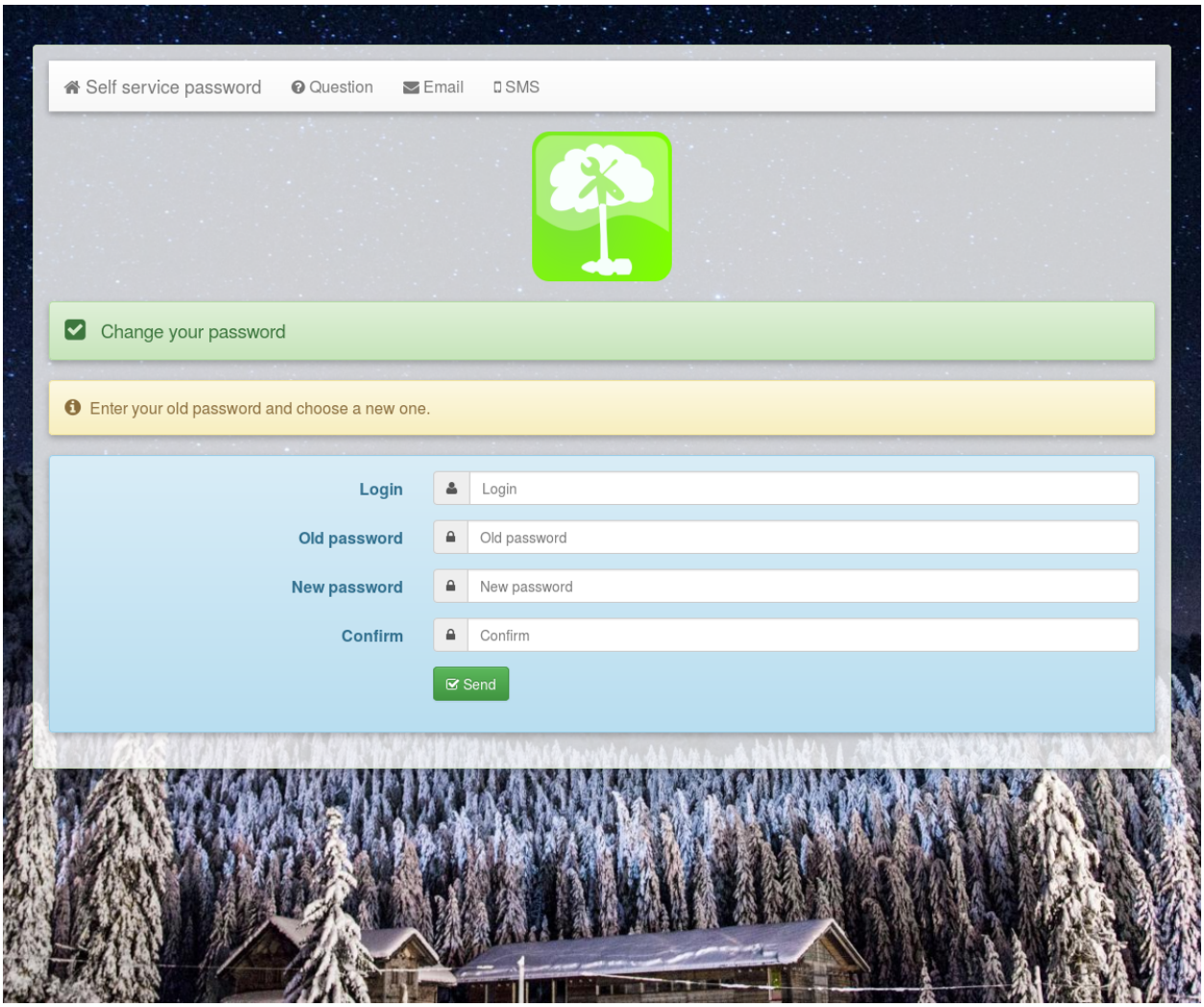
Self Service Password Documentation

Clément OUDOT

Apr 21, 2021

CONTENTS:


Self service password
Question
Email
SMS
Change your password
Enter your old password and choose a new one.
Login
Login
Old password
Old password
New password
New password
Confirm
Confirm
Send

CHAPTER

ONE

PRESENTATION

LDAP Tool Box Self Service Password is a web application for end users. It allows them to change or reset their password if they lost it.

It works with any LDAP directory, including Active Directory.

1.1 Features

- Standard password change
- Reset by questions, token sent by mail, token sent by SMS
- Local password policy
- LDAP advanced usage: password modify extended operation, password policy control
- SSH key change
- Active Directory and Samba modes
- Prehook/Posthook: a script can be launched before and after the password is changed
- Mail notifications

CHAPTER

TWO

INSTALLATION

2.1 From tarball

Uncompress and unarchive the tarball:

```
$ tar -zxvf ltb-project-self-service-password-*.tar.gz
```

Install files in `/usr/share/`:

```
# mv ltb-project-self-service-password-* /usr/share/self-service-password
```

You need to install these prerequisites:

- Apache or another web server
- php (7 or later)
- php-curl (haveibeenpwned api)
- php-filter
- php-gd (captcha)
- php-ldap
- php-mbstring (reset mail)
- php-openssl (token crypt, probably built-in)
- Smarty (version 3)

2.2 Debian / Ubuntu

Configure the repository:

```
# vi /etc/apt/sources.list.d/ltb-project.list
```

```
deb [arch=amd64] https://ltb-project.org/debian/stable stable main
```

Import repository key:

```
# wget -O - https://ltb-project.org/wiki/lib/RPM-GPG-KEY-LTB-project | sudo apt-key add -
```

Then update:

```
# apt update
```

You are now ready to install:

```
# apt install self-service-password
```

2.3 CentOS / RedHat

Warning: You may need to install first the package php-Smarty which is not in official repositories.

Configure the yum repository:

```
# vi /etc/yum.repos.d/ltb-project.repo
```

```
[ltb-project-noarch]
name=LTB project packages (noarch)
baseurl=https://ltb-project.org/rpm/$releasever/noarch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-LTB-project
```

Then update:

```
# yum update
```

Import repository key:

```
# rpm --import https://ltb-project.org/wiki/lib/RPM-GPG-KEY-LTB-project
```

You are now ready to install:

```
# yum install self-service-password
```

Warning: CentOS 7 comes with PHP 5 by default, you need to install PHP 7.

2.4 Docker

Prepare a local configuration file for Self Service Password, for example `/home/test/ssp.conf.php`.

Start container, mounting that configuration file:

```
# docker run -p 80:80      -v /home/test/ssp.conf.php:/var/www/conf/config.inc.local.php      -it doc
```

CHAPTER

THREE

WEBSERVER CONFIGURATION

3.1 Apache configuration

Tip: Debian and RPM packages already include Apache configuration

Here is an example of Apache configuration using a virtual host:

```
<VirtualHost *:80>
    ServerName ssp.example.com

    DocumentRoot /usr/local/self-service-password/htdocs
    DirectoryIndex index.php

    AddDefaultCharset UTF-8

    <Directory /usr/local/self-service-password/htdocs>
        AllowOverride None
        <IfVersion >= 2.3>
            Require all granted
        </IfVersion>
        <IfVersion < 2.3>
            Order Deny,Allow
            Allow from all
        </IfVersion>
    </Directory>

    Alias /rest /usr/local/self-service-password/rest

    <Directory /usr/local/self-service-password/rest>
        AllowOverride None
        <IfVersion >= 2.3>
            Require all denied
        </IfVersion>
        <IfVersion < 2.3>
            Order Deny,Allow
            Deny from all
        </IfVersion>
    </Directory>

    LogLevel warn
    ErrorLog /var/log/apache2/ssp_error.log
```

(continues on next page)

(continued from previous page)

```
    CustomLog /var/log/apache2/ssp_access.log combined
</VirtualHost>
```

You have to change the server name to fit your own domain configuration.

This file should then be included in Apache configuration.

With Debian package, just enable the site like this:

```
# a2ensite self-service-password
```

You can also configure Self Service Password in the default virtual host:

```
Alias /ssp /usr/local/self-service-password/htdocs

<Directory /usr/local/self-service-password/htdocs>
        AllowOverride None
        <IfVersion >= 2.3>
            Require all granted
        </IfVersion>
        <IfVersion < 2.3>
            Order Deny,Allow
            Allow from all
        </IfVersion>
        DirectoryIndex index.php
        AddDefaultCharset UTF-8
</Directory>

Alias /ssp/rest /usr/local/self-service-password/rest

<Directory /usr/local/self-service-password/rest>
        AllowOverride None
        <IfVersion >= 2.3>
            Require all denied
        </IfVersion>
        <IfVersion < 2.3>
            Order Deny,Allow
            Deny from all
        </IfVersion>
        DirectoryIndex index.php
        AddDefaultCharset UTF-8
</Directory>
```

Check you configuration and reload Apache:

```
# apachectl configtest
# apachectl reload
```

3.2 Nginx configuration

Configuration with FastCGI:

```
server {
listen 80;

root /var/www/html;
index index.php index.html index.htm;

# Make site accessible from http://localhost/
server_name _;

# Disable sendfile as per https://docs.vagrantup.com/v2/synced-folders/virtualbox.html
sendfile off;

    gzip on;
    gzip_comp_level 6;
    gzip_min_length 1000;
    gzip_types text/plain text/css application/json application/x-javascript text/xml␣
↪application/xml application/xml+rss text/javascript application/javascript text/x-
↪js;
    gzip_vary on;
    gzip_proxied any;
    gzip_disable "MSIE [1-6]\.(?!.*SV1)";

# Add stdout logging

error_log /dev/stdout warn;
access_log /dev/stdout main;


# pass the PHP scripts to FastCGI server listening on socket
#
location ~ \.php {
    fastcgi_pass unix:/var/run/php-fpm.socket;
    fastcgi_split_path_info       ^(.+\.php)(/.+)$;
    fastcgi_param PATH_INFO       $fastcgi_path_info;
    fastcgi_param PATH_TRANSLATED $document_root$fastcgi_path_info;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_index index.php;
        try_files $fastcgi_script_name =404;
    fastcgi_read_timeout 600;
    include fastcgi_params;
}

    error_page 404 /404.html;
    location = /404.html {
            root /usr/share/nginx/html;
            internal;
}

# deny access to . files, for security
#
location ~ /\. {
        log_not_found off;
        deny all;
```

(continues on next page)

(continued from previous page)

```
}

location ~ /scripts {
        log_not_found off;
        deny all;
}

}
```

Tip: If you get sessions errors with Nginx, try to set `output_buffering` to `4096` (see https://github.com/ltb-project/self-service-password/issues/74)

Example of php.ini:

```
session.save_path = /tmp
upload_max_filesize = 10M
post_max_size = 16M
max_execution_time = 600
request_terminate_timeout = 600
expose_php = Off
output_buffering = 4096
```

CHAPTER

FOUR

GENERAL PARAMETERS

4.1 Configuration files

To configure Self Service Password, you need to create a *local* configuration file named `config.inc.local.php` in `self-service-password/conf`. For example :

```
<?php
// Override config.inc.php parameters below

?>
```

Self Service Password default configuration file is `self-service-password/conf/config.inc.php`. It includes `config.inc.local.php`. Consequently, you can override all parameters in `config.inc.local.php`. This prevents you to be disturbed by an upgrade.

Warning: Do not copy `config.inc.php` into `config.inc.local.php`, as the first one includes the second. You would then create an infinite loop and crash your application.

4.2 Multi tenancy

You can load a specific configuration file by passing a HTTP header. This feature is disable by default. To enable it:

```
$header_name_extra_config = "SSP-Extra-Config";
```

Then if you send the header `SSP-Extra-Config: mydomain`, the file `conf/config.inc.mydomain.php` will be loaded.

4.3 Language

Available languages are:

- Basque (eu)
-  Brazilian (pt-BR)
-  Catalonia (ca)
-  Chinese (cn, zh-CN, zh-TW)

-  Czech (cs)
-  Dutch (nl)
-  English (en)
-  Estonian (ee)
-  French (fr)
-  German (de)
-  Greek (el)
-  Hungarian (hu)
-  Italian (it)
-  Japanese (ja)
-  Norwegian bokmål (nb-NO)
-  Polish (pl)
-  Portuguese (pt-PT)
-  Russian (ru)
-  Serbian (rs)
-  Slovak (sk)
-  Slovenian (sl)
-  Spanish (es)
-  Swedish (sv)
-  Turkish (tr)
-  Ukranian (uk)

Set one of them in `$lang`:

```
$lang = "en";
```

4.4 Menu

To display a top menu, activate the option:

```
$show_menu = true;
```

If menu is not shown, the default application title will be displayed.

4.5 Messages

Help messages provide information to users on how use the interface. They can be disabled with `$show_help`:

```
$show_help = false;
```

You can add extra messages by setting values in these parameters:

```
$messages['passwordchangedextramessage'] = "Congratulations!";
$messages['changehelpextramessage'] = "Contact us if you are lost...";
```

4.6 Graphics

4.6.1 Logo

You change the default logo with your own. Set the path to your logo in `$logo`:

```
$logo = "images/ltb-logo.png";
```

Tip: Comment this parameter to hide logo

4.6.2 Background

You change the background image with your own. Set the path to image in `$background_image`:

```
$background_image = "images/unsplash-space.jpeg";
```

Tip: Comment this parameter to falll back to default background color

4.6.3 Custom CSS

To easily customize CSS, you can use a separate CSS file:

```
$custom_css = "css/custom.css";
```

4.6.4 Footer

You can hide the footer bar:

```
$display_footer = false;
```

4.7 Debug

You can turn on debug mode with `$debug`:

```
$debug = true;
```

4.8 Security

You need a key phrase if you use ciphered tokens (see *Reset by mail tokens*)

```
$keyphrase = "secret";
```

There is also a protection on login to avoid LDAP injections. Some characters are forbidden, you can change the list of forbidden characters in login with `$login_forbidden_chars`:

```
$login_forbidden_chars = "*()&|";
```

Tip: If no characters are configured in `$login_forbidden_chars`, only alphanumeric characters are allowed.

You can configure "obscure" messages, so that some errors are not displayed and replaced by a generic "bad credentials" error:

```
$obscure_failure_messages = array("mailnomatch");
```

You may want to limit number of tries per user/ip in a short time (especially with sms option). If you enable this defaults are 2 tries per login and per hour, and same for ip address:

```
$enable_ratelimit = true;
```

Other possible options for rate limiting:

```
$ratelimit_dbdir = '/tmp';
$max_attempts_per_user = 2;
$max_attempts_per_ip = 2;
$max_attempts_block_seconds = "60";
$client_ip_header = 'REMOTE_ADDR';
```

4.9 Default action

By default, the password change page is displayed. You can configure which page should be displayed when no action is defined:

```
$default_action = "change";
```

Possibles values are:

- `change`
- `sendtoken`
- `sendsms`

You can disable the standard password change if you don't need it:

```
$use_change = false;
```

In this case, be sure to also remove "change" from default action, else the change page will still be displayed.

4.10 Prefill user login

If Self Service Password is called from another application, you can prefill the login but sending an HTTP header.

To enable this feature:

```
$header_name_preset_login = "Auth-User";
```

4.11 Captcha

To require a captcha, set `$use_captcha`:

```
$use_captcha = true;
```

Tip: The captcha is used on every form in Self Service Password (password change, token, questions, etc.)

CHAPTER

FIVE

LDAP CONNECTION

5.1 Server address

Use an LDAP URI to configure the location of your LDAP server in `$ldap_url`:

```
$ldap_url = "ldap://localhost:389";
```

You can set several URI, so that next server will be tried if the previous is down:

```
$ldap_url = "ldap://server1 ldap://server2";
```

To use SSL, set `ldaps` in the URI:

```
$ldap_url = "ldaps://localhost";
```

To use StartTLS, set `true` in `$ldap_starttls`:

```
$ldap_starttls = true;
```

Warning: LDAP certificate management in PHP relies on LDAP system libraries. Under Linux, you can configure `/etc/ldap.conf` (or `/etc/ldap/ldap.conf` on Debian/Ubuntu, or `C:\OpenLDAP\sysconf\ldap.conf` for Windows).

- Provide the certificate from the certificate authority that issued your LDAP server's certificate:

```
TLS_CACERT /etc/ssl/ca.crt
```

- Or, disable server certificate checking:

```
TLS_REQCERT allow
```

If you face issues with non matching TLS versions between SSP and your LDAP server, you can try to set this option:

```
TLS_CIPHER_SUITE TLSv1+RSA
```

5.2 Credentials

Configure DN and password in `$ldap_bindn` and `$ldap_bindpw`, for example a service account:

```
$ldap_binddn = "cn=ssp,ou=dsa,dc=example,dc=com";
$ldap_bindpw = "secret";
```

Tip: You can leave these parameters empty to bind anonymously. In this case, the password modification must be done with user's credentials. But this will not work for password reset.

If you want an SSP account to do this on behalf of the user set the value of `$who_change_password` to `manager`.

To instead use user's credentials when writing in LDAP directory, replace `manager` with `user` in `$who_change_password`:

```
$who_change_password = "user";
```

Warning: The user account can only be used for standard password change, when user is giving its old password. For other password changes (token, questions, ...), manager account will always be used, whatever value is set in `$who_change_password`.

5.3 Search parameters

You can set the base of the search in `$ldap_base`:

```
$ldap_base = "dc=example,dc=com";
```

The filter can be set in `$ldap_filter`:

```
$ldap_filter = "(&(objectClass=person)(uid={login}))";
```

Tip: The string `{login}` is replaced by submitted login.

5.4 Extensions

You can use LDAP password modify extended operation with `$ldap_use_exop_passwd`:

```
$ldap_use_exop_passwd = true;
```

You can also enable LDAP password policy control with `$ldap_use_ppolicy_control`:

```
$ldap_use_ppolicy_control = true;
```

5.5 Special modes

5.5.1 Active Directory

Password in Active Directory is not managed like in other LDAP directories. Use option `$ad_mode` to use `unicodePwd` as password field:

```
$ad_mode = true;
```

You must also use SSL on LDAP connection because AD refuses to change a password on a clear connection. See this documentation to manage Active Directory certificates.

Adapt the search filter too:

```
$ldap_filter = "(&(objectClass=user)(sAMAccountName={login})(!(userAccountControl:1.2.
↪840.113556.1.4.803:=2)))";
```

You can tune some options:

- Force unlock: will unlock a locked account when password is changed

```
$ad_options['force_unlock'] = true;
```

- Force user to change password at next login:

```
$ad_options['force_pwd_change'] = true;
```

- Allow user to change password if password is expired:

```
$ad_options['change_expired_password'] = true;
```

You need to have an account on Active Directory with rights to change password of users. To set the minimum rights for this account, do the following:

- Create a basic domain account without any additional privileges
- Use Delegate control wizard within "User and computers", then
  - User Object
  - Reset Password
  - Write lockoutTime (if unlock is enabled)
  - Write shadowlastchange

If you enabled the reset by questions feature (see *Reset by questions*), you also need to give rights on the question attribute:

- Right click the OU where you want delegation of permissions to propagate down from and select "Delegate Control…"
- Add the account to delegate to, click Next
- Create a custom task to delegate
- Select the radio button for "Only the following objects in the folder", then select "User objects" at the bottom of the list, click Next
- Select the "Property-specific" checkbox only, then locate the attribute you are using to store the "Reset by questions" answer in.

5.5.2 Samba 3 or lower

To manage compatibility with Windows world, Samba stores a specific hash of the password in a second attribute (`sambaNTpassword`). It also store modification date in `sambaPwdLastSet`. Use `$samba_mode` to manage these attributes:

```
$samba_mode = true;
```

You can also update `sambaPwdCanChange` and `sambaPwdMustChange` attributes by settings minimal and maximal age, in days:

```
$samba_options['min_age'] = 5;
$samba_options['max_age'] = 45;
```

To set an expiration date for a Samba account (attribute `sambaKickofftime`), configure a maximal age, in days:

```
$samba_options['expire_days'] = 90;
```

Tip: Samba modifications will only be done on entries of class `sambaSamAccount`

Tip: For Samba 4, you must use AD mode, not Samba mode.

5.5.3 Shadow

If using `shadowAccount` object class for users, you can update the `shadowLastChange` attribute when changing password:

```
$shadow_options['update_shadowLastChange'] = true;
```

You can also update the `shadowExpire` attribute to define when the password will expire. Use `-1` to never expire, else configure the number of days:

```
$shadow_options['update_shadowExpire'] = true;
$shadow_options['shadow_expire_days'] = 365;
```

Tip: Shadow modifications will only be done on entries of class `shadowAccount`

CHAPTER

SIX

PASSWORD POLICY

6.1 Hashing

You can use these schemes to hash the password before sending it to LDAP directory:

- SHA
- SSHA
- MD5
- SMD5
- CRYPT
- clear
- auto

Set one of them in `$hash`:

```
$hash = "clear";
```

Warning: This option is ignored with Active Directory mode.

Tip: Use `auto` to get the current password value and find the hash. This requires a read access to the password.

You can configure the crypt salt prefix to choose the algorithm (see crypt documentation):

```
$hash_options['crypt_salt_prefix'] = "$6$";
```

6.2 Size

Set minimal and maximal length in `$pwd_min_length` and `$pwd_max_length`:

```
$pwd_min_length = 4;
$pwd_max_length = 8;
```

Tip: Set `0` in `$pwd_max_length` to disable maximal length checking.

6.3 Characters

You can set the minimal number of lower, upper, digit and special characters:

```
$pwd_min_lower = 3;
$pwd_min_upper = 1;
$pwd_min_digit = 1;
$pwd_min_special = 1;
```

Special characters are defined with a regular expression, by default:

```
$pwd_special_chars = "^a-zA-Z0-9";
```

This means special characters are all characters except alphabetical letters and digits.

You can check that these special characters are not at beginning or end of the password:

```
$pwd_no_special_at_ends = true;
```

You can also disallow characters from being in password, with `$pwd_forbidden_chars`:

```
$pwd_forbidden_chars = "@%";
```

This means that `@` and `%` could not be present in a password.

You can define how many different class of characters (lower, upper, digit, special) are needed in the password:

```
$pwd_complexity = 2;
```

6.4 Pwned Passwords

Allows to check if the password was already compromised, using https://haveibeenpwned.com/ database:

```
$use_pwnedpasswords = true;
```

6.5 Reuse

You can prevent a user from using his old password as a new password if this check is not done by the directory:

```
$pwd_no_reuse = true;
```

You may also want to check for partial password reuses, ensuring the new password includes at least N distinct new characters:

```
$pwd_diff_last_min_chars = 3;
```

6.6 Forbidden words

Give a list of forbidden words that the password should not contain:

```
$pwd_forbidden_words = array("azerty", "qwerty", "password");
```

6.7 Forbidden LDAP fields

Give a list of LDAP fields which values should not be present in the password:

```
$pwd_forbidden_ldap_fields = array('cn', 'givenName', 'sn', 'mail');
```

6.8 Show policy

Password policy can be displayed to user by configuring `$pwd_show_policy`. Three values are accepted:

- `always`: policy is always displayed
- `never`: policy is never displayed
- `onerror`: policy is only displayed if password is rejected because of it, and the user provided his old password correctly.

```
$pwd_show_policy = "never";
```

You can also configure if the policy will be displayed above or below the form:

```
$pwd_show_policy_pos = "above";
```

6.9 Extended error

You can display the error message returned by the directory when password is refused. The message content depends on your LDAP server software:

```
$show_extended_error = true;
```

CHAPTER

SEVEN

RESET BY QUESTIONS

7.1 How it works?

First, the user should choose a question and register an answer. This answer will be stored in an attribute of its LDAP entry with this syntax:

```
{questionid}answer
```

Warning: You should configure your LDAP directory to protect this data, to be only accessed by Self Service Password. See also in this page how to encrypt values into LDAP directory.

Warning: The data will be written by the user or by the manager, depending on `$who_change_password` parameter.

Then, the user can reset its password by entering its answer and setting a new password.

7.2 Activation

You can enable or disable this feature with `$use_questions`:

```
$use_questions = true;
```

7.3 Multiple answers

By default, a user can only register an answer to one question. You can allow users to register an answer to more than one question with this parameter:

```
$multiple_answers = true;
```

Then the user can use any valid answer to reset its password.

You can also configure how many questions are displayed in the form. If you want to require 2 answers to 2 different questions, configure `$questions_count`:

```
$questions_count = 2;
```

7.4 Populate questions

This feature allows users to first submit an empty form with just their login. The form will be displayed again with questions already registered for this user. As this lowers the security, this is disabled by defaut. Configure `$question_populate_enable` to enable it:

```
$question_populate_enable = true;
```

7.5 Attribute and object class

Set the attribute in which the answer will be stored:

```
$answer_attribute = "info";
```

Warning: The attribute name must be in lower case, this is required by php-ldap API.

If the above attribute is not in a standard user object class, configure the object class to use with this attribute:

```
$answer_objectClass = "extensibleObject";
```

Tip: The object class will be added to the entry only if it is not already present.

If you enabled multiple answers, you can choose if they will be stored as multiple values of the attribute, or stored in a single value:

```
$multiple_answers_one_str = true;
```

On Active Directory, extensibleObject is not known. You can use for example:

```
$answer_attribute = "comment";
$answer_objectClass = "user";
```

7.6 Crypt answers

Before 1.3 release, answers could not be encrypted in LDAP directory. An option can now be used to encrypt answers:

```
$crypt_answers = true;
```

You can set this option to `false` to keep the old behavior.

Warning: If you enable this option, you must change the default value of the security keyphrase

A script is provided to encrypt all clear text answers in LDAP directory, to allow a swooth migration. Just run the script (it will use your SSP LDAP settings to update values):

```
# php /usr/share/self-service-password/scripts/encrypt_answers.php
```

7.7 Edit questions

Default questions are registered in lang files: `lang/**codelang**.inc.php`.

To add a question, you can create a new value in the `$messages['questions']` array, directly in local configuration file (`config.inc.local.php`):

```
$messages['questions']['ice'] = "What is your favorite ice cream flavor?";
```

Or better, to be able to translate it, create it in every customized lang file under `conf/`.

To disable the default questions form the main configuration file, set:

```
$questions_use_default = true;
```

CHAPTER

EIGHT

RESET BY MAIL TOKENS

8.1 How it works?

First, the user will enter his login and his mail address. A mail is sent to him.

Then, the user click on the link in the mail, an can set a new password.

Tip: PHP sessions are used to store and retrieve token on server side.

8.2 Activation

You can enable or disable this feature with `$use_tokens`:

```
$use_tokens = true;
```

8.3 Mail configuration

See *Mail*.

You can also avoid to request the mail to the user, only the login will be asekd, and the mail will be read in LDAP:

```
$mail_address_use_ldap = true;
```

8.4 Security

You can crypt tokens, to protect the session identifier:

```
$crypt_tokens = true;
```

Warning: If you enable this option, you must change the default value of the security keyphrase.

You should set a token lifetime, so they are deleted if unused. The value is in seconds:

```
$token_lifetime = "3600";
```

Warning: Token deletion is managed by PHP session garbage collector.

8.5 Log

By default, generated URLs are logged in the default Apache error log. This behavior can be changed, to log in a specific file:

```
$reset_request_log = "/var/log/self-service-password";
```

Warning: Apache user must have write permission on this file.

8.6 Reset URL

By default, reset URL is computed using server name and port, but these values can be wrong if the application is behind a reverse proxy. In this case you can set yourself the reset URL:

```
$reset_url = $_SERVER['HTTP_X_FORWARDED_PROTO'] . "://" . $_SERVER['HTTP_X_FORWARDED_
↪HOST'] . $_SERVER['SCRIPT_NAME'];
```

CHAPTER

NINE

RESET BY SMS

9.1 How it works?

First, the user will enter his login. With this login, SSP will try to get information, like name and mobile phone number.

If information is found, the user can check it and confirm the sent of reset code trough SMS.

A message is sent either to an Email to SMS gateway, either trough an API (called with PHP code or by script).

9.2 SMS provider

You first have to choose SMS provider. Search the web to find one, many have a free trial so you can test the feature.

Some known providers:

- Email 2 SMS:
  - SMS Global
- API:
  - SMS Global
  - Smart Focus
  - OVH SMS API

9.3 Activation

You can enable or disable this feature with $use_sms:

```
$use_sms = true;
```

Warning: If you enable this option, you must change the default value of the security keyphrase.

9.4 Method

Choose which method to use, `mail` or `api`:

```
$sms_method = "mail";
```

9.4.1 Mail

If you choose the mail method, the mail will be sent to SMS provider trough mail configuration (see *Mail*).

You can adjust some settings here, depending on provider guidelines:

```
# Send SMS mail to address
$smsmailto = "{sms_attribute}@service.provider.com";
# Subject when sending email to SMTP to SMS provider
$smsmail_subject = "Provider code";
```

9.4.2 API

If you choose API, you need to define which library will be called:

```
$sms_api_lib = "lib/smsapi.inc.php";
```

In this library, you must define the `send_sms_by_api` function:

```
function send_sms_by_api($mobile, $message) {

    # PHP code
    # ...

    # Or call to external script
    # $command = escapeshellcmd(/path/to/script).' '.escapeshellarg($mobile).' '.
↪escapeshellarg($message);
    # exec($command);

    return 1;
}
```

Read the provider guidelines to know how to access its API.

Tip: An example is given in lib/smsapi-example.inc.php. Copy this file to lib/smsapi.inc.php and start coding!

See also sms_api.

9.5 Mobile attribute

Set here which LDAP attribute hold the user mobile phone:

```
$sms_attribute = "mobile";
```

You can also partially hide the value when it is displayed on the confirmation page:

```
$sms_partially_hide_number = true;
```

To remove any non digit character from SMS number;

```
$sms_sanitize_number = true;
```

To truncate SMS number:

```
$sms_truncate_number = true;
$sms_truncate_number_length = 10;
```

9.6 Message

Set the message here, it uses by default the `smsresetmessage` message defined in lang files and the `smstoken` parameter:

```
# Message
$sms_message = "{smsresetmessage} {smstoken}";
```

9.7 Token

You can set the token length:

```
$sms_token_length = 6;
```

You can also configure the allowed attempts:

```
$max_attempts = 3;
```

After these attempts, the sent token is no more valid.

CHAPTER

TEN

MAIL

10.1 LDAP Attribute

Set the LDAP attribute where user email is stored:

```
$mail_attribute = "mail";
```

Tip: Only the first value of this attribute will be used to get the mail address.

10.2 Sender name

You can change the default `From` header and add a signature:

```
$mail_from = "admin@example.com";
$mail_from_name = "Self Service Password administrator";
$mail_signature = "";
```

10.3 Change password notification

Use this option to send a confirmation mail to the user, just after a successful mail change:

```
$notify_on_change = true;
```

10.4 PHPMailer

You can set all parameters for PHPMailer:

```
$mail_sendmailpath = '/usr/sbin/sendmail';
$mail_protocol = 'smtp';
$mail_smtp_debug = 0;
$mail_debug_format = 'html';
$mail_smtp_host = 'localhost';
$mail_smtp_auth = false;
$mail_smtp_user = '';
```

(continues on next page)

(continued from previous page)

```
$mail_smtp_pass = '';
$mail_smtp_port = 25;
$mail_smtp_timeout = 30;
$mail_smtp_keepalive = false;
$mail_smtp_secure = 'tls';
$mail_smtp_autotls = true;
$mail_smtp_options = array();
$mail_contenttype = 'text/plain';
$mail_wordwrap = 0;
$mail_charset = 'utf-8';
$mail_priority = 3;
```

Tip: See https://github.com/PHPMailer/PHPMailer for more information

CHAPTER

ELEVEN

PRE & POST HOOK CONFIGURATION

11.1 How it works?

You can write a script that will be called before changing a password (pre hook) or after a successful password change (post hook).

This allow for example to update a file or a database on password change.

This script must be executable by the user running Apache. It will take 3 arguments:

- `$login` : the user login
- `$newpassword` : the new password
- `$oldpassword` : the old password

Tip: The old password is only provided on standard password change, not on password reset

To declare this script, use:

```
$prehook = "/usr/share/self-service-password/prehook.sh";
$posthook = "/usr/share/self-service-password/posthook.sh";
```

You can choose to display an error if the script return code is greater than 0:

```
$display_prehook_error = true;
$display_posthook_error = true;
```

The displayed message will be the first line of the script output.

Another option can be enabled to encode the password in base64 before sending it to the script, which can avoid an execution issue if the password contains special characters:

```
$prehook_password_encodebase64 = false;
$posthook_password_encodebase64 = false;
```

By default With prehook script, the password will not be changed in LDAP directory if the script fails. You can change this behavior to ignore script error. This could be useful to run prehook script and display a warning if it fails, but still try to update password in the directory.

```
$ignore_prehook_error = true;
```

Here is an example of a simple hook script:

```
#!/bin/bash

LOGIN=$1
NEWPASSWORD=$2
OLDPASSWORD=$3

echo `date` >> /tmp/posthook.log
echo "$LOGIN / $NEWPASSWORD / $OLDPASSWORD" >> /tmp/posthook.log

... there is an error ...
echo "Posthook script has failed"
exit 1
... there is no error ...
exit 0
```

Warning: This script is an example, do use not it in production: passwords should never be put in logs. Write your own script to propagate the password in a safe place

Warning: If you are using systemd, it is possible that the PrivateTmp feature is enabled by default for Apache (in your httpd.service or apache2.service).

When enabled, all logs written from posthook.sh to /tmp will be redirected to /tmp/systemd-private-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-apache2.service-XXXXXX/tmp or similar.

11.2 Example : Multi LDAP posthook

You can configure multiple LDAP backend.

To enable this option, you have to add the posthook :

```
$posthook = "php /usr/share/self-service-password/multi_ldap_change.php $login
↪$newpassword";
```

You need to add the list of your ldap backend :

```
$secondaries_ldap[0]['ldap_url'] = 'ldap://ldap2.example.com';
$secondaries_ldap[1]['ldap_url'] = 'ldap://ldap3.example.com';
```

It's necessary to activate the base64 enconding for special characters. You can be enabled this option with this configuration line :

```
$posthook_password_encodebase64 = true;
```

Warning: This script suppose that you use the same credentials on all your backend.

CHAPTER

TWELVE

WEBSERVICES (REST API)

12.1 Configuration

REST API access is forbidden by default in web server configuration.

You must allow and protect access (for example with htaccess).

You must also enable rest_api in configuration:

```
$use_restapi = true;
```

12.2 API

Here are available services:

POST /rest/v1/checkpassword.php
Check if a password respect the password policy

Status Codes

- 200 OK – Successful response

POST /rest/v1/changepassword.php
Allow user to change his own password

Status Codes

- 200 OK – Successful response

POST /rest/v1/adminchangepassword.php
Allow admin to change the password for a user

Status Codes

- 200 OK – Successful response

HTTP ROUTING TABLE

/rest