, il est possible depuis de la version 7.4.0 de PHP de mettre ici une chaîne vide.

`multiple_value_delimiter`

Le caractère utilisé pour délimiter au sein d'un champs, les valeurs multiples d'un attribut (Par défaut : |).

4.3. Configuration des LSaddons

Cette partie décrit la manière de configurer les différents LSaddons actuellement supportés par LdapSaisie. Ces addons peuvent avoir un fichier de configuration et il sera alors stocké dans le dossier `conf/LSaddons/` et portera le nom `config.LSaddons.[addon name].php`.

4.3.1. LSaddon_asterisk

Cet LSaddon est utilisé pour gérer les fonctionnalités spécifiques au serveur de téléphonie Asterisk. Cet LSaddon donne accès à une fonction permettant l'encodage d'un mot de passe au format spécifique attendu par Asterisk. Ce format est un hash MD5 d'une chaîne de caractère composée du nom d'utilisateur, d'une chaîne fixe spécifiée dans la configuration d'Asterisk et du mot de passe en clair.

4.3.2. LSaddon_exportSearchResultAsCSV

Cet LSaddon fournit une fonction du même nom pouvant être utilisée comme `customActions` et permettant de télécharger le résultat d'une recherche au format CSV. L'export généré reprend exactement le contenu des colonnes du tableau du résultat de la recherche. Le DN de l'objet LDAP correspondant est également fournis dans une colonne.

Des paramètres de configuration sont disponibles dans le fichier de configuration `config.LSaddons.exportSearchResultAsCSV.php`. Ils permettent notamment de contrôler le format du fichier CSV généré.

```
Structure du fichier
// CSV file delimiter
define('LS_EXPORTSEARCHRESULTASCsv_DELIMITER', ',');
```

```
// CSV file enclosure
define('LS_EXPORTSEARCHRESULTASCSV_ENCLOSURE','');

// CSV file escape character (available since PHP 5.5.4)
define('LS_EXPORTSEARCHRESULTASCSV_ESCAPE_CHAR','\\');
```

Ci-dessous, vous trouverez un exemple de configuration de la fonction `exportSearchResultAsCSV()` comme `customActions` :

```
Exemple d'utilisation$GLOBALS['LSobjects']['LSpeople']['LSsearch'] = array (
    [...]
    'customActions' => array (
        'exportSearchResultAsCSV' => array (
            'label' => 'Export result as CSV',
            'icon' => 'export_csv',
            'function' => 'exportSearchResultAsCSV',
            'noConfirmation' => true,
            'disableOnSuccessMsg' => true,
            'rights' => array (
                'admin'
            )
        )
    ),
    [...]
);
```

Note : Le label et l'icône fournis dans cet exemple sont traduits et délivrés avec `LdapSaisie`.

4.3.3. LSaddon_LSaccessRightsMatrixView

Cet `LSaddon` offre une interface de visualisation des droits d'accès des différents `LSprofiles` configurés. Pour chaque type d'objet, la matrice des droits d'accès par attribut et par profil est affichée sous la forme d'un tableau.

Le fichier de configuration permet de définir au travers la variable `$GLOBALS['LSaccessRightsMatrixView_allowed_LSprofiles']` la liste des `LSprofiles` autorisés à accéder à cette interface.

4.3.4. LSaddon_mail

Cet LSaddon est utilisé pour gérer l'envoi de mail. Il utilise pour cela les bibliothèques PEAR (<http://pear.php.net/>) *Mail* et *Mail_Mime* qui doivent être installés. Cet LSaddon doit être configuré en éditant son fichier de configuration `config.LSaddons.mail.php`.

```
Structure du fichier/*
*****
* Configuration du support de l'envoi de mail *
*****
*/

// Pear :: Mail
define('PEAR_MAIL', '/usr/share/php/Mail.php');

// Pear :: Mail_mime
define('PEAR_MAIL_MIME', '/usr/share/php/Mail/mime.php');

/*
* Méthode d'envoi :
* - mail : envoie avec la méthode PHP mail()
* - sendmail : envoie la commande sendmail du système
* - smtp : envoie en utilisant un serveur SMTP
*/
define('MAIL_SEND_METHOD', 'smtp');

/*
* Paramètres d'envoi :
* Ces paramètres dépendent de la méthode utilisée. Reportez-vous à la documentation
* de PEAR :: Mail pour plus d'informations.
* Lien : http://pear.php.net/manual/en/package.mail.mail.factory.php
* Infos :
* List of parameter for the backends
* mail
*   o If safe mode is disabled, $params will be passed as the fifth
*     argument to the PHP mail() function. If $params is an array,
*     its elements will be joined as a space-delimited string.
* sendmail
*   o $params["sendmail_path"] - The location of the sendmail program
*     on the filesystem. Default is /usr/bin/sendmail.
*   o $params["sendmail_args"] - Additional parameters to pass to the
*     sendmail. Default is -i.
* smtp
*   o $params["host"] - The server to connect. Default is localhost.
*   o $params["port"] - The port to connect. Default is 25.
*   o $params["auth"] - Whether or not to use SMTP authentication.
*     Default is FALSE.
*   o $params["username"] - The username to use for SMTP authentication.
*   o $params["password"] - The password to use for SMTP authentication.
*   o $params["localhost"] - The value to give when sending EHLO or HELO.
*     Default is localhost
*   o $params["timeout"] - The SMTP connection timeout.
```

```

*      Default is NULL (no timeout).
*      o $params["verp"] - Whether to use VERP or not. Default is FALSE.
*      o $params["debug"] - Whether to enable SMTP debug mode or not.
*      Default is FALSE.
*      o $params["persist"] - Indicates whether or not the SMTP connection
*      should persist over multiple calls to the send() method.
*/
$MAIL_SEND_PARAMS = NULL;

/*
* Headers :
*/
$MAIL_HEADERS = array(
);

```

Cet LSaddon offre la possibilité d'utiliser la fonction PHP (<http://www.php.net/>) `sendMail()`.

```

bool sendMail(string $to, string $subject, string $msg, array $headers, array
$attachments, string $eol, string $encoding, boolean $html);

```

4.3.5. LSaddon_maildir

Cet LSaddon est utilisé pour gérer la manipulation distante de maildir. FIXME

4.3.6. LSaddon_mailquota

Cet LSaddon fournit une fonction `mailquota_get_usage` pouvant être utilisée pour récupérer l'utilisation du quota d'une boîte mail IMAP. Pour cela, LdapSaisie se connecte au serveur IMAP en utilisant un compte maître.

Cet LSaddon fournit également une fonction `mailquota_show_usage` pouvant être utilisée comme `customActions` et permettant d'afficher l'utilisation du quota de la boîte mail correspondante via un message dynamique (`LSinfo`).

Des paramètres de configuration sont disponibles dans le fichier de configuration `config.LSaddons.mailquota.php`.

```

Structure du fichier
// IMAP Mailbox connection string LSformat (composed with LSldapObject attributes)
// See : https://php.net/imap\_open (parameter $mailbox)
define('MAILQUOTA_IMAP_MAILBOX', '{localhost}');

```

```

// IMAP Master user
define('MAILQUOTA_IMAP_MASTER_USER', 'ldapsaisie');

// IMAP Master user's password
define('MAILQUOTA_IMAP_MASTER_USER_PWD', 'secret');

// IMAP Master user LSformat composed with :
// * masteruser = master username (MAILQUOTA_IMAP_MASTER_USER)
// * LSldapObject attributes
define('MAILQUOTA_IMAP_MASTER_USER_FORMAT', '%{mail}*%{masteruser}');

// IMAP quota root mailbox
define('MAILQUOTA_IMAP_QUOTA_ROOT_MAILBOX', 'INBOX');

```

Ci-dessous, vous trouverez un exemple de configuration de la fonction `mailquota_show_usage()` comme `customActions`

```

Exemple d'utilisation$GLOBALS['LSobjects']['LSpeople'] = array (
    [...]
    'customActions' => array (
        'showmailquotausage' => array (
            'function' => 'mailquota_show_usage',
            'label' => 'Show mail quota usage',
            'noConfirmation' => true,
            'disableOnSuccessMsg' => true,
            'icon' => 'mail',
            'rights' => array (
                'admin'
            )
        ),
        [...]
    ),
    [...]
);

```

4.3.7. LSaddon_phpldapadmin

Cet LSaddon est utilisé pour permettre un lien facile entre le logiciel `PhpLdapAdmin` (<http://phpldapadmin.sourceforge.net>) et `LdapSaisie`. Il sera possible ainsi à partir d'un objet dans `LdapSaisie` de voir ce même objet dans `PhpLdapAdmin` (<http://phpldapadmin.sourceforge.net>).

Il est nécessaire de configurer l'URL de votre installation de `PhpLdapAdmin` (<http://phpldapadmin.sourceforge.net>) dans le fichier de configuration `config.LSaddons.phpldapadmin.php`.

Structure du fichier

```

// PhpLdapAdmin View Object URL format
define('LS_PHPLDAPADMIN_VIEW_OBJECT_URL_FORMAT', '///'.$_SERVER['SERVER_NAME'].'/phpldapadmin

```

Cet LSaddon offre la possibilité d'utiliser la fonction PHP (<http://www.php.net/>) `redirectToPhpLdapAdmin()` comme `customActions`.

```
bool redirectToPhpLdapAdmin(LSldapObject $ldapObject);
```

```
Exemple d'utilisation$GLOBALS['LSobjects']['LSpeople'] = array (
    [...]
    'customActions' => array (
        'redirectToPhpLdapAdmin' => array (
            'function' => 'redirectToPhpLdapAdmin',
            'label' => 'See in PhpLdapAdmin',
            'hideLabel' => True,
            'noConfirmation' => true,
            'disableOnSuccessMsg' => true,
            'icon' => 'phpldapadmin',
            'rights' => array (
                'admin'
            )
        ),
    ),
    [...]
);
```

4.3.8. LSaddon_showTechInfo

Cet LSaddon fournit une fonction du même nom pouvant être utilisée comme `customActions` et permettant d'afficher les informations techniques d'un objet de l'annuaire.

Ci-dessous, vous trouverez un exemple de configuration de la fonction `showTechInfo()` comme `customActions` :

```
Exemple d'utilisation$GLOBALS['LSobjects']['LSpeople'] = array (
    [...]
    'customActions' => array (
        'showTechInfo' => array (
            'function' => 'showTechInfo',
            'label' => 'Show technical information',
            'hideLabel' => True,
            'noConfirmation' => true,
            'disableOnSuccessMsg' => true,
            'icon' => 'tech_info',
            'rights' => array (
                'admin'
            )
        ),
    ),
    [...]
);
```

```
[...]
);
```

Note : Le label et l'icône fournis dans cet exemple sont traduits et délivrés avec LdapSaisie.

4.4. Configuration des LSauthMethods

Cette partie décrit la manière de configurer les méthodes d'authentification d'LdapSaisie appelée LSauthMethod. Ces bibliothèques peuvent avoir un fichier de configuration et il sera alors stocké dans le dossier `conf/LSauth/`.

4.4.1. LSauthMethod_HTTP

Cette LSauthMethod est utilisée pour gérer l'authentification via les variables d'environnements définies suite à une authentification, potentiellement déléguée au serveur web.

Cette méthode récupère dans l'environnement d'exécution PHP, le nom d'utilisateur et le mot de passe de l'utilisateur connecté. À partir du nom d'utilisateur, une recherche dans l'annuaire sera effectuée pour trouver l'utilisateur correspondant. L'authentification sera réussie uniquement si un et un seul utilisateur est retourné par la recherche et si une authentification auprès de l'annuaire LDAP réussie à l'aide du DN de l'objet LDAP trouvé et du mot de passe fourni.

Note : En cas d'authentification déléguée au serveur web, il est possible de désactiver la vérification du mot de passe via le paramètre `LSAUTHMETHOD_HTTP_TRUST_WITHOUT_PASSWORD_CHALLENGE` (voir ci-dessous).

Les variables d'environnements utilisées pour authentifier l'utilisateur connecté dépendent de la méthode configurée via la constante `LSAUTHMETHOD_HTTP_METHOD` (voir ci-dessous). Si ces variables ne sont pas disponibles, une erreur HTTP 403 sera générée pour réclamer une authentification à l'utilisateur.

Note : Cette LSauthMethod supporte le mode API et il s'agit de la méthode utilisée par défaut dans ce mode.

Cette bibliothèque peut être configurée en éditant le fichier de configuration `conf/LSauth/config.LSauthMethod_HTTP.php`.

*Structure du fichier/**

```

*****
* Configuration of the HTTP authentication support *
*****
*/

// Don't check HTTP server's login/password by LDAP authentication challenge
//define('LSAUTHMETHOD_HTTP_TRUST_WITHOUT_PASSWORD_CHALLENGE',true);

// Authentication realm (API mode only)
//define('LSAUTHMETHOD_HTTP_API_REALM', ___('LdapSaisie API - Authentication required'));

```

Paramètres de configuration

LSAUTHMETHOD_HTTP_TRUST_WITHOUT_PASSWORD_CHALLENGE

Permet de désactiver le test d'authentification auprès de l'annuaire LDAP. Pour cela, cette constante doit être définie et valoir `True`.

LSAUTHMETHOD_HTTP_METHOD

Permet de définir la méthode utilisée par le serveur web pour passer à PHP l'identifiant de l'utilisateur connecté et son mot de passe.

Cette constance peut prendre les valeurs suivantes :

PHP_PASS

Dans cette méthode, le serveur web définit les variables d'environnement `PHP_AUTH_USER` et `PHP_AUTH_PW`. Cette méthode est la méthode par défaut et convient en cas d'utilisation de `mod_php`.

REMOTE_USER

Dans cette méthode, le serveur web définit la variable d'environnement `REMOTE_USER`. Cette variable ne contient que l'identifiant de l'utilisateur connecté. Cette méthode ne peut donc être utilisée que conjointement avec l'activation du paramètre `LSAUTHMETHOD_HTTP_TRUST_WITHOUT_PASSWORD_CHALLENGE`.

AUTHORIZATION

Dans cette méthode, le serveur web passe le contenu de l'entête `HTTP Authorization` dans la variable d'environnement `HTTP_AUTHORIZATION`. Cette méthode convient en cas d'utilisation de PHP en mode CGI ou encore via PHP-FPM.

Pour utiliser cette méthode, il faudra adapter la configuration du serveur web. Par exemple, pour Apache HTTPd, vous pouvez utiliser le module `rewrite` et la règle de réécriture suivante :

```
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
```

LSAUTHMETHOD_HTTP_LOGOUT_REMOTE_URL

URL de déconnexion externe, utile par exemple dans le contexte d'une connexion via un service SSO. L'utilisateur sera automatiquement redirigé vers cette URL après sa déconnexion effective au niveau d'LdapSaisie.

Note : Si cette URL de déconnexion n'est pas défini, le bouton de déconnexion sera masqué.

LSAUTHMETHOD_HTTP_REALM

Domaine d'authentification (`realm`) utilisé pour réclamer l'authentification de l'utilisateur (facultatif).

Note : Pour que le message soit traduit, utilisez la fonction `__()` (voir exemple).

4.4.2. LSauthMethod_CAS

Cette LSauthMethod est utilisée pour gérer l'authentification via un service SSO CAS (<http://www.jasig.org/cas>). Cette librairie doit être configurée en éditant le fichier de configuration `conf/LSauth/config.LSauthMethod_CAS.php`.

*Structure du fichier/**

```
*****
* Configuration of the CAS authentication support *
*****
*/
```

```
// phpCAS Path (http://www.ja-sig.org/wiki/display/CASC/phpCAS)
define('PHP_CAS_PATH', '/usr/share/php/CAS.php');
```

```
// phpCAS Debug File
// define('PHP_CAS_DEBUG_FILE', '/tmp/phpCAS.log');
```

```
// Disable logout
define('LSAUTH_CAS_DISABLE_LOGOUT', false);
```

```
// CAS Server version (used constant name know by phpCAS : CAS_VERSION_1_0 or CAS_VERSION_2_0)
define('LSAUTH_CAS_VERSION', 'CAS_VERSION_2_0');
```

```
// CAS Server hostname
```

```
define('LSAUTH_CAS_SERVER_HOSTNAME', 'cas.univ.fr');

// CAS Server port
define('LSAUTH_CAS_SERVER_PORT', 443);

// CAS Server URI (empty by default)
// define('LSAUTH_CAS_SERVER_URI', 'cas/');

// No SSL validation for the CAS server
define('LSAUTH_CAS_SERVER_NO_SSL_VALIDATION', false);

// CAS server SSL CA Certificate path
//define('LSAUTH_CAS_SERVER_SSL_CACERT', "");
```

Paramètres de configuration

PHP_CAS_PATH

Le chemin d'accès du fichier `CAS.php` de la librairie `phpCAS` (<http://www.ja-sig.org/wiki/display/CASC/phpCAS>). Le chemin d'exemple correspond au chemin résultant d'une installation via PEAR sur une Debian (Lenny).

PHP_CAS_DEBUG_FILE

Chemin du fichier de log de la librairie `phpCAS` (<http://www.ja-sig.org/wiki/display/CASC/phpCAS>). Commenter la ligne pour désactiver les logs.

LSAUTH_CAS_DISABLE_LOGOUT

Booléen définissant si l'utilisateur peut se déconnecter du serveur CAS (<http://www.jasig.org/cas>) depuis l'interface.

Note : Remarque : l'appel de l'URL de déconnexion via une requête `GET` supprimera la session PHP (<http://www.php.net/>) et donc la session `LdapSaisie` sans déconnecter pour autant l'utilisateur au niveau du serveur CAS (<http://www.jasig.org/cas>). Cela peut donc permettre de gérer la déconnexion automatique au niveau de `LdapSaisie` suite à une déconnexion au niveau du CAS à travers le concept de `Global Logout`.

LSAUTH_CAS_VERSION

Nom de la constante `phpCAS` (<http://www.ja-sig.org/wiki/display/CASC/phpCAS>) permettant de définir la version CAS du serveur. Actuellement, la librairie `phpCAS` (<http://www.ja-sig.org/wiki/display/CASC/phpCAS>) ne reconnaît que la constante `CAS_VERSION_1_0` pour la version 1 de CAS ou la constante `CAS_VERSION_2_0` pour la version 2 de CAS.

Note : Remarque : Des tests ont montrés que l'utilisation d'une compatibilité CAS version 2 peut également fonctionner sur un version 3 du serveur CAS.

LSAUTH_CAS_SERVER_HOSTNAME

Le nom d'hôte du serveur CAS (<http://www.jasig.org/cas>).

LSAUTH_CAS_SERVER_PORT

Le port d'écoute du serveur CAS (<http://www.jasig.org/cas>).

LSAUTH_CAS_SERVER_URI

Le dossier HTTP dans lequel se trouve le service CAS (<http://www.jasig.org/cas>). Exemple : Pour un service CAS accessible via l'URL <https://cas.univ.fr/cas/>, la constante devra valoir `cas/`.

LSAUTH_CAS_SERVER_NO_SSL_VALIDATION

Booléen permettant de désactiver la validation du certificat SSL du serveur CAS lors des requêtes de validation des tickets CAS.

LSAUTH_CAS_SERVER_SSL_CACERT

Chemin d'accès du fichier contenant le certificat SSL de la CA du serveur CAS au format PEM. Commenter la ligne pour désactiver ce paramètre.

4.4.3. LSauthMethod_anonymous

Cette LSauthMethod est utilisée pour gérer l'authentification automatique des utilisateurs arrivant (équivalent à un mode anonyme). Cette librairie doit être configurée en éditant le fichier de configuration `conf/LSauth/config.LSauthMethod_anonymous.php` et notamment en définissant la constante `LSAUTHMETHOD_ANONYMOUS_USER` contenant le login d'un utilisateur dont les droits d'accès seront endossés par tout les personnes utilisant LdapSaisie.

Chapitre 5. API

Depuis la version 4.0, LdapSaisie offre une API visant à permettre de faire les mêmes choses que ce qu'il est possible d'accomplir via l'interface web. L'idée n'est bien entendue pas de se substituer systématiquement à la possibilité de se connecter directement à l'annuaire, mais plutôt d'offrir une API web pour l'intégration d'outil préférant ce mode d'interaction, ou encore, pour exposer des méthodes accès aux données de l'annuaire tout en profitant des logiques métiers implémentées/configurées dans LdapSaisie : validation syntaxique et d'unicité, règle de génération et d'interdépendances des attributs, déclencheurs, ...

Note : Cette API est actuellement dans une phase de test et n'offre pas encore toutes les fonctionnalités proposées dans l'interface web. Elle est vouée à évoluer pour intégrer petit à petit un maximum de fonctionnalités. Des contributions à ce sujet seront plus qu'appréciées !

5.1. Authentification

L'authentification à l'API utilise le même composant `LSauth` que lors d'une authentification à l'interface web, cependant, ce composant s'adapte pour prendre en compte de mode de connexion. Par défaut, la méthode d'authentification utilisée sera `LSauthMethod_HTTP` et permettra de se connecter en spécifiant le nom d'utilisateur et le mot de l'utilisateur cherchant à se connecter via une authentification basique HTTP.

Avertissement

Il est à noter que tous les types d'utilisateur ne peuvent pas forcément utiliser l'API : le paramètre `api_access` doit être explicitement positionné à `True` dans la configuration du serveur LDAP.

Une fois connecté, l'utilisateur endossera les droits associés à ses `LSprofiles`, tout comme un utilisateur connecté à l'interface web.

5.2. Méthodes exposées

Les URLs des méthodes de l'API ont été construites par mimétisme sur celle de l'interface web et sous la racine `web/api/`. Par ailleurs, un numéro de version d'API a été insérée dans chacune d'elles afin d'anticiper toutes évolutions futures majeures nécessitant de conserver une rétrocompatibilité avec les anciennes versions de l'API.

Toutes les méthodes retournent des informations au format JSON et accepte le paramètre `pretty` permettant d'obtenir un retour plus facilement lisible. Les chaînes de caractères échangées doivent par ailleurs être encodées en UTF-8. On trouvera par ailleurs dans le retour JSON :

`success`

Booléen précisant si l'action demandée a correctement été exécutée.

`messages`

Ce tableau pourra être présent et lister les messages d'informations générées par l'action demandée. Il s'agira des mêmes messages que ceux affichés dans l'interface web lorsque les actions équivalentes y sont faites.

`errors`

Ce tableau pourra être présent et lister les messages d'erreurs générées par l'action demandée.

Note : Les messages d'informations et d'erreurs générées par l'application sont traduites dans la langue courante qui peut être spécifiée via le paramètre `lang` accepté par toutes les méthodes (exemple : `fr_FR` ou `en_US`).

Lorsqu'une méthode cible un type d'objets, voir un objet en particulier, ces informations seront transmises dans l'URL appelée. Si le type d'objet ou l'objet demandé est introuvable, une erreur HTTP 404 sera générée.

Liste des méthodes exposées

`/api/1.0/object/[object type]`

Cette méthode permet de rechercher/lister les informations d'un type d'objets de l'annuaire en particulier. Le type de l'objet est précisé dans l'URL et doit être encodé en conséquence. Par mimétisme du comportement de l'interface web, la recherche est paginée et accepte des paramètres similaires en plus de paramètre plus appropriés à un fonctionnement programmatique :

Paramètres acceptés

`filter`

Permet de spécifier un filtre de recherche LDAP personnalisé. Celui-ci sera combiné avec les paramètres propres au type d'objets recherchés et aux autres paramètres spécifiés (`pattern` par exemple).

Avertissement

Pour le moment, seuls les filtres simples du type `attribut=valeur` sont acceptés ici.

predefinedFilter

Permet de spécifier un des filtres de recherche LDAP prédéfinis dans la configuration du type d'objet.

pattern

Permet de spécifier un mot clé de recherche, comme proposé dans l'interface web.

approx

Booléen permettant d'activer/désactiver la recherche approximative sur le mot clé. Les valeurs acceptées sont 1 ou 0.

basedn

Permet de spécifier une base de recherche personnalisé pour la recherche.

subDn

Dans le cas d'un serveur LDAP configuré avec des sous-niveaux de connexion, permet de spécifier le sous-niveau pour la recherche.

scope

Permet de spécifier l'étendue de la recherche dans l'annuaire. Valeurs acceptées: sub, one et base.

recursive

Booléen permettant d'activer/désactiver la recherche recursive, c'est à dire une recherche à la racine de l'annuaire (ou du sous-niveau de connexion) avec une étendue de recherche maximale. Les valeurs acceptées sont 1 ou 0.

displayFormat

Permet de spécifier un LSformat personnalisé pour le nom des objets dans le résultat de recherche.

extraDisplayedColumns

Booléen permettant d'activer le retour des colonnes personnalisées dans le résultat de recherche. Les valeurs acceptées sont 1 ou 0.

attributes

Liste des attributs supplémentaires que devra retourner la recherche.

Avertissement

Attention, ici ce sont les valeurs brutes des attributs qui seront retournées et par forcément les valeurs telle que retournées habituellement.

sortBy

Permet de préciser sur quelle information le résultat de recherche doit être trié. Valeurs acceptées : `displayName`, `subDn` ou un des noms des colonnes personnalisées.

sortDirection

Permet de préciser l'ordre de tri du résultat de recherche. Valeurs acceptées : `ASC (A-Z)` ou `DESC (Z-A)`.

page

Permet de préciser la page du résultat de recherche.

nbObjectsByPage

Permet de préciser le nombre maximum d'objets retournés par page du résultat de recherche.

all

Permet de réclamer le résultat complet de la recherche (désactivation de la pagination). Seul la présence de ce paramètre suffit à activer ce comportement, sa valeur n'a pas d'importance.

withoutCache

Booléen permettant de désactiver l'utilisation du cache. Les valeurs acceptées sont 1 ou 0.

Exemple

```
# curl -u username:secret 'https://ldapsaisie/api/1.0/object/LSpeople?extraDisplayedCol
{
  "success": true,
  "objects": {
    "uid=hmartin,ou=people,o=ls": {
      "name": "Henri MARTIN",
      "Mail": "henri.martin@ls.com"
    },
    "uid=s.ldapsaisie,ou=people,o=ls": {
      "name": "Secretariat LdapSaisie",
      "Mail": "secretariat@ldapsaisie.biz"
    },
    "uid=ls,ou=people,o=ls": {
      "name": "LdapSaisie",
      "Mail": "ldap.saisie@ls.com"
    },
    "uid=erwpa,ou=people,o=ls": {
      "name": "Erwan PAGE",
      "Mail": "erwan.page@ldapsaisie.biz"
    },
    "uid=user2,ou=people,ou=company1,ou=companies,o=ls": {
      "name": "prenom2 nom2",
      "Mail": "user2@ls.com"
    }
  },
  "total": 14,
```

```

    "page": 1,
    "nbPages": 3
  }

```

`/api/1.0/object/[object type]/[dn]`

Cette méthode permet de récupérer les informations d'un objet de l'annuaire au format JSON. Le type de l'objet et son DN sont précisés dans l'URL et doivent être encodés en conséquence. Par défaut, les valeurs des attributs retournées sont au format tel qu'attendu en cas de création/modification de l'objet. Il est cependant possible d'ajouter le paramètre `details` afin d'obtenir des informations complémentaires sur les valeurs des attributs.

Exemple

```

# curl -u username:secret 'https://ldapsaisie/api/1.0/object/LSpeople/uid=hmartin,ou=peo
{
  "success": true,
  "dn": "uid=hmartin,ou=people,o=ls",
  "type": "LSpeople",
  "name": "Henri MARTIN",
  "details": false,
  "attributes": {
    "uid": "hmartin",
    "givenName": "Henri",
    "sn": "MARTIN",
    "cn": "Henri MARTIN",
    "mail": "henri.martin@ls.com",
    "personalTitle": "M.",
    "description": [],
    "jpegPhoto": null,
    "lsGodfatherDn": [
      "uid=eeggs,ou=people,o=ls"
    ],
    "uidNumber": "101022",
    "gidNumber": "102001",
    "loginShell": "no",
    "homeDirectory": "\\home\\com",
    "gecos": null,
    "shadowExpire": null,
    "shadowMax": null,
    "shadowInactive": null,
    "shadowLastChange": null,
    "sambaSID": "S-1-5-21-2421470416-3566881284-3047381809-203044",
    "sambaPrimaryGroupSID": "S-1-5-21-2421470416-3566881284-3047381809-205003",
    "sambaAcctFlags": [
      "U"
    ],
    "sambaHomeDrive": null,
    "sambaHomePath": null,
    "sambaProfilePath": null,
    "sambaLogonScript": null,
    "sambaLogonTime": null,
    "sambaLogoffTime": null,
    "sambaKickoffTime": null,

```

```

        "sambaPwdLastSet": null,
        "sambaPwdMustChange": null,
        "sambaPwdCanChange": null
    },
    "relations": {
        "groups": {
            "cn=direction,ou=groups,o=ls": "direction",
            "cn=secretariat,ou=groups,o=ls": "secretariat"
        },
        "godfather": []
    }
}

```

/api/1.0/object/[object type]/create

Cette méthode permet de créer un objet dans l'annuaire. Le type de l'objet qui sera créé est précisé dans l'URL et doit être encodé en conséquence. Les informations de l'objet doivent être transmises au format `x-www-form-urlencoded`. Elles peuvent également être au format `multipart/form-data`, en particulier si votre requête contient une image. Par mimétisme avec l'interface web, seuls les attributs prévus dans le formulaire de création du type d'objet peuvent être passés ici. De la même manière, les attributs non-spécifiés ici, pourront être auto-générés en accord avec leur configuration et la requête sera acceptée uniquement si tous les attributs obligatoires y sont spécifiés ou s'ils peuvent être auto-générés.

Le format et la syntaxe des valeurs des attributs dépendent de leur type HTML. Ainsi, par exemple, un attribut de type HTML `boolean` acceptera comme valeurs possibles `yes` ou `no`. Pour plus de détails sur le type de valeur acceptée par un type d'attribut HTML en particulier, consultez sa documentation. Vous pouvez également analyser le code de la méthode `getPostData()` de la classe PHP correspondante.

Si l'application détecte un souci avec les informations transmises pour les attributs, un tableau `fields_errors` sera présent dans la réponse JSON et contiendra pour chacun des attributs problématique, un tableau des messages d'erreurs générées par l'application.

Si le type d'objet en prévoit, vous pouvez également utiliser un masque de saisie via le paramètre `dataEntryForm`.

Exemple

```

# curl -u username:secret 'https://ldapsaisie/api/1.0/object/LSpeople/create?pretty' -d
{
    "success": true,
    "type": "LSpeople",
    "dn": "uid=foo.bar,ou=people,o=ls",
    "name": "Foo Bar",
    "messages": [
        "Le mail de notification a \u00e9t\u00e9 envoy\u00e9.",
        "L'objet a \u00e9t\u00e9 ajout\u00e9."
    ]
}

```

`/api/1.0/object/[object type]/[dn]/modify`

Cette méthode permet de modifier un objet dans l'annuaire. Le type de l'objet et son DN sont précisés dans l'URL et doivent être encodés en conséquence. Les informations de l'objet à modifier doivent être transmises au même format que pour la méthode `create` (voir ci-dessus). Comme pour cette dernière, seuls les attributs prévus dans le formulaire de modification du type d'objet peuvent être passés ici et la réponse JSON pourra contenir un tableau `fields_errors` contenant les erreurs générées par l'application au sujet des valeurs transmises pour les attributs.

Exemple

```
# curl -u username:secret 'https://ldapsaisie/api/1.0/object/LSpeople/uid=foo.bar,ou=people'
{
  "dn": "uid=foo.bar,ou=people,o=ls",
  "type": "LSpeople",
  "name": "Foo Bar",
  "success": true,
  "messages": [
    "L'objet a bien \u00e9t\u00e9 modifi\u00e9."
  ]
}
```

`/api/1.0/object/[object type]/[dn]/remove`

Cette méthode permet de supprimer un objet dans l'annuaire. Le type de l'objet et son DN sont précisés dans l'URL et doivent être encodés en conséquence.

Exemple

```
# curl -u username:secret 'https://ldapsaisie/api/1.0/object/LSpeople/uid=foo.bar,ou=people'
{
  "dn": "uid=foo.bar,ou=people,o=ls",
  "type": "LSpeople",
  "name": "Foo Bar",
  "success": true,
  "messages": [
    "Foo Bar a bien \u00e9t\u00e9 supprim\u00e9."
  ]
}
```

`/api/1.0/object/[object type]/import`

Cette méthode permet d'importer des objets d'un type en particulier à partir de données d'import formatées selon un `ioFormat` configuré pour ce type d'objets. Le type de l'objet est précisé dans l'URL et doit être encodé en conséquence. Par mimétisme du comportement de l'interface web,

cette méthode accepte des paramètres similaires et s'attend à récupérer les données d'import dans le corps de la requête.

Paramètres acceptés

`ioFormat`

Le nom de l'`ioFormat` des données d'import.

`updateIfExists`

Booléen permettant d'activer/désactiver la mise à jour des données des objets s'ils existent déjà. Si ce mode est inactif et qu'un objet des données d'import existe déjà, une erreur sera remontée. Les valeurs acceptées sont 1 ou 0.

`justTry`

Booléen permettant d'activer/désactiver le mode de vérification des données d'import uniquement. Si ce mode est actif, les données d'import seront analysées pour vérifier qu'elles sont correctes, mais l'import en lui-même ne sera pas effectué. Les valeurs acceptées sont 1 ou 0.

Note : Le retour de cette méthode en mode `justTry` est identique à une exécution en mode normal. Ce mode permet donc d'anticiper le résultat d'un import à partir d'un jeu de données sources.

Avertissement

En mode `justTry`, seule la vérification syntaxique des données est fiable, car les informations doublonnées au sein des données d'import ne pourront être détectées.

En cas d'erreurs détectées dans les informations des objets des données d'import, le tableau `errors` du retour de la méthode contiendra une entrée pour chaque objet en erreur sous le format d'un dictionnaire dont la clé `data` reprendra les informations de l'objet telle que chargé (ou générée) depuis les données sources, ainsi qu'un dictionnaire sous la clé `errors` qui contiendra les erreurs globales concernant l'objet sous la clé `globals` et les erreurs propres à ses attributs dans un dictionnaire sous la clé `attrs`.

Note : Les erreurs d'importation sur un objet sont non-bloquantes : l'importation des autres objets ne sera pas interrompue.

Exemple

```
# curl -u username:secret --data-binary @/path/to/input.file 'https://ldapsaisie/api/1.0'
```

```

{
  "success": false,
  "LSobject": "LSpeople",
  "ioFormat": "mycsv",
  "updateIfExists": false,
  "justTry": false,
  "imported": {
    "uid=rturin,ou=people,o=ls": "M. Roger TURIN"
  },
  "updated": [],
  "errors": [
    {
      "data": {
        "uid": [
          "lmartin"
        ],
        "personalTitle": [
          "Mme"
        ],
        "givenName": [
          "Ludivine"
        ],
        "sn": [
          "MARTIN"
        ],
        "mail": [
          "lmartin@gmail.com"
        ],
        "userPassword": [
          "123Yh%uT"
        ],
        "gidNumber": [
          "102009"
        ],
        "loginShell": [
          "no"
        ],
        "cn": [
          "Mme Ludivine MARTIN"
        ]
      },
      "errors": {
        "globals": [
          "Un objet existe d\u00e9j\u00e0 dans l'annuaire LDAP avec le DN uid="
        ],
        "attrs": []
      }
    }
  ],
  "messages": [
    "Le mail de notification a \u00e9t\u00e9 envoy\u00e9."
  ]
}

```

/api/1.0/object/[object type]/export

Cette méthode permet d'exporter les objets d'un type en particulier dans un ioFormat configuré pour ce type d'objets. Le type de l'objet est précisé dans l'URL et doit être encodé en conséquence.

Paramètres acceptés

ioFormat

Le nom de l'ioFormat.

En tant normal, le retour de cette méthode sera directement le fichier d'export demandé. Cependant, si une erreur survient, les paramètres d'export seront repris dans le retour JSON de la méthode qui contiendra également les erreurs survenues.

Exemple

```
# curl -u username:secret --data-binary @/path/to/input.file 'https://ldapsaisie/api/1.0/object/ls?ioFormat=json;login;civility;firstname;name;mail;password;gid;shell
hmartin;M.;Henri;MARTIN;henri.martin@ls.com;*****;102001;no
s.ldapsaisie;M.;Secretariat;LdapSaisie;secretariat@ldapsaisie.biz;*****;70000;no
ls;M.;Ldap;Saisie;ldap.saisie@ls.com;*****;102001;no
erwpa;M.;Erwan;PAGEARD;erwan.page@ldapsaisie.biz;*****;102009;no
[...]
```

/api/1.0/object/[object type]/[dn]/relation/[relation]

Cette méthode permet de gérer les objets en relation avec un objet en particulier de l'annuaire. Le type de l'objet, son DN et le nom de la relation sont précisés dans l'URL et doivent être encodés en conséquence. Cette méthode accepte les paramètres `add` et `remove` permettant de lister le ou les DN d'objet(s) à respectivement ajouter ou supprimer parmi les objets actuellement en relation avec l'objet spécifié. Si aucun DN n'est spécifié comme devant être ajouté ou supprimé, la méthode retournera simplement les DN des objets en relation. En cas de modification demandée, la méthode retournera la nouvelle liste des DN des objets en relation, quel que soit le résultat de l'opération de mise à jour.

Exemple

```
# curl -u username:secret 'https://ldapsaisie/api/1.0/object/LSpeople/uid=foo.bar,ou=people,ou=groups,ou=ls?relation=groups&add=true'
{
  "dn": "uid=foo.bar,ou=people,o=ls",
  "type": "LSpeople",
  "name": "Foo Bar",
  "relation": "groups",
  "success": true,
  "relatedObjects": [
    "cn=ls,ou=groups,o=ls",
    "cn=invite,ou=groups,o=ls"
  ],
  "messages": [
    "Objects in relation updated."
  ]
}
```

}

Chapitre 6. Contribution

Comme tout projet libre qui se respecte, les contributions à LdapSaisie sont les bienvenues. Ce chapitre explique les possibilités de contribution.

6.1. LSaddons

Les LSaddons sont utilisés pour implémenter dans LdapSaisie des fonctionnalités spécifiques tel que :

- le support d'une famille d'attributs spécifiques (POSIX, Samba, SUPANN...) par le biais de méthodes de génération de la valeur de ces attributs par exemple (paramètre `generate_function`) ;
- des tâches communes et génériques (envoi de mails, connexion FTP/SSH...);
- l'implémentation de déclencheurs spécifiques à votre environnement : création automatique du dossier client sur le serveur de fichiers de l'entreprise, création de la boîte mail de l'utilisateur... ;
- l'implémentation de vues personnalisées proposées dans l'interface
- l'implémentation d'action personnalisée sur les objets (synchronisation, archivage...) ou sur les résultats de recherches (export, rapport personnalisé...);

6.1.1. Structure d'écriture

L'écriture d'un LSaddon doit respecter une structure suffisamment souple afin de ne pas être un frein à vos contributions, tout en permettant d'assurer la bonne intégration de votre contribution au projet. Le code que vous écrirez sera réparti dans deux fichiers :

`conf/LSaddons/config.LSaddons.[addon name].php`

Ce fichier contiendra la configuration de votre LSaddon. On y retrouvera la déclaration de constantes et/ou variables de configuration permettant d'adapter votre LSaddon à une installation et à un environnement.

`includes/addons/LSaddons.[addon name].php`

Ce fichier contiendra le code à proprement dit de votre LSaddon.

Structure du fichier `includes/addons/LSaddons.[addon name].php`

```
<?php
```

```
/*  
 * Error messages  
 */
```

```
// Support error messages
```

```

LSError :: defineError('MYADDON_SUPPORT_01',
    ____("MYADDON Support : Unable to load %{dep}."))
);

LSError :: defineError('MYADDON_SUPPORT_02',
    ____("MYADDON Support : The constant %{const} is not defined."))
);

// Other error messages
LSError :: defineError('MYADDON_01',
    ____("An error : %{msg}."))
);

LSError :: defineError('MYADDON_02',
    ____("An other error about %{about} : %{msg}"))
);

LSError :: defineError('MYADDON_03',
    ____("Unknown error."))
);

/**
 * Verify support of my addon by LdapSaisie
 *
 * @author My Name <my.email@example.com>
 *
 * @retval boolean true if my addon is totally supported, false in other cases
 */
function LSaddon_myaddon_support() {

    $retval=true;

    // Check/load dependencies
    if ( !class_exists('mylib') ) {
        if ( !LSession::includeFile(LS_LIB_DIR . 'class.mylib.php') ) {
            LSError :: addErrorCode('MYADDON_SUPPORT_01', 'mylib');
            $retval=false;
        }
    }

    $MUST_DEFINE_CONST= array(
        'LS_MYADDON_CONF_01',
        'LS_MYADDON_CONF_02',
        ...
    );

    foreach($MUST_DEFINE_CONST as $const) {
        if ( (!defined($const)) || (constant($const) == "")) {
            LSError :: addErrorCode('MYADDON_SUPPORT_02', $const);
            $retval=false;
        }
    }
}

```

```

if ($retval) {
    // Register LSaddon view using LSession :: registerLSaddonView()

    if (php_sapi_name() == 'cli') {
        // Register LSaddon CLI command using LScli :: add_command()
    }
}

return $retval;
}

/**
 * My first function
 *
 * Description of this wonderfull function
 *
 * @author My Name <my.email@example.com>
 *
 * @retval [type(s) of returned values (pipe separator)] Description of the return of thi
 **/
function myaddon_first_function($arg1, $arg2) {
    // Do some stuff
    if (something) {
        LSError :: addErrorCode(
            'MYADDON_01',
            'something went wrong' // Error LSformat unique argument
        );
        return false;
    }

    if (something else) {
        LSError :: addErrorCode(
            'MYADDON_02',
            array( // Error LSformat arguments
                'about' => 'second step',
                'msg' => 'something went wrong'
            )
        );
        return false;
    }

    if (still something else) {
        LSError :: addErrorCode('MYADDON_03'); // Error without argument
        return false;
    }
    return true;
}

[...]

// Defined custom CLI commands functions only on CLI context
if (php_sapi_name() != 'cli')

```

```

return true; // Always return true to avoid some warning in log

// Defined functions handling custom CLI commands and optionnaly
// their arguments autocompleter functions.

```

Par convention, la structure de ce fichier est toujours à peu près la même:

- On déclare tout d'abord les messages d'erreurs qui seront potentiellement émis par notre LSaddon en commençant par les messages d'erreurs liés au support de cet LSaddon. On utilise pour cela la méthode `LSError :: defineError()` qui attends en premier argument, l'identifiant du message d'erreur et en tant que second argument, le LSformat du message d'erreur. Par convention, les identifiants des messages d'erreurs seront en majuscule et préfixés du nom du LSaddon.
- On déclare ensuite une fonction `LSaddon_[myaddon]_support` qui sera exécutée lors du chargement de l'addon et qui permettra de s'assurer du support de celui-ci. Cette fonction devra retourner `True` si c'est le cas ou `False` dans le cas contraire.

Cette fonction s'assura notamment :

- que les librairies dont l'addon dépends sont bien chargées et fonctionnelles ;
 - que ses variables et constantes de configuration sont bien définies ;
 - de déclarer les vues personnalisées fournies par cet LSaddon ;
 - de déclarer les commandes *CLI* personnalisées fournies par cet LSaddon ;
- On déclare ensuite les fonctions, classes et éléments fournies et manipulés par l'addon.
 - Si notre addon offre des commandes *CLI* personnalisées, les fonctions les implémentants ne seront définies, dans un souci de performance, que dans un contexte ou elles seraient potentiellement appelables, c'est à dire dans un contexte d'exécution *CLI*. Pour cela, nous utilisons communément la fonction `php_sapi_name` pour déterminer le contexte d'exécution et si celui-ci vaut `cli`, nous stoppons l'exécution du reste du code du fichier via un `return true`.

Note : Il est important dans ce contexte de ne jamais retourner autre chose que `True` pour éviter tout message d'erreur inutile dans les logs.

- On déclare, pour finir, les fonctions implémentant les commandes *CLI* personnalisées et leur éventuelle fonction gérant l'autocomplétion des arguments qu'elles acceptent.

6.1.2. Les vues personnalisées

Les LSaddons peuvent fournir des vues personnalisées qui seront accessibles à tout ou parties des utilisateurs de l'application. Ce filtrage d'accès sera fait en utilisant les LSprofiles de l'utilisateur connecté sur la racine courante de l'annuaire LDAP.

Pour mettre en place une telle vue personnalisée, il est nécessaire de :

- Déclarer cette vue dans la fonction `LSaddon_[addon]_support` de l'addon à l'aide de la méthode `LSsession :: registerLSaddonView`;
- Déclarer la fonction implémentant cette vue. Cette fonction n'acceptera aucun paramètre et ne retournera rien. Elle devra en outre s'occuper de définir son fichier template et charger les dépendances de ce dernier (fichiers *CSS & JS*, variables...).

Pour implémenter une telle vue personnalisée, vous pouvez vous inspirer de l'exemple fourni ci-dessous ou encore des vues fournies par les autres LSaddons (par exemple, l'addon `exportSearchResultAsCSV`).

Structure du fichier includes/addons/LSaddons.[addon name].php

```
<?php
function LSaddon_myaddon_support() {

    $retval=true;

    // Some check

    if ($retval) {
        $retval = LSsession :: registerLSaddonView(
            'myaddon',          // addon name
            'myaddon_view',    // addon view ID
            __('MyAddon view'), // addon view label
            'myaddon_view',    // callable (ex: function name) that implement addon view
            array('user'),     // array listing allowed LSprofiles
            true                // Show/hide this addon view in user menu
        );
    }

    return $retval;
}

[...]
```

```
/**
 * My addon view handler function
 *
 * Description of this view
 *
 * @author My Name <my.email@example.com>
 */
```

```

* @retval void
**/
function myaddon_view() {
    // Do some stuff and set some template variables
    $list = array ([...]);
    LStemplate :: assign('list', $list);

    // Load some CSS & JS files need on this view
    LStemplate :: addCssFile('LSaddon_myadon.css');
    LStemplate :: addJScript('LSaddon_myadon.js');

    // Set template file of the view
    LSession :: setTemplate('LSaddon_myadon_view.tpl');
}

```

6.1.3. Les commandes *CLI* personnalisées

Les LSaddons peuvent fournir des commandes *CLI* personnalisées qui seront accessibles via la commande `ldapsaisie` fournie avec l'application. Cela peut, par exemple, vous permettre de rendre accessible en ligne de commandes une procédure implémentée dans le code d'`LdapSaisie` et vous permettre de mettre en place une tâche planifiée exécutant cette procédure régulièrement.

Pour mettre en place une telle commande *CLI* personnalisée, il est nécessaire de :

- Déclarer cette vue dans la fonction `LSaddon_[addon]_support` de l'addon à l'aide de la méthode `LScLi :: add_command`;
- Déclarer la fonction implémentant cette commande *CLI* personnalisée. Cette fonction acceptera, en tant qu'unique paramètre, un tableau des arguments reçus lors de l'exécution de la commande et retournera `True` ou `False` en cas de succès/d'erreur d'exécution de la commande. Cette valeur de retour influencera le code retourné par la commande : 0 en cas de succès, 1 en cas d'erreur.
- Bien que cela ne soit pas obligatoire, il sera également possible de déclarer une fonction permettant l'autocomplétion des arguments acceptés par la commande.

Cette méthode recevra en paramètre:

`$command_args`

Un tableau des arguments déjà reçus par la commande.

`$comp_word_num`

Un entier indiquant le rang de l'argument que l'autocomplétion tente de compléter. Il peut s'agir du rang d'un paramètre déjà fourni et présent dans le tableau `$command_args` ou bien d'un rang supérieur aux nombres d'arguments déjà fournis à la commande et dans ce cas il s'agira d'autocompléter tous potentiels autre argument que pourrait accepter cette commande.

\$comp_word

Une chaîne de caractères correspondant à ce qu'a déjà saisi l'utilisateur de l'argument que l'on tente d'autocompléter. Cette chaîne de caractères peut être vide ou non, en fonction de s'il s'agit d'un nouvel argument à autocompléter ou non.

\$opts

Un tableau des potentiels arguments globaux acceptés par *LScli* dans le contexte actuel (par exemple, `-d` ou `--debug` pour l'activation du mode debug). La réponse de cette fonction devra inclure ces potentiels arguments si le contexte d'autocomplétion si prête (nouvel argument par exemple).

Pour finir, cette fonction devra retourner un tableau des potentielles valeurs que pourrait prendre l'argument autocomplété. Si une unique proposition est faite à l'utilisateur, celle-ci sera automatiquement proposée à l'utilisateur et à défaut, la liste des valeurs possibles lui seront affichées.

Note : Pour vous aider dans l'écriture d'une telle méthode d'autocomplétion, des méthodes statiques sont fournies par la classe `LScli` pour les autocomplétions les plus courantes:

`LScli :: autocomplete_class_name()`

Autocomplétion du nom d'une classe PHP.

`LScli :: autocomplete_addon_name()`

Autocomplétion du nom d'un LSaddon.

`LScli :: autocomplete_int()`

Autocomplétion d'un nombre entier.

`LScli :: autocomplete_LSubject_types()`

Autocomplétion du nom d'un type d'LSubject.

`LScli :: autocomplete_LSubject_dn()`

Autocomplétion du DN d'un type précis d'LSubject de l'annuaire.

Par ailleurs, la méthode `LScli :: autocomplete_opts()` vous facilitera la construction de la liste des valeurs d'autocomplétion de l'argument courant en fonction de ce qui a déjà été saisi par l'utilisateur (paramètre `$comp_word`). Cette méthode s'occupera en l'occurrence de filtrer parmi toutes les valeurs contextuelles possibles, celles qui correspondent au préfixe fourni par l'utilisateur.

Pour implémenter une telle commande *CLI* personnalisée, vous pouvez vous inspirer de l'exemple fourni ci-dessous ou encore des commandes *CLI* fournies par les autres LSaddons ou classes PHP de l'application.

```

Structure du fichier includes/addons/LSaddons.[addon name].php
<?php
function LSaddon_myaddon_support() {

    $retval=true;

    // Some check

    if ($retval) {
        if (php_sapi_name() == 'cli') {
            LScli :: add_command(
                'my_custom_cli_cmd',           // The CLI command name (required)
                'cli_my_custom_cli_cmd',      // The CLI command handler (must be call
                'My custom CLI command',      // A short description of what this comm
                '[arg1] [arg2] [...]',        // A short list of commands available a
                                           // (optional, default: false)
                'This command permit to ...', // A long description of what this comm
                                           // false)
                true,                         // Permit to define if this command nee
                                           // (optional, default: true)
                'cli_my_custom_cli_cmd_autocomplete', // Callable of the CLI command argument
                                           // default: null)
                true                          // Allow override if a command already
                                           // (optional, default: null)
            );
        }
    }

    return $retval;
}

[...]

// Defined CLI commands functions only on CLI context
if (php_sapi_name() != 'cli')
    return true; // Always return true to avoid some warning in log

/**
 * My addon CLI command my_custom_cli_cmd handler function
 *
 * Description of this CLI command.
 *
 * @param[in] $command_args array Command arguments
 * - Positional arguments :
 *   - LSubject
 *   - dn
 * - Optional arguments :
 *   - -f|--force : Force mode
 *
 */

```

```

* @author My Name <my.email@example.com>
*
* @retval boolean True on succes, false otherwise
**/
function cli_my_custom_cli_cmd($command_args) {
    $objType = null;
    $dn = null;
    $force_mode = false;
    foreach ($command_args as $arg) {
        if ($arg == '-f' || $arg == '--force')
            $force_mode = true;
        elseif (is_null($objType)) {
            $objType = $arg;
        }
        elseif (is_null($dn)) {
            $dn = $arg;
        }
        else
            LScli :: usage("Invalid $arg parameter.");
    }

    if (is_null($objType) || is_null($dn))
        LScli :: usage('You must provide LSobject type and DN.');
```

```

    if (!LSsession :: loadLSobject($objType))
        return false;

    $obj = new $objType();
    if (!$obj->loadData($dn)) {
        self :: log_fatal("Fail to load object $dn data from LDAP");
        return false;
    }

    // Do some stuff on loaded object
    [...]

    return true;
}

/**
 * Args autocompleter for CLI my_custom_cli_cmd command
 *
 * @param[in] $command_args array List of already typed words of the command
 * @param[in] $comp_word_num int The command word number to autocomplete
 * @param[in] $comp_word string The command word to autocomplete
 * @param[in] $opts array List of global available options
 *
 * @retval array List of available options for the word to autocomplete
 **/
public static function cli_my_custom_cli_cmd_autocompleter($command_args, $comp_word_num,
    $opts = array_merge($opts, array ('-f', '--force')));

// Handle positional args

```

```

$objType = null;
$objType_arg_num = null;
$dn = null;
$dn_arg_num = null;
for ($i=0; $i < count($command_args); $i++) {
    if (!in_array($command_args[$i], $opts)) {
        // If object type not defined
        if (is_null($objType)) {
            // Defined it
            $objType = $command_args[$i];
            LScli :: unquote_word($objType);
            $objType_arg_num = $i;

            // Check object type exists
            $objTypes = LScli :: autocomplete_LSubject_types($objType);

            // Load it if exist and not trying to complete it
            if (in_array($objType, $objTypes) && $i != $comp_word_num) {
                LSession :: loadLSubject($objType, false);
            }
        }
        elseif (is_null($dn)) {
            $dn = $command_args[$i];
            LScli :: unquote_word($dn);
            $dn_arg_num = $i;
        }
    }
}

// If objType not already choiced (or currently autocomplete), add LSubject types to av
if (!$objType || $objType_arg_num == $comp_word_num)
    $opts = array_merge($opts, LScli :: autocomplete_LSubject_types($comp_word));

// If dn not already choiced (or currently autocomplete), try autocomplete it
elseif (!$dn || $dn_arg_num == $comp_word_num)
    $opts = array_merge($opts, LScli :: autocomplete_LSubject_dn($objType, $comp_word));

return LScli :: autocomplete_opts($opts, $comp_word);
}

```

6.2. LSformElements

Les LSformElements sont les types de champs de formulaire supportés par l'application.

Pour chaque type implémenté, on devra trouver :

- Une classe PHP dérivée de la classe `LSattr_html` et devant s'appeler `LSattr_html_[nom du type d'attribut HTML]`. Dans celle-ci, il devra être défini à minima la variable de classe `LSformElement_type` permettant de référencer le type d'`LSformElement` à utiliser ;
- Une classe PHP dérivée de la classe `LSformElement` et devant s'appeler `LSformElement_[nom du type d'LSformElement]`. Cette classe implémentera tous ce qui concerne l'affichage du champ dans le formulaire et le traitement d'une valeur retourné par ce dernier. Cela concerne notamment les méthodes suivantes:

`getDisplay()`

Retourne les informations d'affichage du champ dans un formulaire sous la forme d'un tableau (*implémentation obligatoire, pas de méthode par défaut*). Il sera possible de s'appuyer sur la méthode `getLabelInfos()` permettant de générer et récupérer tous ce qui concerne le label du champ du formulaire. Il faudra cependant à minima fournir également la clé `html` dans le tableau retourné qui devra contenir le bout de code HTML correspondant au champ du formulaire. Communément, ce code HTML est généré en appelant la méthode `fetchTemplate()`.

`fetchTemplate()`

Retourne le code HTML du champ dans le formulaire. L'implémentation de cette méthode est facultative et par défaut, cette méthode utilisera la variable de classe `$template` pour connaître le fichier de template à utiliser. Ce fichier de template permettra la génération de la liste de tous les champs associés à chacune des valeurs de l'attribut. Individuellement, le champ d'une des valeurs de l'attribut est généré à l'aide du fichier de template référencé dans la variable de class `$fieldTemplate`.

Note : La variable de classe `$fieldTemplate` est également utilisée par la méthode `LSformElement :: getEmptyField()` qui sert à générer le code HTML d'un champ du formulaire pour une nouvelle valeur de l'attribut. Cette méthode est notamment utilisée lorsque l'on clique sur le bouton permettant d'ajouter une valeur à un champ du formulaire.

`getPostData()`

Récupère dans les données postées par le formulaire, celle concernant ce champ. Cette méthode devra potentiellement traiter l'ensemble des valeurs de l'attribut envoyées par le formulaire et les définir dans le tableau passé en référence en tant que premier argument, les valeurs de l'attribut. L'implémentation de cette méthode est facultative et par défaut, un tableau de valeurs portant le nom de l'attribut LDAP correspondant sera récupérée comme valeur de l'attribut.

Note : Pour plus d'informations sur le rôle et fonctionnement de cette méthode, référer à la méthode par défaut, définie dans la classe PHP parente `LSformElement`.

`setValueFromPostData()`

Définit les valeurs de l'attribut à partir des données reçues du formulaire (et récupérée par la méthode `getPostData()`). L'implémentation de cette méthode est facultative et par défaut, aucune transformation ne sera faite à cette étape sur les données récupérées depuis le formulaire. Implémenter cette méthode pourra cependant se révéler utile en cas de champs de formulaire complexe (attribut composite par exemple).

`autocomplete_attr_values()`

Génère de la liste des valeurs possibles de l'attribut dans un contexte *CLI*.

Note : Pour plus d'informations sur le rôle et fonctionnement de cette méthode, référez-vous aux commentaires de la méthode par défaut, définie dans la classe PHP parente `LSformElement`. Vous pouvez également vous inspirer des exemples d'implémentations fournies avec les autres types d'`LSformElement`.

- Un (ou plusieurs) fichier template pour la génération du code HTML du champ du formulaire. Communément, le fichier `LSformElement.tpl` est utilisé pour générer la structure de la liste des champs correspondant aux différentes valeurs de l'attribut. Ce template utilise une variable `$fieldTemplate` pour définir quel fichier template devra être utilisé pour générer le code HTML de chaque champ associé à une valeur. C'est ce second fichier de template qui est en général à fournir à minima avec votre `LSformElement`.

Note : Il peut être utile d'étendre un type d'`LSformElement` existant pour faciliter l'implémentation d'un nouveau type. Pour cela, vous devez utiliser l'héritage de classe PHP en faisant dériver vos nouvelles classes des classes du `LSformElement` dont vous vous inspirez, plutôt que les classes génériques. Vous pouvez prendre exemple sur le type d'`LSformElement` `pre` qui s'inspire du type `textarea`, ou encore du type `url` dérivé du type `text`.

6.3. LSformRules

Les `LSformRules` sont les règles syntaxiques applicables aux champs des formulaires. Ces règles serviront à s'assurer que les valeurs des champs récupérées des formulaires sont syntaxiquement correctes. Elles seront configurables via le paramètre `check_data` des attributs des `LSobjects`.

Pour chaque type implémenté, on trouvera une classe PHP dérivée de la classe `LSformRule` et devant s'appeler `LSattr_rule_[nom du type]`. Dans celle-ci, il devra être défini la méthode statique `validate()` qui implémentera le contrôle syntaxique. Cette méthode prendra en paramètres :

`$value`

La valeur à tester.

`$options`

Un tableau des options définies dans la configuration pour ce contrôle syntaxique.

`$formElement`

Une référence au champ du formulaire (objet `LSformElement`).

Cette méthode devra retourner `True` ou `False` si la valeur testée est respectivement valide ou invalide.