Validation Rules

To submit any form you need some rules to give inputs.

This is the list of rules you can use for fields.

1. required
    - **Returns FALSE if the form field is empty.**
2. valid_email
    - **Returns FALSE if the form field does not contain a valid email address.**
3. valid_emails
    - **Returns FALSE if any value provided in a comma separated list is not a valid email.**
4. min_length
    - **Returns FALSE if the form field is shorter than the parameter value.**
    - **Example: min_length[5]. Where 5 is a parameter of length.**
5. max_length
    - **Returns FALSE if the form field is longer than the parameter value.**
    - **Example: max_length[10]. Where 10 is a parameter of length.**
6. exact_length
    - **Returns FALSE if the form field is not exactly the parameter value.**
    - **Example: exact_length[10]. Where 10 is a parameter of length.**
7. matches
    - **Returns FALSE if the form field does not match the defined value of parameter.**
    - **Example: matches[field]. Where field is parameter name of field to match with.**
8. Alpha

- **Returns FALSE if the form field contains anything other than alphabetical characters.**

9. alpha_numeric
    - **Returns FALSE if the form field contains anything other than alpha-numeric characters.**
10. alpha_dash
    - **Returns FALSE if the field contains anything other than alpha-numeric characters, underscores or dashes.**
11. numeric
    - **Returns FALSE if the form field contains anything other than numeric characters.**
12. integer
    - **Returns FALSE if the form field contains anything other than an integer.**
13. decimal
    - **Returns FALSE if the form field contains anything other than a decimal number.**
14. is_natural
    - **Returns FALSE if the form field contains anything other than a natural number.**
15. is_natural_no_zero
    - **Returns FALSE if the form field contains anything other than a natural number, but not zero.**
16. valid_ip
    - **Returns FALSE if the supplied IP is not valid.**
17. sm_captcha_validate
    - **Returns FALSE if the EE Captcha is not valid. (Default for EE captcha field.)**
18. is_unique
    - **Returns FALSE id the field value match with the field in database table.**
    - **Example: is_unique[members.email]. Where "members" is table name and "email" is field of that table.**

19. exists_email
    - **Returns FALSE if email field entered by user is not exists in members table.**
    - **Used in forgot password form email field by default.**
20. auth_password
    - **Returns FALSE if user entered wrong password.**
    - **Use in edit profile form to validate current password.**

Hooks

Smart Members: Hooks

Hooks are use to modify the data or writing your own flow on addon without changing the core code of addon. List of hooks given in smart members are listed below:

1. Build form starts:
    1. sm_build_form_start
    2. sm_before_logout_link
2. Build forms ends:
    1. sm_build_form_end
3. Start submit forms:
    1. sm_submit_form_start
    2. sm_logout_start
4. End submit forms:
    1. sm_submit_form_end
5. If errors found in forms:
    1. sm_error_in_form
    2. sm_outer_error
6. Initialize validation rules:

    1. sm_init_validation

7. View profile start
    1. sm_view_profile_start
    2. sm_view_profile_end
8. Before send email:
    1. sm_before_send_email
9. Total fields listed (Constructor Hook)
    1. sm_total_fields
10. Social Media login (Pro Feature)
    1. sm_before_social_login
    2. sm_after_social_login
11. Import member Hooks (Pro Feature [EE4 only])
    1. sm_element_before_import
    2. sm_element_after_import
12. You can also use ExpressionEngine default hooks given below:
    1. member_member_register_start
    2. member_member_register_errors
    3. member_member_register
    4. member_member_logout
    5. member_delete
    6. member_update_start
    7. member_update_end
    8. cp_members_validate_members
    9. member_register_validate_members

sm_build_form_start

This hook will call every time a forms build start. This hook let user to modify data before form generates.

Hook Name	sm_build_form_start
Parameter 1	Source: Actual source of form.[i.e, “registration”, “edit_profile”, “forgot_password”, “reset_password”]
Parameter 2	Total Fields: Array of total possible fields.
Return	Total Fields.

sm_before_logout_link

This hook will call when generates logout url from {exp:smart_members:logout}. It has Logout URL in parameter of hook function.

Hook Name	sm_before_logout_link
Parameter	Logout URL
Return	Logout URL

sm_build_form_end

This hook will call every time a forms ends build. This hook will call before replace the actual data with tagdata. The parameter contains Source and the Variable array of form.

Hook Name	sm_build_form_end
Parameter 1	Source: Actual source of form.[i.e, “registration”, “edit_profile”, “forgot_password”, “reset_password”]
Parameter 2	Variable array of form parameters and hidden fields.
Return	Variable array of form parameters and hidden fields.

sm_submit_form_start

This hook will call when user submits a form. This hook will call before validation hook.

Hook Name	sm_submit_form_start
Parameter 1	Source: Actual source of form.[i.e, “registration”, “edit_profile”, “forgot_password”, “reset_password”]

Parameter 2	Parameter array passed in exp loop of form
Return	Parameter array passed in exp loop of form

sm_logout_start

This hook will call when a user click on logout URL. The data passed in parameter is the GET array which can secure action parameter or return url.

Hook Name	sm_logout_start
Parameter	GET array
Return	GET array

sm_submit_form_end

This hook will call when we submits a form without any error and it update the form data into database.

Hook Name	sm_submit_form_end
Parameter	Source: Actual source of form.[i.e, “registration”, “edit_profile”, “forgot_password”, “reset_password”]

1	
Parameter 2	Return URL
Return	Return URL

sm_error_in_form

This hook will call when we found errors in our forms submitted data.

Hook Name	sm_error_in_form
Parameter 1	Source: Actual source of form.[i.e, “registration”, “edit_profile”, “forgot_password”, “reset_password”]
Parameter 2	Array of errors
Return	Array of errors

sm_outer_error

It is possible that error occurs in the form of the field of some xyz field and you forgot to add {error:xyz} in form. In that case error will show in ee default gray screen.

(Only called if error_reporting="inline" is passed as parameter in form)

Hook Name	sm_outer_error
Parameter 1	Source: Actual source of form.[i.e, "registration", "edit_profile", "forgot_password", "reset_password"]
Parameter 2	Array of errors
Return	Array of errors

sm_init_validation

This hook will call before initialization of validation array to field. So you can control rules of fields before assign rules to fields.

Hook Name	sm_init_validation
Parameter	Source: Actual source of form.[i.e, "registration", "edit_profile", "forgot_password", "reset_password"]

1	
Parameter 2	Array of validation array (Form settings)
Return	Array of validation array (Form settings)

sm_view_profile_start

This hook will call when you trigger {exp:smart_members:profile} tag.

Hook Name	sm_view_profile_start
Parameter	NA
Return	NA

sm_view_profile_end

This hook will call before the actual profile data will replace with tagdata.

It will pass member data array in parameter and accept the same as return.

Hook Name	sm_view_profile_end
Parameter	Array of member data
Return	Array of member data

sm_before_send_email

This hook will call before any email sends with smart members. i.e., Registration email, forgot password email.

Hook Name	sm_before_send_email
Parameter 1	Source: Actual source of form.[i.e, “registration”, “edit_profile”, “forgot_password”, “reset_password”]
Parameter 2	Array of validation array (Form settings)
Return	Array of validation array (Form settings)

sm_total_fields

This is constructor hook. All possible fields that can exists in form or profile can contains. Array of this field is parameter of this hook.

Hook Name	sm_total_fields
Parameter	Array of total possible fields a form can contains.
Return	Array of total possible fields a form can contains.

sm_before_social_login (Pro Feature)

This hook will call before social login API call.

Hook Name	sm_before_social_login
Parameter	URL to be called for authenticate user with social site.
Return	URL to be called for authenticate user with social site.

sm_after_social_login (Pro Feature)

This is constructor hook. All possible fields that can exists in form or profile can contains. Array of this field is parameter of this hook.

Hook Name	sm_after_social_login
Parameter	Return URL to be called after successful authentication.
Return	Return URL to be called after successful authentication.

sm_element_before_import (Pro Feature [EE4 >= v2.0.3])

This hook will call just before member will insert or update in database. Hook has one parameter that contains array that going to be insert/update so one can modify member data before insert it into the database.

Hook Name	sm_element_before_import
Parameter	Full Array that going to be save in database.
Return	Same array after modification.

sm_element_after_import (Pro Feature [EE4 >= v2.0.3])

This hook will call just after member will save in database. Hook has one parameter that contains array that just saved into the database and second parameter as member_id which holds the data.

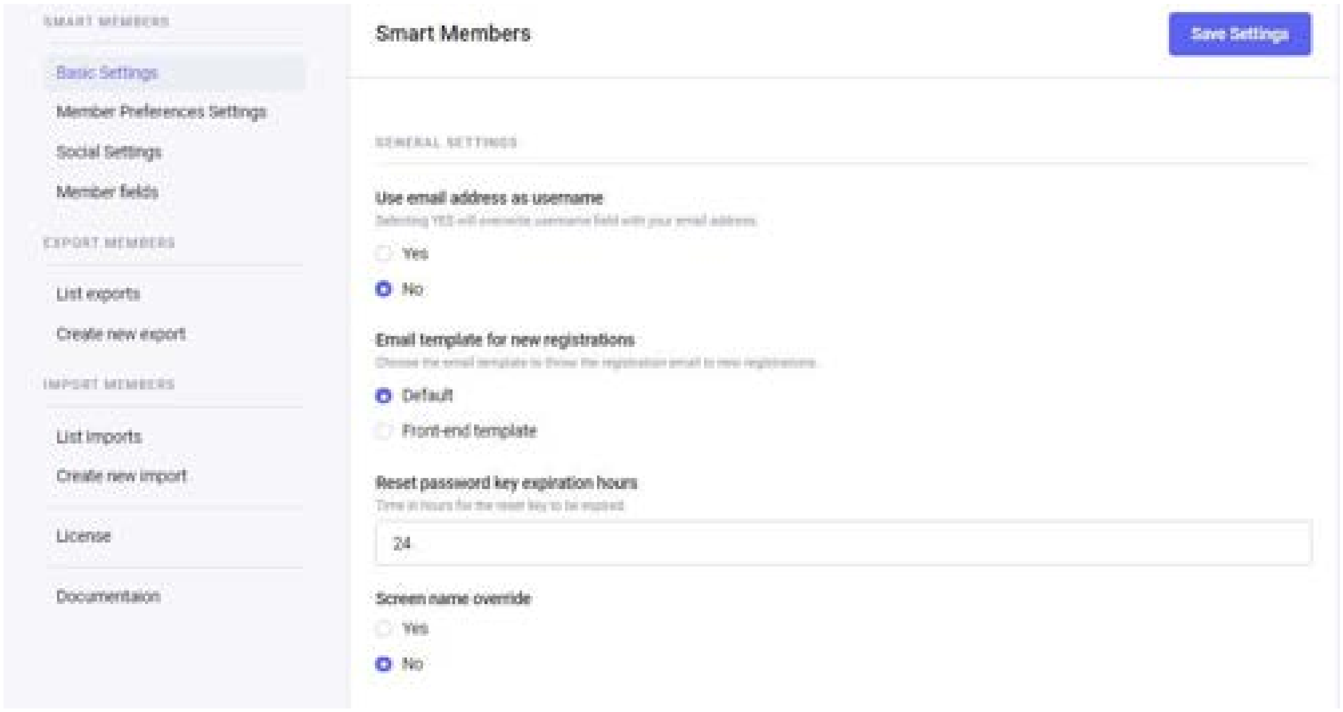
Hook Name	sm_element_after_import
Parameter 1	Full Array that saved in database.
Parameter 2	Member ID holding particular member.
Return	NA

Basic setting

Basic settings allows you to customize addon settings as per your requirements. Form is easy to understand and settings will effect in every module we use in frontend.

Setting form allows you to mark username as password, override screen name field, set registration email template and forgot password email template, set reset password template etc.

Basic setting form (Default view)



Use email address as username
Selecting YES will overwrite username field with your email address.
Yes [✓ No]

Email template for new registrations
Choose the email template to throw the registration email to new registrations.
[✓ Default] Front-end template

Reset password key expiration hours
Time in hours for the reset key to be expired.
24

Screen name override
Yes [✓ No]

There are multiple fields that can change as per your requirements. Details about those fields are given below:

1. Use email address as username:
   This field can be use to have same value for both username and email field. Checking this to YES will no longer need email field in registration

page. You only need to use username field and put the type as email so one can not enter the text input rather then email address.Example:

```
<input type="email" name="username" placeholder="Email Address">
{error:username} //if error_reporting="inline"
```

Note “type” of input is “email” and “name” of input is “username”. Same will follow in edit profile module.

2. Email template for new registrations:
   You have 2 options here. You can either put every code of email templates for registrations and forgot password in backend setting form or you can generate template to frontend.
   Select “Default” if you want to use backend setting form email templates for send emails.
   Select front end template if you want to generate your own templates to send emails to user. Select the templates from front end is necessary for this option.

3. Screen name override:
   This option allows you to use any custom field as Screen name. That will took the field on registration process and override screen name.As screen name field must be unique, If one take a field name and that name is already in screen name field of any other user, the default screen name will use instead of override.

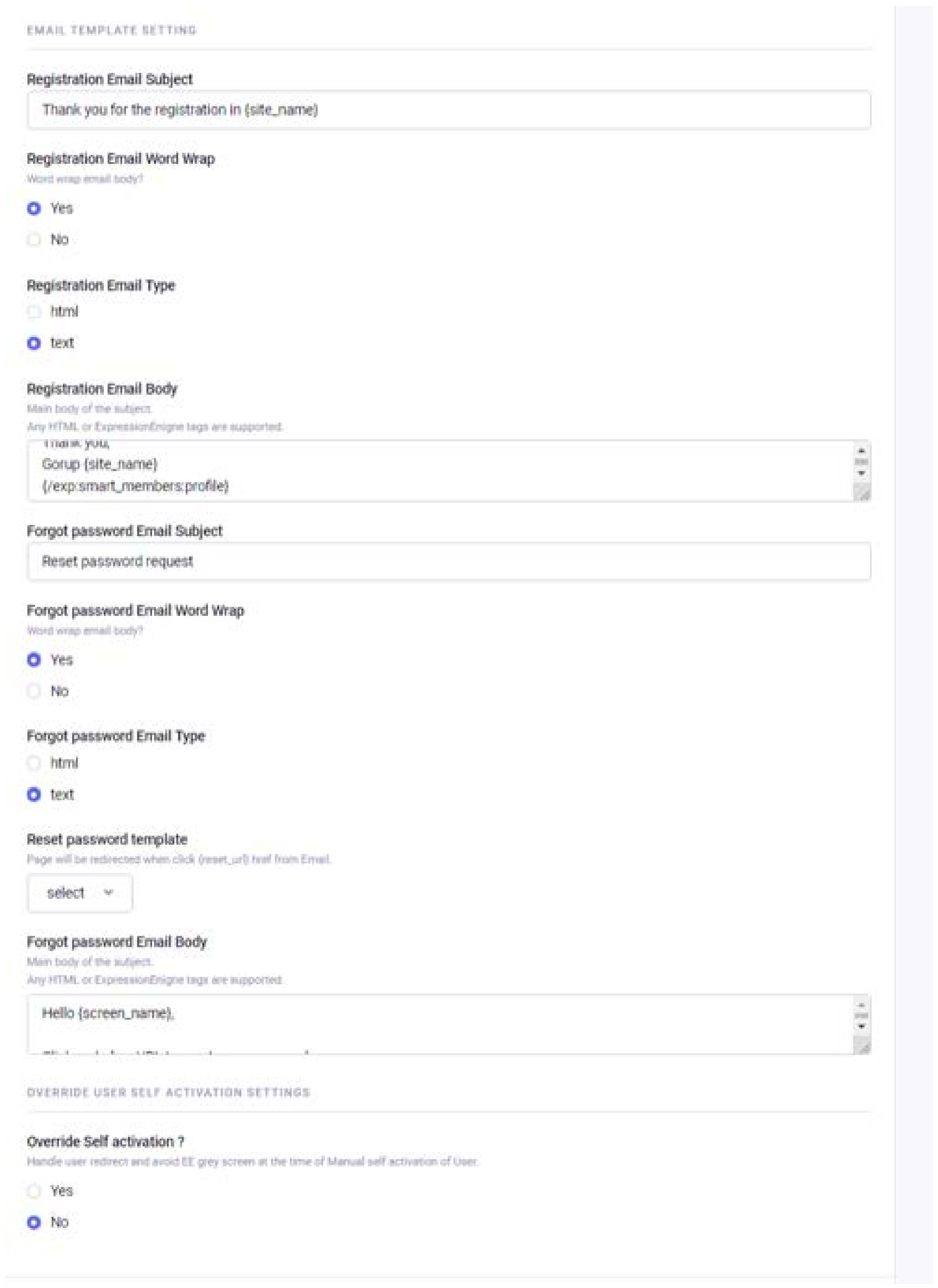
4. Enable Re-captcha ?Enabling this will ask you recaptcha key and secret. You can generate recaptcha key and secret from google recaptcha site.
   URL: https://www.google.com/recaptcha/admin#list
   Enter your domain and generate key and secret. You can then use google recaptcha in your front end module by entering enable_recaptcha=”yes” in parameter and code of google recaptcha in template:

```
{if captcha}
<p>
       <label for="captcha">Please enter in the word you see:</label>
       {captcha}
       <input type="text" name="captcha" id="captcha" />
       {error:captcha}
</p>
{if:elseif recaptcha}
<p style="margin: 0;"><label for="recaptcha">Click the checkbox</labe
       {recaptcha}
       {error:recaptcha}
</p>
{/if}
```

Captcha and recaptcha to use in if else condition is helpful in case if you pass enable_recaptcha=”yes” in parameter and not give API key and Secret in backend setting form. In such case it will display normal captcha in place of recaptcha.

1. Email template setting:
   If you choose Email template for new registrations to default, You need to pass the email settings in form at down otherwise you only need to select the email templates.

Basic Settings (Email template view)



EMAIL TEMPLATE SETTING

Registration Email Subject

Thank you for the registration in {site_name}

Registration Email Word Wrap

Word wrap email body?

[✓ Yes] No

Registration Email Type

html [✓ text]

Registration Email Body

Main body of the subject.
Any HTML or ExpressionEnigne tags are supported.

```
Thank you,
Gorup {site_name}
{/exp:smart_members:profile}
```

Forgot password Email Subject

Reset password request

Forgot password Email Word Wrap

Word wrap email body?

[✓ Yes] No

Forgot password Email Type

html [✓ text]

Reset password template

Page will be redirected when click {reset_url} href from Email.

select

Forgot password Email Body

Main body of the subject.
Any HTML or ExpressionEnigne tags are supported.

```
Hello {screen_name},
```

OVERRIDE USER SELF ACTIVATION SETTINGS

Override Self activation ?

Handle user redirect and avoid EE grey screen at the time of Manual self activation of User.

Yes [✓ No]

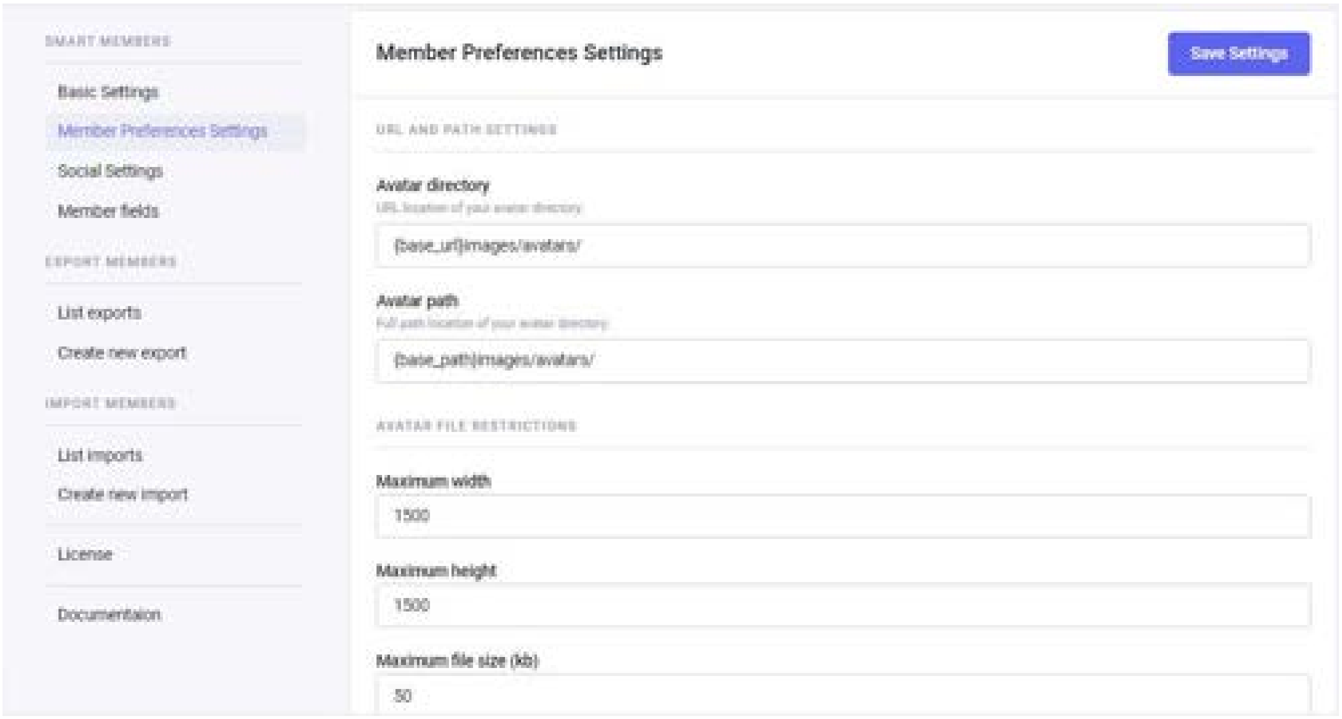
Member preferences

Member preferences setting form is only developed for EE3 users as static images such as Photo file and Signature Image fields settings of path, height, width etc. Settings are available in member preferences in EE2, But such setting forms are not available for EE3 right now.

“Allow avatars?” and “Allow avatars?” setting is removed from EE6 because EE6 makes it compulsory to enable avatars for members.

You can simply adjust the settings for photo file, avatar and signature image settings from this module.

Member Preferences setting form



Member Preferences Settings

URL AND PATH SETTINGS

Avatar directory
URL location of your avatar directory
{base_url}images/avatars/

Avatar path
Full path location of your avatar directory
{base_path}images/avatars/

AVATAR FILE RESTRICTIONS

Maximum width
1500

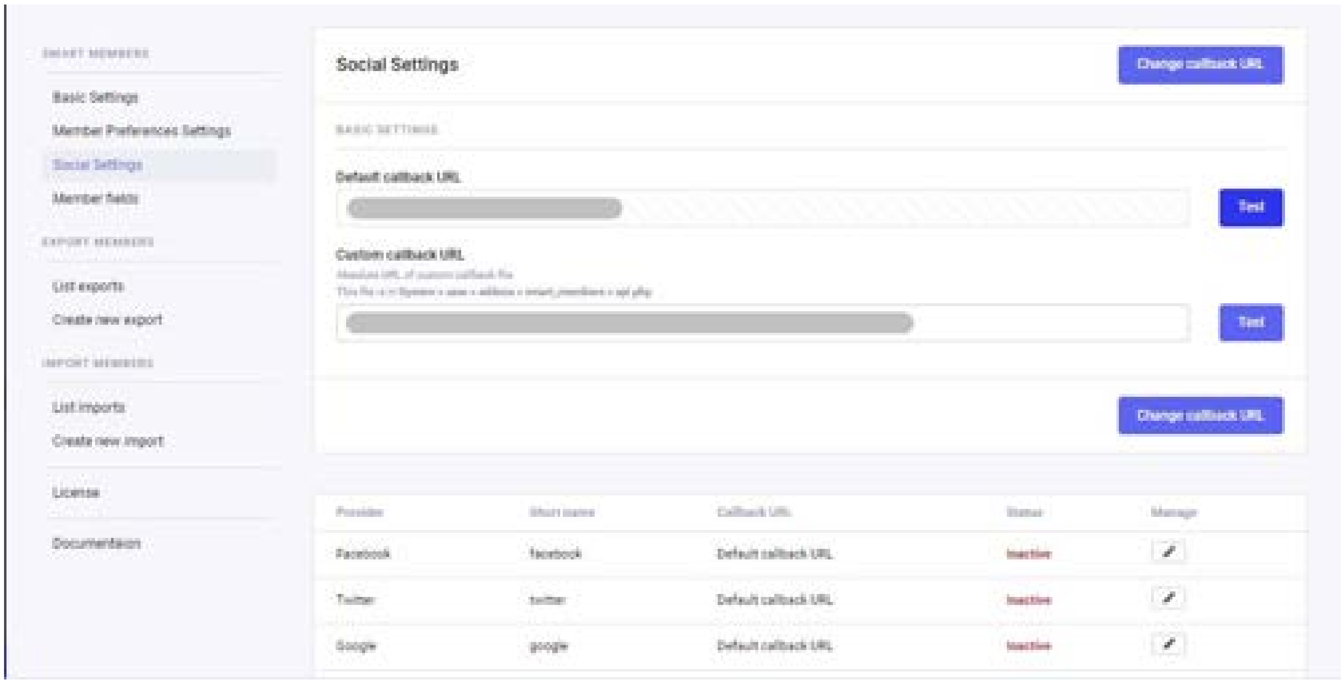
Maximum height
1500

Maximum file size (kb)
50

Social Settings

Social Login provides users to login with their social accounts. User can use login with Facebook, Twitter, Google and 20 more Social sites to login.

Social Settings List page



Social Settings

BASIC SETTINGS

Default callback URL

[redacted]

Custom callback URL

Absolute URL of custom callback file

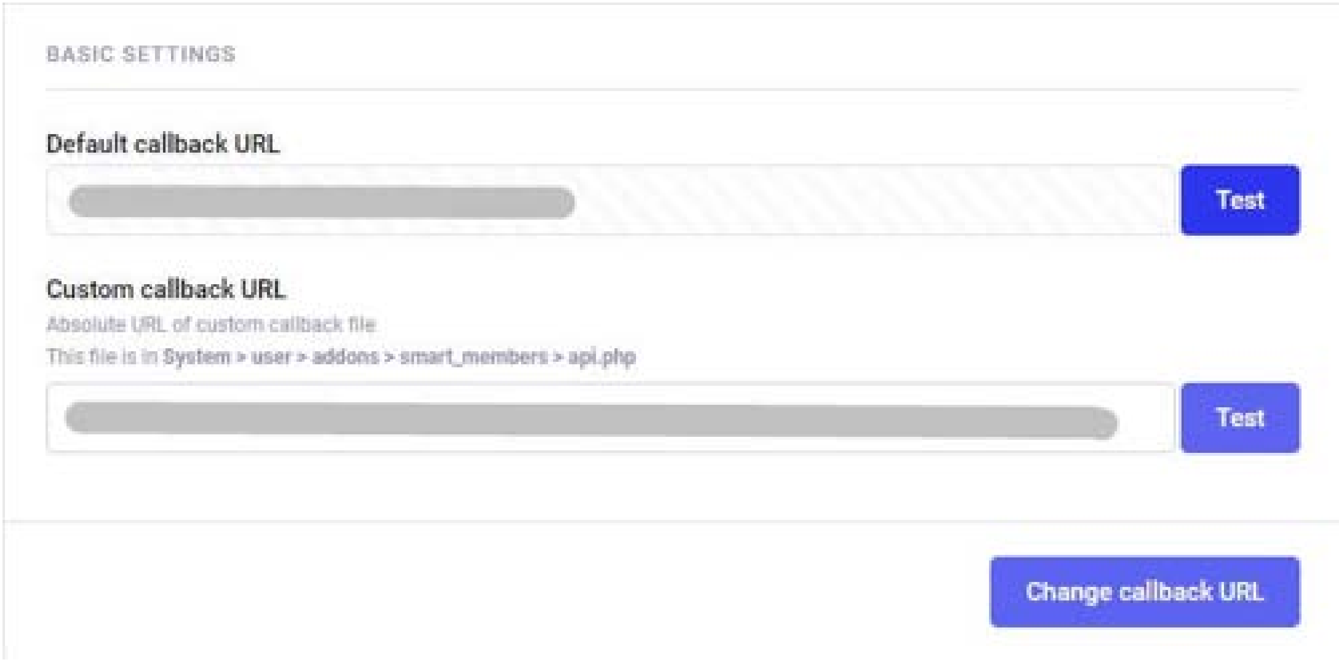
This file is in System > user > addons > smart_members > api.php

[redacted]

Provider	Short name	Callback URL	Status	Manage
Facebook	facebook	Default callback URL	Inactive	✎
Twitter	twitter	Default callback URL	Inactive	✎
Google	google	Default callback URL	Inactive	✎

Every Social login API will ask for a callback URL. You can found a Callback URL at social Settings List page.

Callback URL:



BASIC SETTINGS

Default callback URL

[redacted]

Custom callback URL

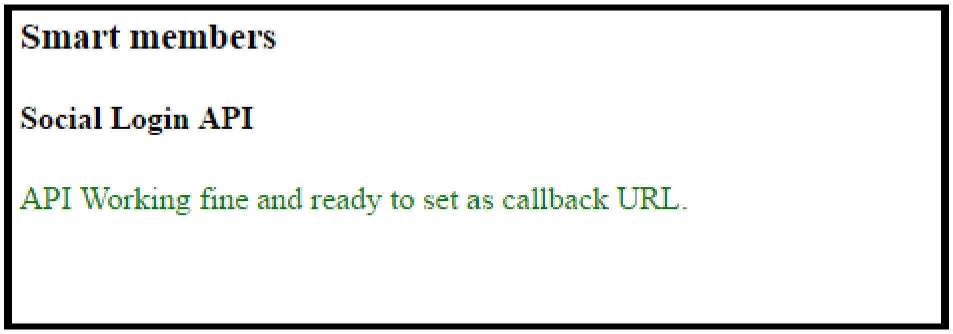
Absolute URL of custom callback file

This file is in System > user > addons > smart_members > api.php

[redacted]

Test the link is working or not by clicking TEST button at right. If link is not working please do check the link path is correct or not.

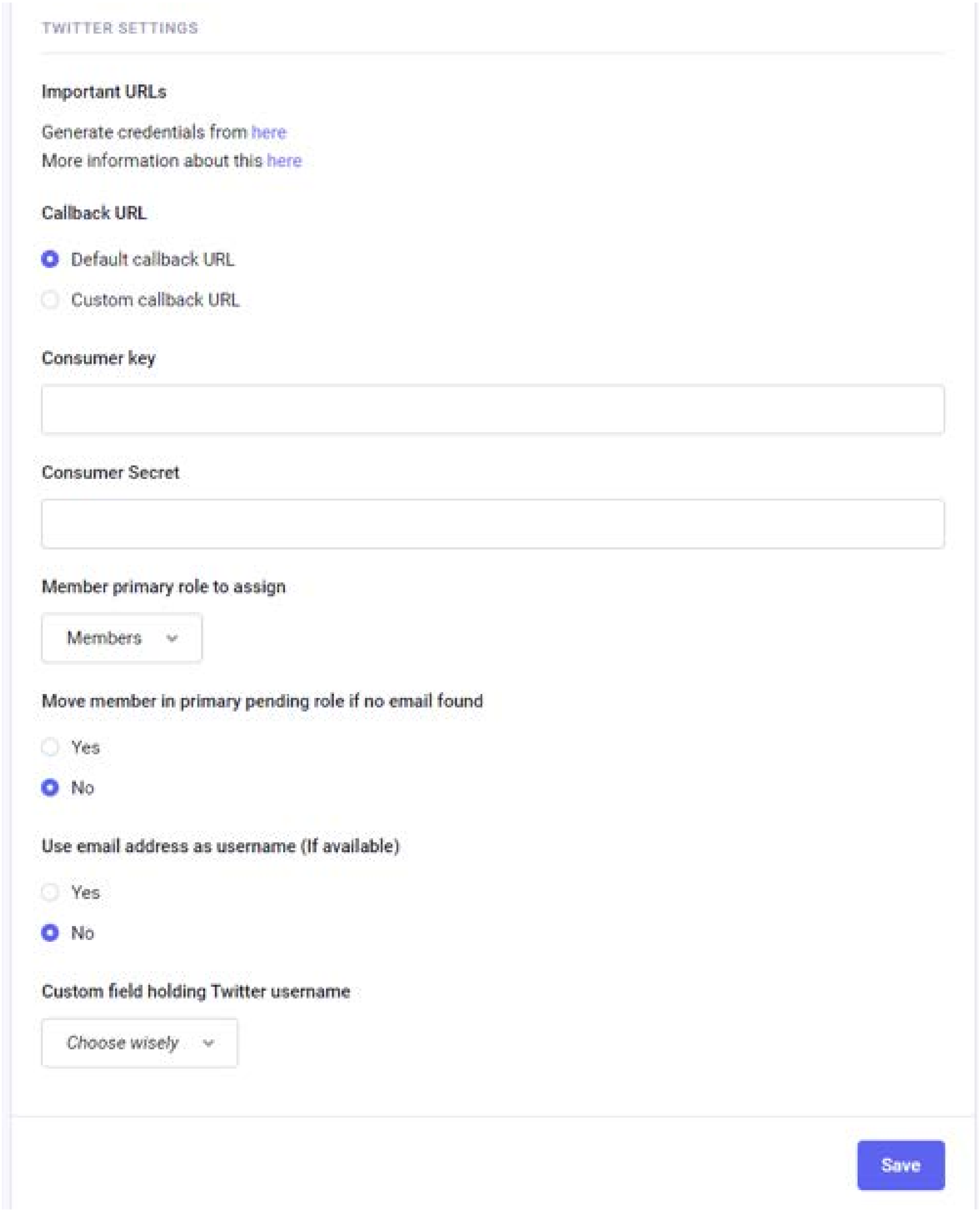
If Test goes success, You will see this page:


Smart members

Social Login API

API Working fine and ready to set as callback URL.

Social Settings Form:


TWITTER SETTINGS

Important URLs

Generate credentials from here
More information about this here

Callback URL

[✓ Default callback URL] Custom callback URL

Consumer key

Consumer Secret

Member primary role to assign: Members

Move member in primary pending role if no email found

Yes [✓ No]

Use email address as username (If available)

Yes [✓ No]

Custom field holding Twitter username: Choose wisely

Available Social sites:

1. Facebook

2. Twitter
3. Google
4. Live
5. Yahoo
6. Foursquare
7. GitHub
8. px500
9. BitBucket
10. Disqus
11. Dribbble
12. Dropbox
13. GitLab
14. Instagram
15. LastFM
16. MailChimp
17. Slack
18. SoundCloud
19. Vimeo
20. Tumblr

Facebook

To activate Facebook method in Social Login settings:

1. Generate Application ID and Application Secret From this link: https://developers.facebook.com/quickstarts/?platform=web
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need. Facebook API need exact callback URL. So

you need to enter the callback URL with done parameter. (EX: if your callback URL is http://www.example.com/?ACT=85, Your callback URL for facebook will be http://www.example.com/?ACT=229&hauth_done=Facebook

3. Once you generate Application ID and Application Secret, Go to social settings page in Smart Members Pro and Edit Facebook settings.
4. Put your Application ID and Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Twitter

To activate Twitter method in Social Login settings:

1. Generate Consumer key and Consumer Secret From this link: https://apps.twitter.com/app/new
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Consumer key and Consumer Secret, Go to social settings page in Smart Members Pro and Edit Twitter settings.
4. Put your Consumer key and Consumer Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Google

To activate Google method in Social Login settings:

1. Generate Client ID andClient Secret From this link: https://console.developers.google.com/
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need. Google API need exact callback URL. So you need to enter the callback URL with done parameter. (EX: if your callback

URL is http://www.example.com/?ACT=85, Your callback URL for Google will be http://www.example.com/?ACT=85&hauth.done=Google

3. Once you generate Client ID andClient Secret, Go to social settings page in Smart Members Pro and Edit Google settings.
4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Live

To activate Live method in Social Login settings:

1. Generate Client ID and Client Secret From this link: https://apps.dev.microsoft.com/#/appList/create/sapi
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit Live settings.
4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Yahoo

To activate Yahoo method in Social Login settings:

1. Generate Client ID and Client Secret From this link: https://developer.yahoo.com/apps/create/
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit Yahoo settings.

4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Foursquare

To activate Foursquare method in Social Login settings:

1. Generate Client ID and Client Secret From this link: https://foursquare.com/developers/apps
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit Foursquare settings.
4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

GitHub

To activate GitHub method in Social Login settings:

1. Generate Client ID and Client Secret From this link: https://github.com/settings/applications/new
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit GitHub settings.
4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

px500

To activate px500 method in Social Login settings:

1. Generate Customer ID and Customer Secret From this link: https://500px.com/settings/applications
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Customer ID and Customer Secret, Go to social settings page in Smart Members Pro and Edit px500 settings.
4. Put your Customer ID and Customer Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

BitBucket

To activate BitBucket method in Social Login settings:

1. Generate Customer Key and Customer Secret From this link: https://bitbucket.org/account/user/testing_eecms/api
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Customer Key and Customer Secret, Go to social settings page in Smart Members Pro and Edit BitBucket settings.
4. Put your Customer Key and Customer Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Disqus

To activate Disqus method in Social Login settings:

1. Generate Public Key and Secret Key From this link: https://disqus.com/api/applications/register/

2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Public Key and Secret Key, Go to social settings page in Smart Members Pro and Edit Disqus settings.
4. Put your Public Key and Secret Key in their fields. Select the callback URL you have put in API.
5. Save the settings.

Dribbble

To activate Dribbble method in Social Login settings:

1. Generate Client ID and Client Secret From this link: https://dribbble.com/account/applications/new
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit Dribbble settings.
4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Dropbox

To activate Dropbox method in Social Login settings:

1. Generate App Key and App Secret From this link: https://www.dropbox.com/developers/apps/create
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate App Key and App Secret, Go to social settings page in Smart Members Pro and Edit Dropbox settings.

4. Put your App Key and App Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

GitLab

To activate GitLab method in Social Login settings:

1. Generate Application ID and Application Secret From this link: https://gitlab.com/oauth/applications
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Application ID and Application Secret, Go to social settings page in Smart Members Pro and Edit GitLab settings.
4. Put your Application ID and Application Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Instagram

To activate Instagram method in Social Login settings:

1. Generate Client ID and Client Secret From this link: https://www.instagram.com/developer/clients/register/
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit Instagram settings.
4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

LastFM

To activate LastFM method in Social Login settings:

1. Generate API key and Shared Secret From this link: http://www.last.fm/api/account/create
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate API key and Shared Secret, Go to social settings page in Smart Members Pro and Edit LastFM settings.
4. Put your API key and Shared Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

MailChimp

To activate MailChimp method in Social Login settings:

1. Generate Client ID and Client Secret From this link: https://us14.admin.mailchimp.com/account/oauth2/client/
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit MailChimp settings.
4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Slack

To activate Slack method in Social Login settings:

1. Generate Client ID and Client Secret From this link: https://api.slack.com/apps/new

2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit Slack settings.
4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

SoundCloud

To activate SoundCloud method in Social Login settings:

1. Generate Client ID and Client Secret From this link: http://soundcloud.com/you/apps/new
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit SoundCloud settings.
4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Vimeo

To activate Vimeo method in Social Login settings:

1. Generate Client ID and Client Secret From this link: https://developer.vimeo.com/apps/new
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate Client ID and Client Secret, Go to social settings page in Smart Members Pro and Edit Vimeo settings.

4. Put your Client ID and Client Secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Tumblr

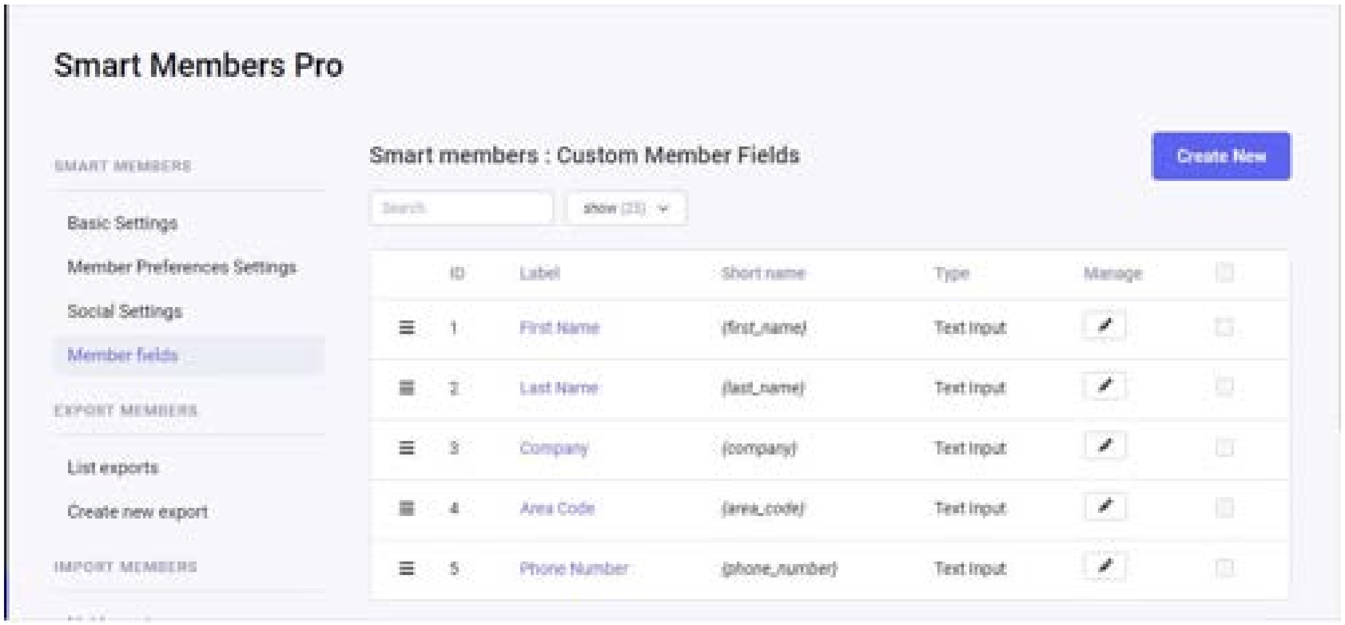
To activate Tumblr method in Social Login settings:

1. Generate OAuth consumer key and OAuth consumer secret From this link: https://www.tumblr.com/oauth/register
2. Enter Callback URL in API form. You can enter Default callback URL or Custom Callback URL as per your need.
3. Once you generate OAuth consumer key and OAuth consumer secret, Go to social settings page in Smart Members Pro and Edit Tumblr settings.
4. Put your OAuth consumer key and OAuth consumer secret in their fields. Select the callback URL you have put in API.
5. Save the settings.

Member Fields

ExpressionEngine has limit for member field types. One can only select Text Input, Textarea and select dropdown as custom member fields. We introduce Multi select box, Radio buttons, Checkboxes and File fields to use as custom member fields.

Member Fields setting List Page:



Smart Members Pro

SMART MEMBERS
- Basic Settings
- Member Preferences Settings
- Social Settings
- Member fields

EXPORT MEMBERS
- List exports
- Create new export

IMPORT MEMBERS

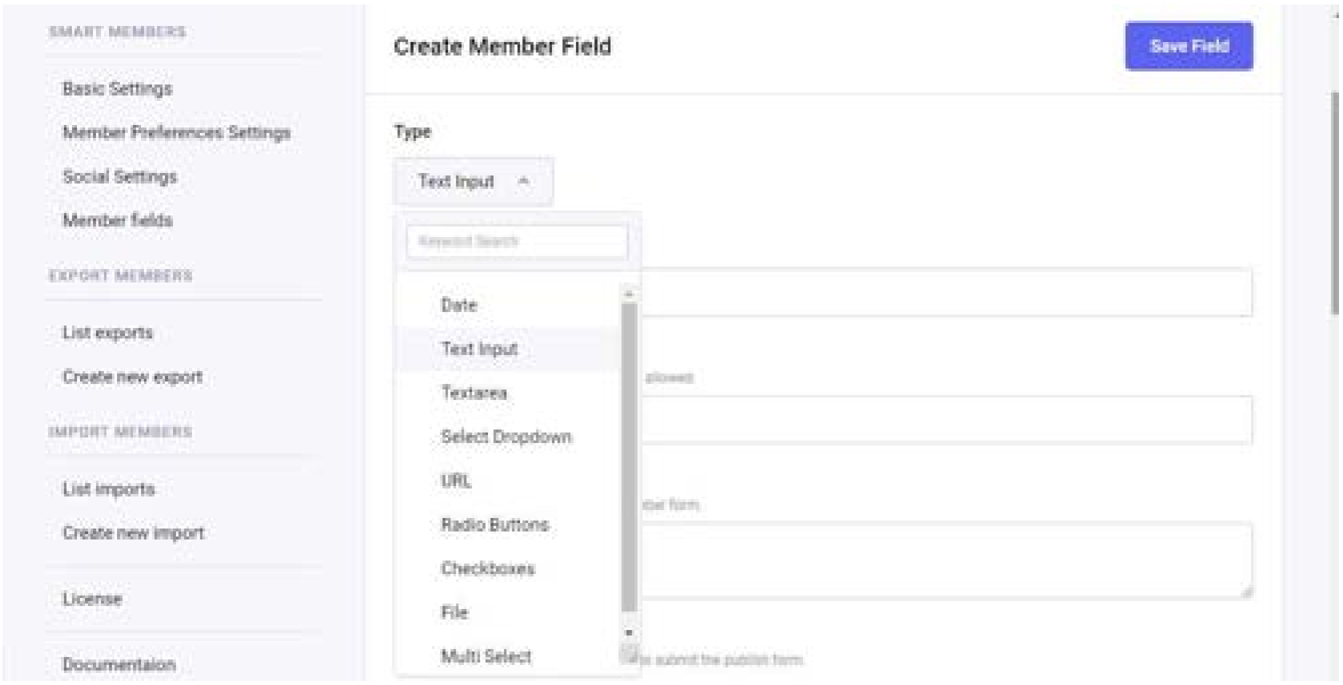
Smart members : Custom Member Fields

Create New

Search | show (25)

	ID	Label	Short name	Type	Manage	
☰	1	First Name	{first_name}	Text Input	✎	☐
☰	2	Last Name	{last_name}	Text Input	✎	☐
☰	3	Company	{company}	Text Input	✎	☐
☰	4	Area Code	{area_code}	Text Input	✎	☐
☰	5	Phone Number	{phone_number}	Text Input	✎	☐

Create Field form:



SMART MEMBERS
- Basic Settings
- Member Preferences Settings
- Social Settings
- Member fields

EXPORT MEMBERS
- List exports
- Create new export

IMPORT MEMBERS
- List imports
- Create new import

License

Documentaion

Create Member Field

Save Field

Type

Text Input

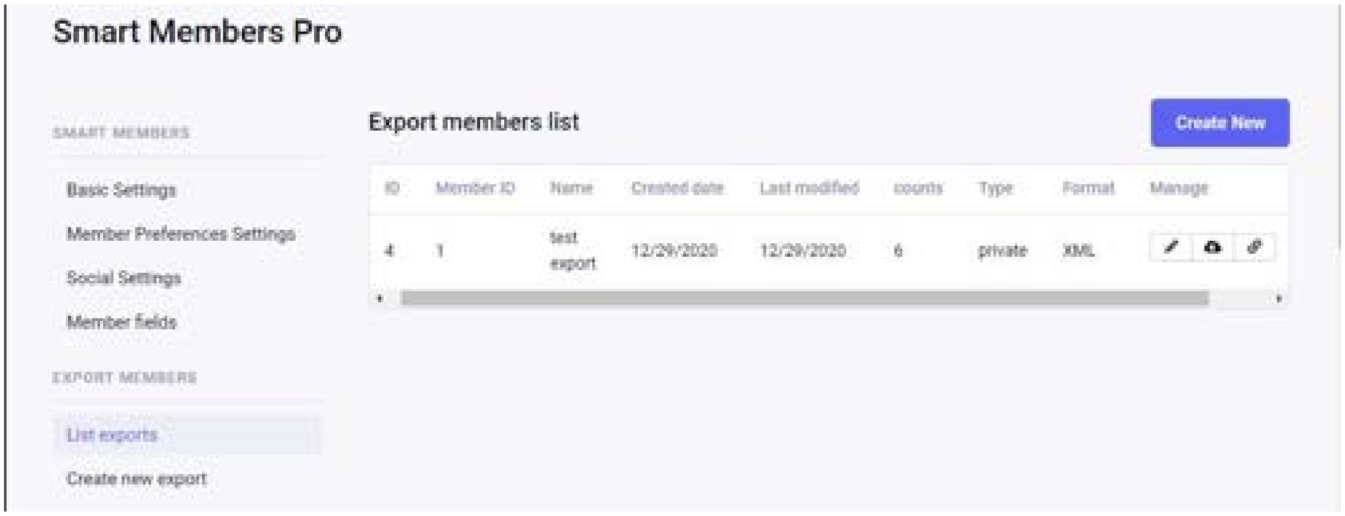
Keyword Search

- Date
- Text Input
- Textarea
- Select Dropdown
- URL
- Radio Buttons
- Checkboxes
- File
- Multi Select

Export member

You can now export members with selected fields filtered by Groups. Exports are available to download in both CSV and XML format. You can save the exports to use it in future. Export can also made by out side of admin panel with auto generated links.

Export member List page:



Smart Members Pro

SMART MEMBERS

- Basic Settings
- Member Preferences Settings
- Social Settings
- Member fields

EXPORT MEMBERS

- List exports
- Create new export

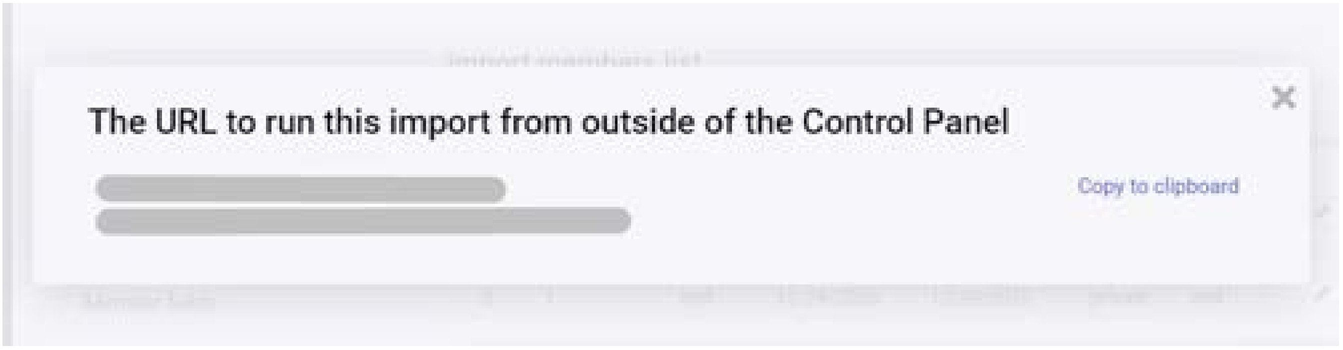
Export members list

Create New

ID	Member ID	Name	Created date	Last modified	counts	Type	Format	Manage
4	1	test export	12/29/2020	12/29/2020	6	private	XML	

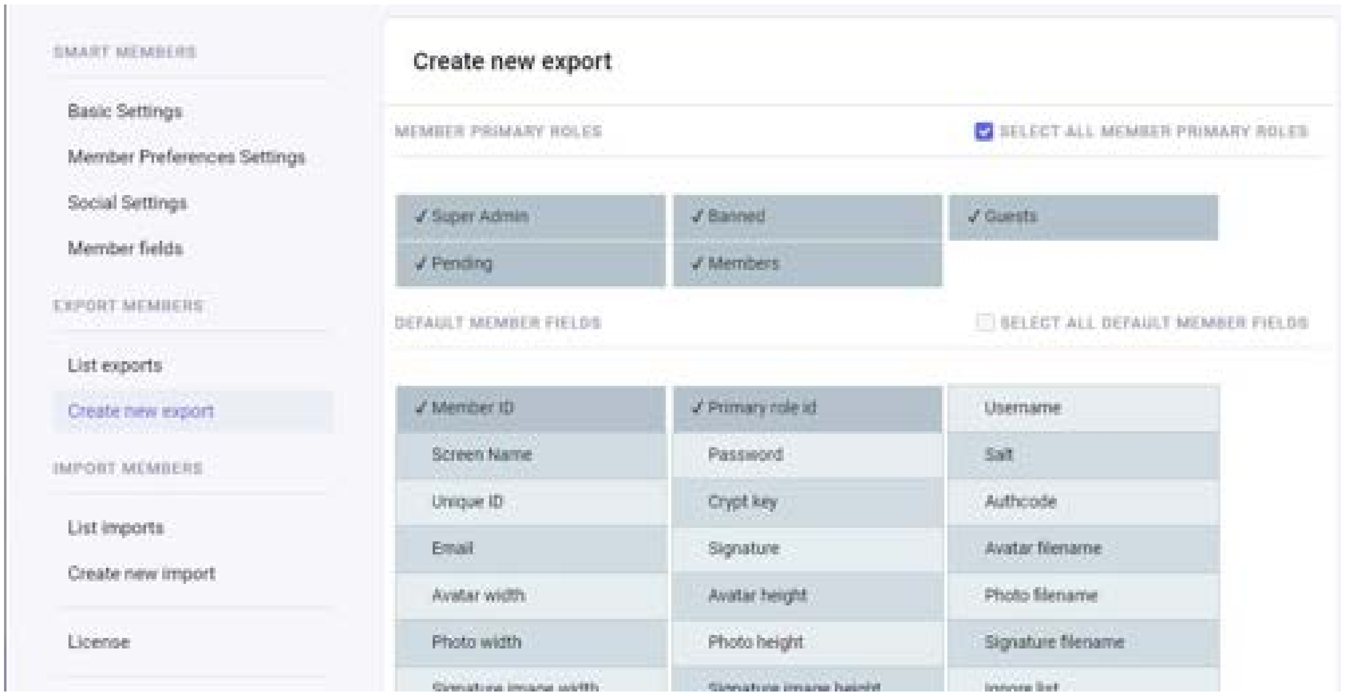
You can download, edit or Delete the exports from here. URL link is will popup the external URL to download export from outsite of EE. There is some security to download exports from outside of EE. If you make export private, No once can download it without login with your account. Public exports can download from outside of EE with anyone who logged in. If you want to download export from outside of EE without login, set “Access export URL without Login ?” to YES.

Popup URL to download export outside of EE:



The URL to run this import from outside of the Control Panel

Generate Export Form 1:



SMART MEMBERS

- Basic Settings
- Member Preferences Settings
- Social Settings
- Member fields

EXPORT MEMBERS

- List exports
- Create new export

IMPORT MEMBERS

- List imports
- Create new import

- License

Create new export

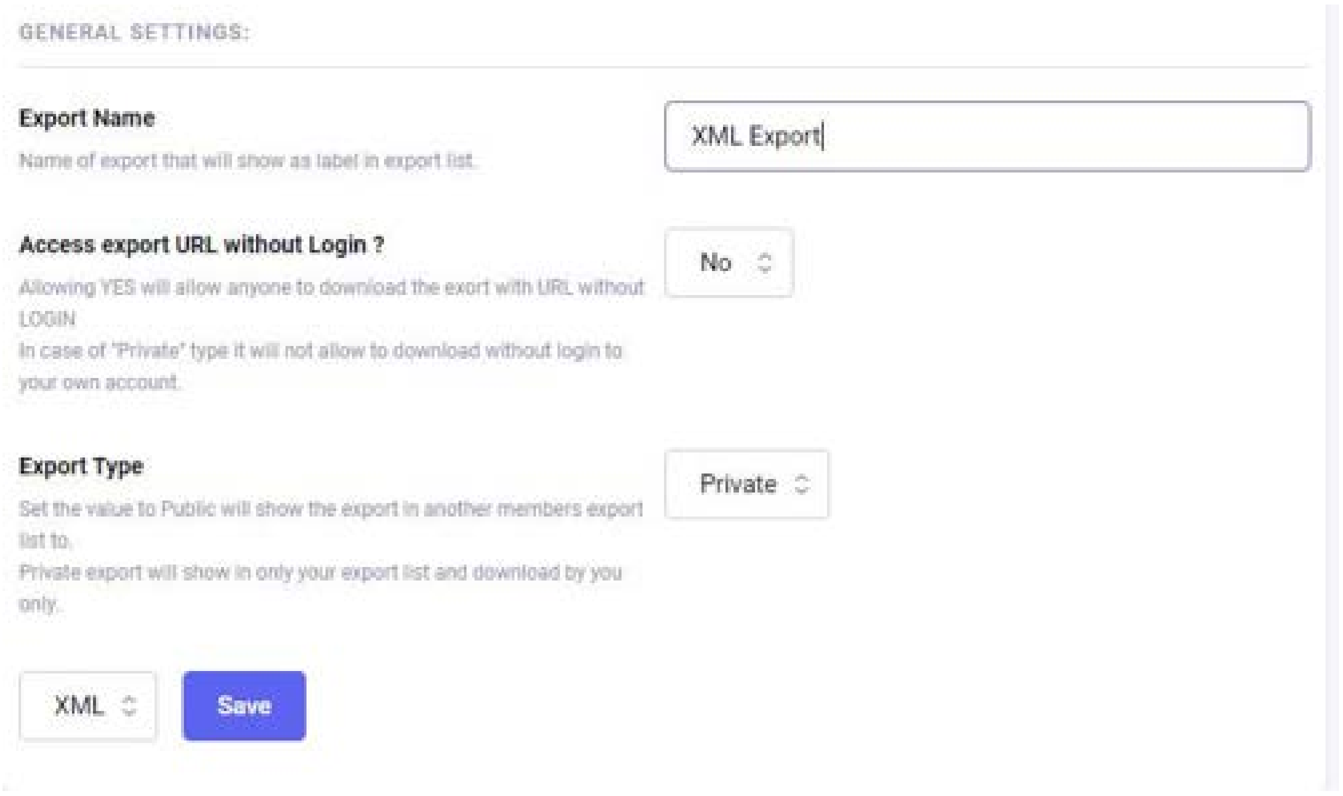
MEMBER PRIMARY ROLES ✓ SELECT ALL MEMBER PRIMARY ROLES

✓ Super Admin	✓ Banned	✓ Guests
✓ Pending	✓ Members	

DEFAULT MEMBER FIELDS ☐ SELECT ALL DEFAULT MEMBER FIELDS

✓ Member ID	✓ Primary role id	Username
Screen Name	Password	Salt
Unique ID	Crypt key	Authcode
Email	Signature	Avatar filename
Avatar width	Avatar height	Photo filename
Photo width	Photo height	Signature filename
Signature image width	Signature image height	Ignore list

Generate Export Form 2:



GENERAL SETTINGS:

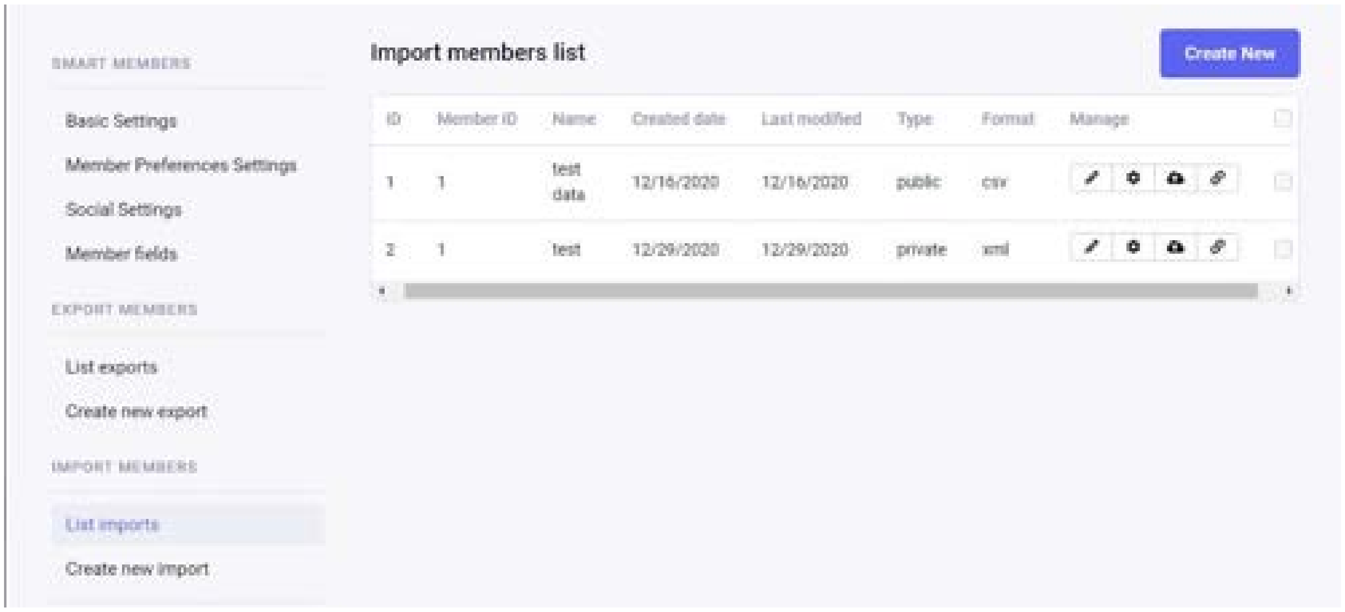
Export Name Name of export that will show as label in export list.	XML Export
Access export URL without Login ? Allowing YES will allow anyone to download the exort with URL without LOGIN In case of "Private" type it will not allow to download without login to your own account.	No
Export Type Set the value to Public will show the export in another members export list to. Private export will show in only your export list and download by you only.	Private

XML Save

Import Members

One can import members with selected fields with filtration. Filtration is done on many basis such as same email, same member ID, same screen name, Sanitize the unique fields or not etc. Import can done from both CSV and XML files at path of both server path or URL. You can save the imports to use it in future. Imports can also done by out side of admin panel with auto generated links.

Import Members List page:

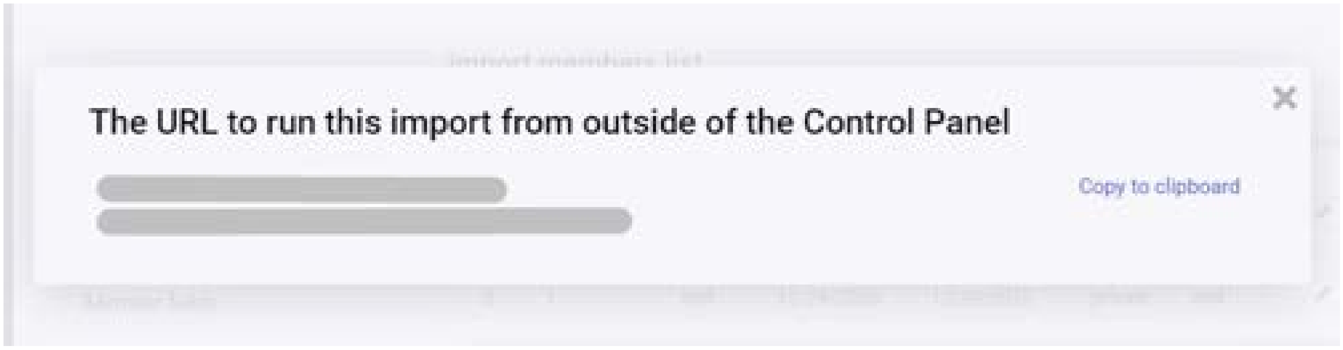


Import members list

ID	Member ID	Name	Created date	Last modified	Type	Format	Manage
1	1	test data	12/16/2020	12/16/2020	public	csv	
2	1	test	12/29/2020	12/29/2020	private	xml	

You can run, edit or Delete the imports from here. URL link is will popup the external URL to run export from outsite of EE. There is some security to run imports from outside of EE. If you make import private, No once can run it without login with your account. Public imports can run from outside of EE with anyone who logged in. If you want to run import from outside of EE without login, set “Access import URL without Login ?” to YES.

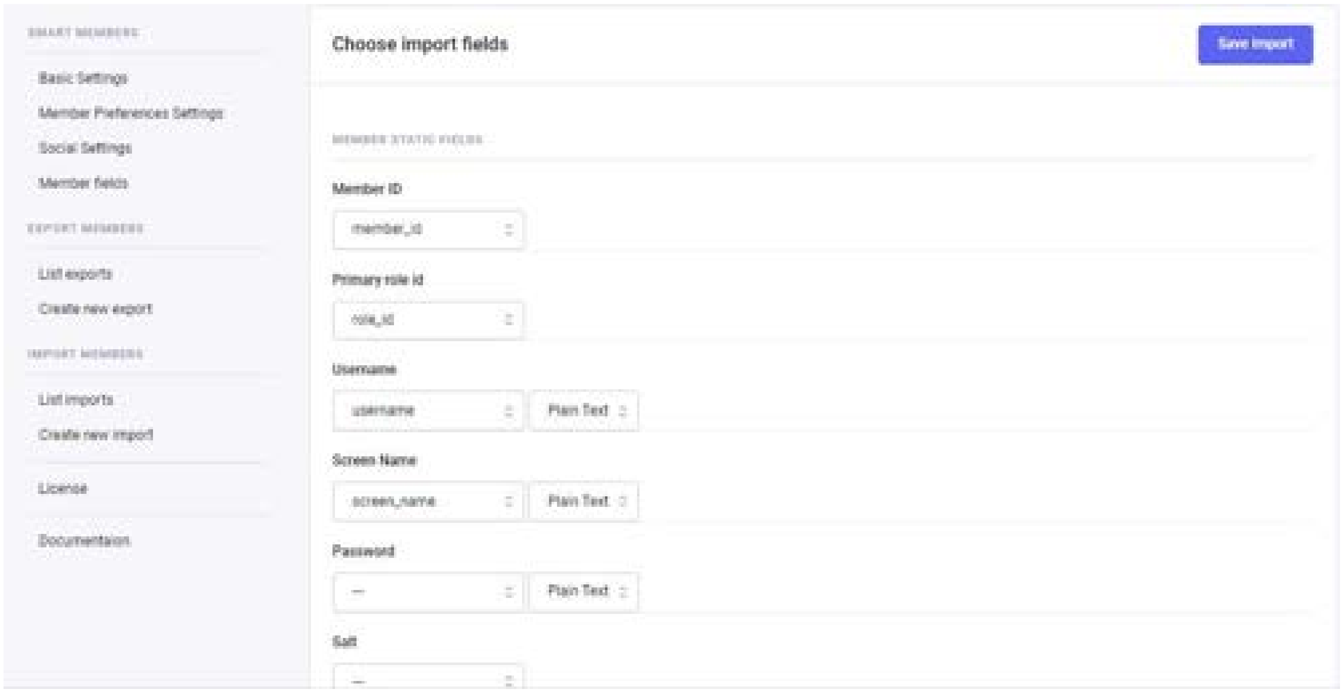
Popup URL to run import outside of EE:



The URL to run this import from outside of the Control Panel

[redacted]

Import Member Choose fields:



Choose import fields

MEMBER STATIC FIELDS

Member ID: member_id

Primary role id: role_id

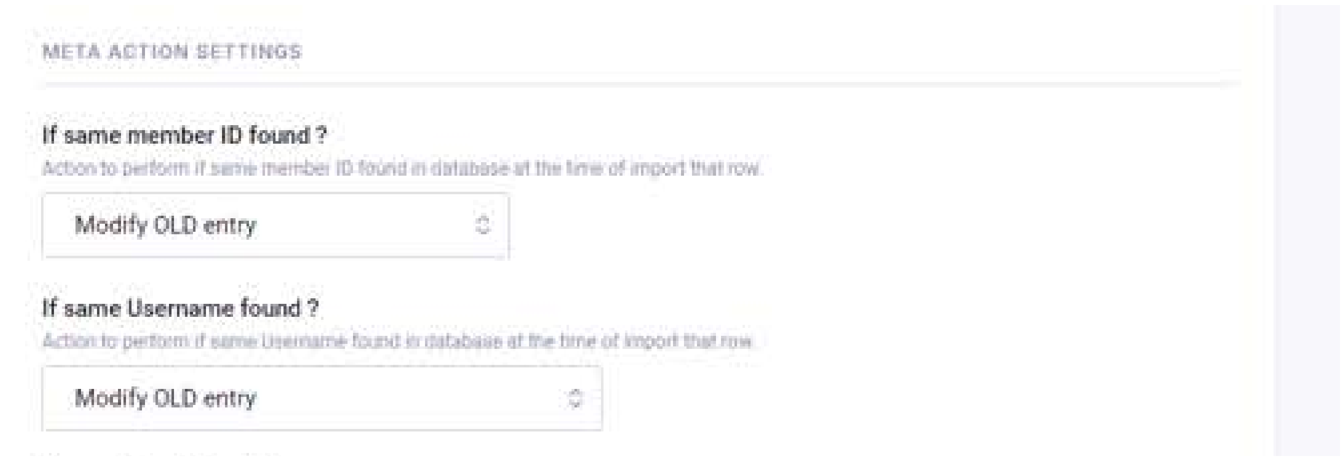
Username: username | Plain Text

Screen Name: screen_name | Plain Text

Password: — | Plain Text

Salt: —

Import Member setup Configurations:



META ACTION SETTINGS

If same member ID found ?

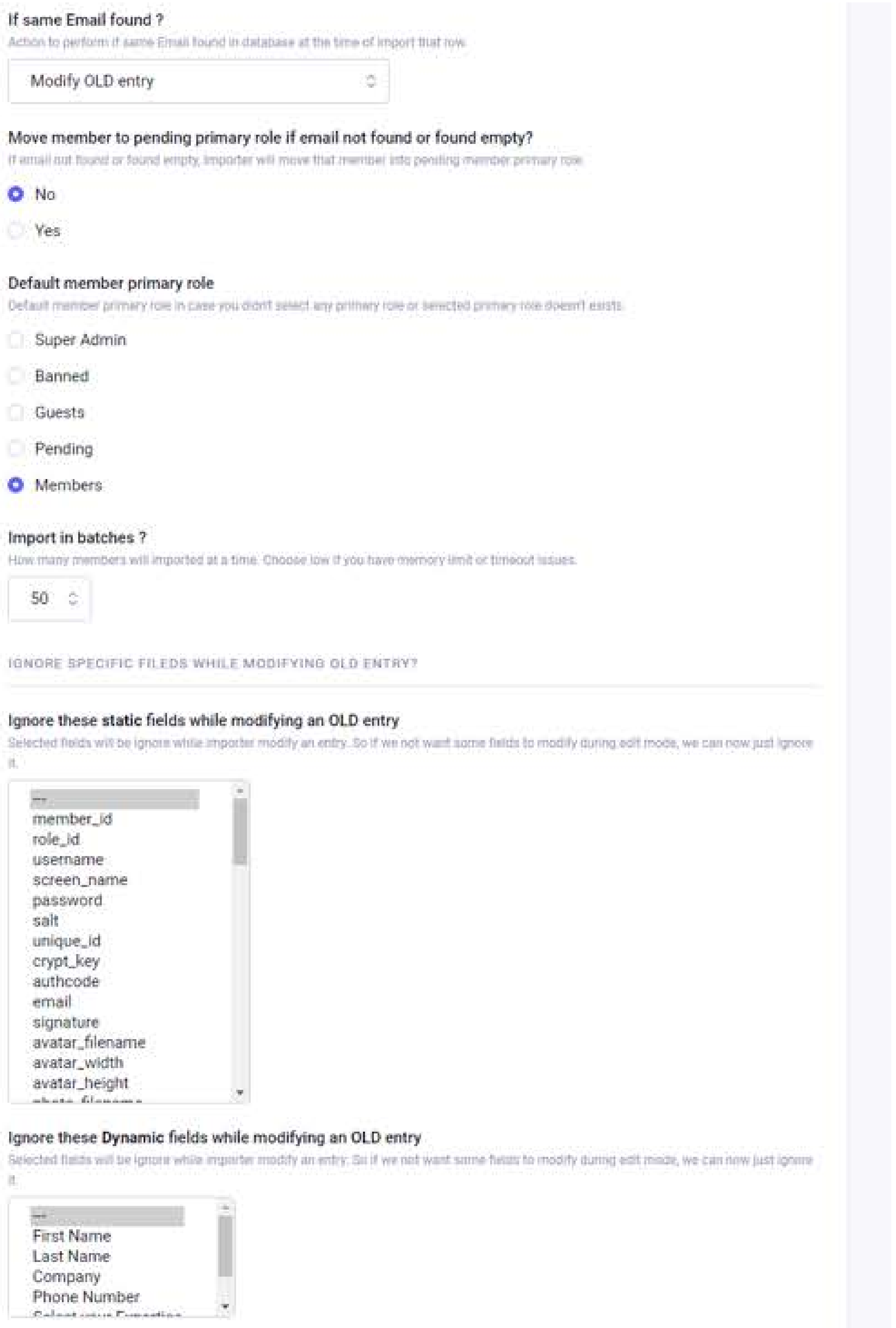
Action to perform if same member ID found in database at the time of import that row.

Modify OLD entry

If same Username found ?

Action to perform if same Username found in database at the time of import that row.

Modify OLD entry



If same Email found ?

Action to perform if same Email found in database at the time of import that row.

Modify OLD entry

Move member to pending primary role if email not found or found empty?

If email not found or found empty, importer will move that member into pending member primary role.

[✓ No]
Yes

Default member primary role

Default member primary role in case you didn't select any primary role or selected primary role doesn't exists.

Super Admin
Banned
Guests
Pending
[✓ Members]

Import in batches ?

How many members will imported at a time. Choose low if you have memory limit or timeout issues.

50

IGNORE SPECIFIC FILEDS WHILE MODIFYING OLD ENTRY?

Ignore these static fields while modifying an OLD entry

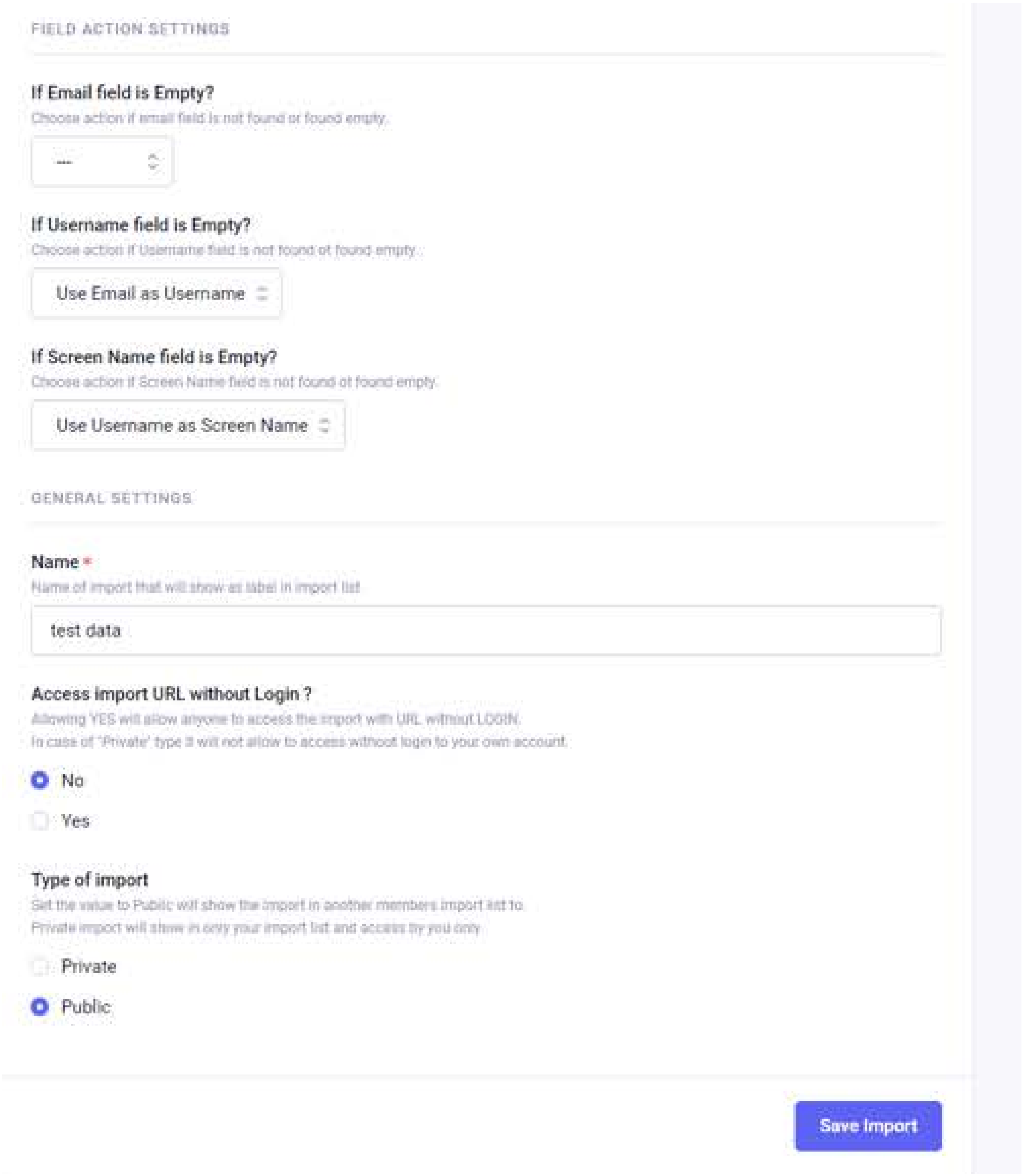
Selected fields will be ignore while importer modify an entry. So if we not want some fields to modify during edit mode, we can now just ignore it.

--
member_id
role_id
username
screen_name
password
salt
unique_id
crypt_key
authcode
email
signature
avatar_filename
avatar_width
avatar_height

Ignore these Dynamic fields while modifying an OLD entry

Selected fields will be ignore while importer modify an entry. So if we not want some fields to modify during edit mode, we can now just ignore it.

--
First Name
Last Name
Company
Phone Number



FIELD ACTION SETTINGS

If Email field is Empty?
Choose action if email field is not found or found empty.

If Username field is Empty?
Choose action if Username field is not found or found empty.
Use Email as Username

If Screen Name field is Empty?
Choose action if Screen Name field is not found or found empty.
Use Username as Screen Name

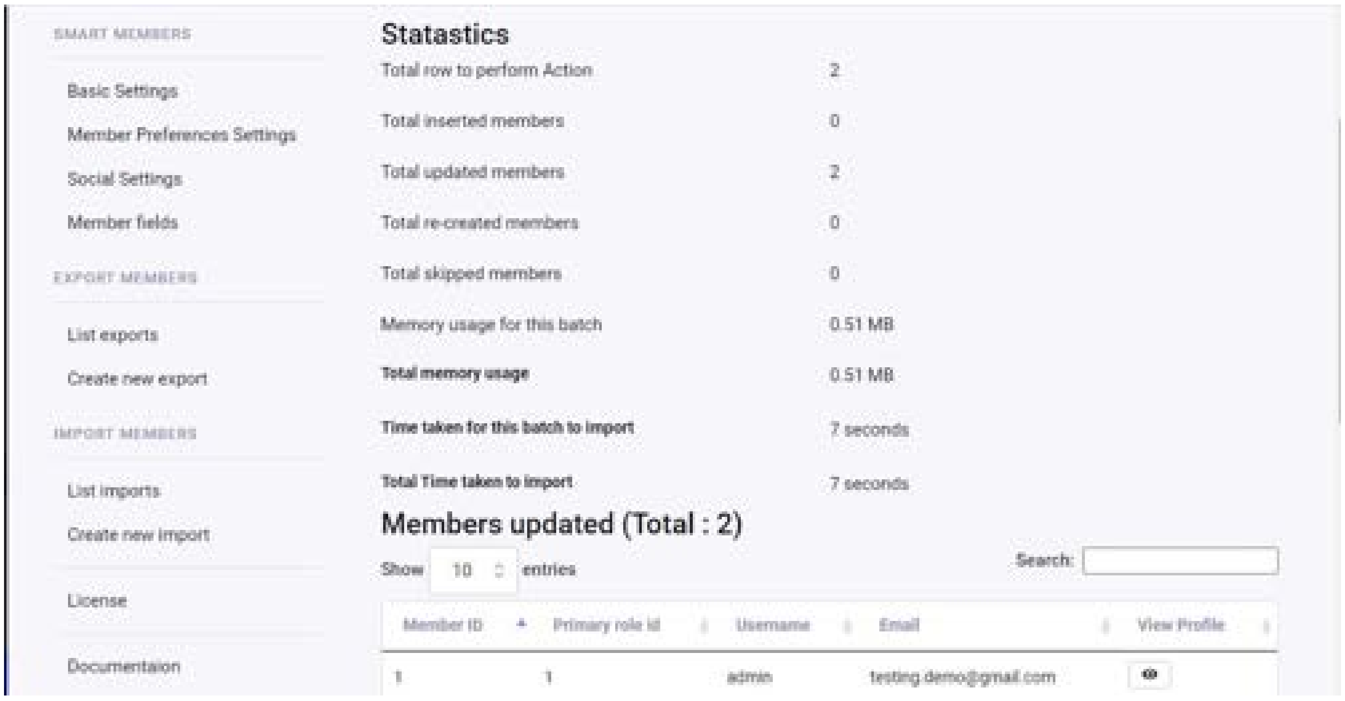
GENERAL SETTINGS

Name *
Name of import that will show as label in import list
test data

Access import URL without Login ?
Allowing YES will allow anyone to access the import with URL without LOGIN.
In case of "Private" type it will not allow to access without login to your own account.
[✓ No] Yes

Type of import
Set the value to Public will show the import in another members import list to.
Private import will show in only your import list and access by you only
Private [✓ Public]

Run Import page inside EE:


SMART MEMBERS

Basic Settings
Member Preferences Settings
Social Settings
Member fields

EXPORT MEMBERS

List exports
Create new export

IMPORT MEMBERS

List imports
Create new import

License

Documentaion

Statastics

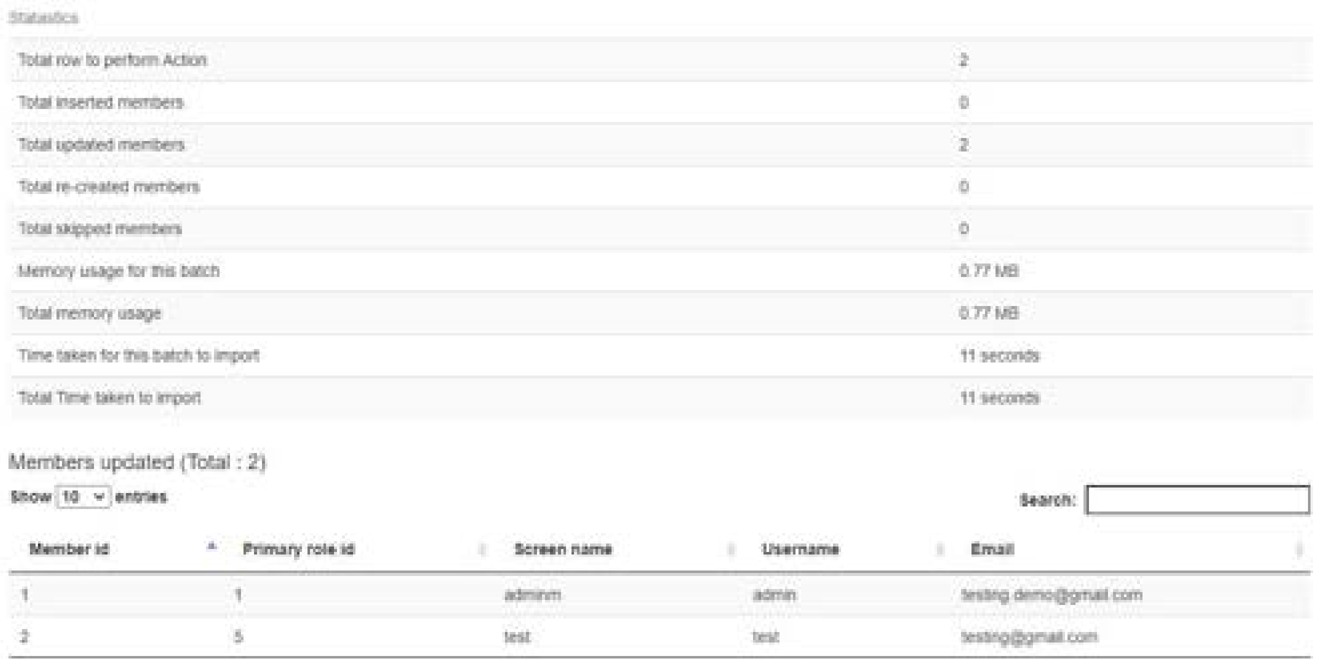
Total row to perform Action	2
Total inserted members	0
Total updated members	2
Total re-created members	0
Total skipped members	0
Memory usage for this batch	0.51 MB
Total memory usage	0.51 MB
Time taken for this batch to import	7 seconds
Total Time taken to import	7 seconds

Members updated (Total : 2)

Show 10 entries Search:

Member ID	Primary role id	Username	Email	View Profile
1	1	admin	testing.demo@gmail.com	👁

Run Import page outside EE:


Statastics

Total row to perform Action	2
Total inserted members	0
Total updated members	2
Total re-created members	0
Total skipped members	0
Memory usage for this batch	0.77 MB
Total memory usage	0.77 MB
Time taken for this batch to import	11 seconds
Total Time taken to import	11 seconds

Members updated (Total : 2)

Show 10 entries Search:

Member id	Primary role id	Screen name	Username	Email
1	1	adminm	admin	testing.demo@gmail.com
2	5	test	test	testing@gmail.com

Custom Sidebar

We provide one “customSidebar” function in the MCP file. You can add more menus in the sidebar of the addon using this function.

```
/* To get dynamic menu in the addon */
   function customSidebar(){
       $sidebar = ee('CP/Sidebar')->make();
       $this->navSettings  = $sidebar->addHeader('Test Li
   }
```

Registration

Registration module is use for register a user with your site. A registration module will process and react as per settings we have passed in member preferences and addon setting form.

Tag for registration module will look like this.

```
 {exp:smart_members:register} Content data {/exp:smart_me
```

Parameters

1. group_id
2. role_id
3. allowed_groups
4. rule:FIELD_NAME
5. attr:ATTRIBUTES
6. return
7. error_reporting
8. wrap_errors

9. on_submit
10. secure_action
11. secure_return
12. enable_recaptcha

Fields

1. group_id
2. username
3. email
4. password
5. password_confirm
6. avatar_filename
7. photo_filename
8. sig_img_filename
9. yahoo_im
10. url
11. location
12. occupation
13. interests
14. aol_im
15. msn_im
16. icq
17. bio
18. signature
19. captcha

20. CUSTOM_FIELD

21. CUSTOM_CHECKBOX_FIELD (Pro Feature)

22. CUSTOM_RADIO_FIELD (Pro Feature)

23. CUSTOM_MULTI_SELECT_FIELD (Pro Feature)

24. CUSTOM_FILE_FIELD (Pro Feature)

Following Parameters can be use in Registration form

group_id (For <= EE5)

Group id of member groups to put the member in any specific group.

Example:

```
group_id = "5"
```

role_id (For EE6)

Primary role id of member to put the member in any specific primary role.

Example:

```
role_id =  "5"
```

allowed_groups

Allowed groups parameter is used to give options to member for select any group from them. Use input field of group_id to enter the group id from defined allowed groups.

For EE6, you will pass the primary role ID to this parameter.

Example:

```
allowed_groups = "5|6|7"
```

rule:FIELD_NAME

Rule parameter is use to give custom rules to fields separate by pipe (|).

Example:

```
rule:username  = "required"
rule:email     = "required|valid_email|is_unique[members
```

attr:ATTRIBUTES

Parameter to add attributes in form. We can add classes, ids etc.

Example:

```
attr:id = "form_id"
attr:class = "form_class"
attr:name = "form_name"
attr:data-id = "form_data_id"
```

return

Return to any specific page after successful submission of form.

Example:

```
return = "smart-members/profile"
```

error_reporting

Error reporting format is defined by this parameter. It can be either “inline” or “outline”.

“Inline” error reporting will show the error in same page.
“outline” error reporting will show error in EE gray box in new page.

Example:

```
error_reporting="inline"
```

wrap_errors

Use this parameter to wrap forms error in any span or div if set error_reporting=”inline”.

Example:

```
wrap_errors="<span class='error-inline'>|</span>"
```

on_submit

This parameter allows us to call any Javascript function on submit of form

Example:

```
on_submit="call_me( )"
```

secure_action

Secure action will post the data on secure site i.e., https

Example:

```
secure_action="yes"
```

secure_return

Secure return will return the page after submit of form on secure site i.e., https

Example:

```
secure_return="yes"
```

enable_recaptcha

This parameter enables recaptcha instead of normal captcha.

1. Scenario EE2:
    1. If this parameter is set and API key and SECRET is not passed for recaptcha in backend, The normal captcha will show.
    2. If recaptcha API key and SECRET is passed in backend and this parameter not set:
        1. If the page is registration page and you have set captcha as required from member preferences, normal captcha will show.
        2. If the page is not registration page, neither captcha or recaptcha will show.

3. If recaptcha API key and SECRET is passed in backend and this parameter is set:
    1. If the page is registration page and you have set captcha as required from member preferences, The normal captcha will not show and recaptcha will override the settings.
    2. If the page is not registration page, recaptcha will show.
2. Scenario EE3:
    1. Same scenario like EE2. Only change is, there is backend member preference settings in EE3 that allows to not enter any captcha if member is logged in. So if you set this parameter and API key and SECRET is also passed, If you are logged in this recaptcha or captcha will not show until you set “Require CAPTCHA while logged in?” to “Yes” from backend member settings > captcha settings.

Example:

```
enable_recaptcha="yes"
```

Different methods that can be use by user are given below.

Input

```
<input type="email" name="username" placeholder="Email A
{error:username} //if error_reporting="inline"
```

Textarea

```
<textarea name="signature"></textarea>
{error:signature} //if error_reporting="inline"
```

Default File field (Avatar, Photo and Signature files)

```
<input type="file" name="avatar_filename" />
{error:avatar_filename} //if error_reporting="inline"
```

Select Dropdown: (For member group) (For <= EE5)

```
{data_group_id}
{if data_group_id:count == 1}
<select name="group_id">
    <option value=""></option>
    {/if}
    <option value = "{group_id_value}" > {group_id_label
    {if data_group_id:count == data_group_id:total_resul
</select>
{error:group_id} //if error_reporting="inline"
{/if}
{/data_group_id}
```

Select Dropdown: (For member primary role) (For EE6)

```
{data_group_id}
{if data_group_id:count == 1}

<select name="role_id">
<option value=""></option>
{/if}
<option value = "{group_id_value}" > {group_id_label}</c
{if data_group_id:count == data_group_id:total_results}
</select>
{error:role_id} //if error_reporting="inline"
{/if}
{/data_group_id}
```

Select Dropdown: (For custom field)

```
{data_CUSTOM_FIELD}
{if data_CUSTOM_FIELD:count == 1}
<select name="CUSTOM_FIELD">
<option value=""></option>
    {/if}
    <option value = "{CUSTOM_FIELD_value}"> {CUSTOM_FIEL
    {if data_CUSTOM_FIELD:count == data_CUSTOM_FIELD:tot
</select>
{error:CUSTOM_FIELD} //if error_reporting="inline"
{/if}
{/data_CUSTOM_FIELD}
```

Captcha

```
{if captcha}
<p>
    <label for="captcha">Please enter in the word you se
    {captcha}
    <input type="text" name="captcha" id="captcha" />
    {error:captcha}
</p>
{if:elseif recaptcha}
<p style="margin: 0;">
    <label for="recaptcha">Click the checkbox</label>{re
    {error:recaptcha}
</p>
{/if}
```

Accept terms

```
<input type="checkbox" name="accept_terms" value="y" />
{error:accept_terms} //if error_reporting="inline"
```

CUSTOM_CHECKBOX_FIELD (Pro Feature)

```
{data_CUSTOM_CHECKBOX_FIELD}
    {if data_CUSTOM_CHECKBOX_FIELD:count == 1}
    <p style="display: inline-flex; ">
        {/if}
        <input type="checkbox" name="CUSTOM_CHECKBOX_FIE
        <label for="test_{data_CUSTOM_CHECKBOX_FIELD:cou
        {if data_CUSTOM_CHECKBOX_FIELD:count == data_CUS
        {error:CUSTOM_CHECKBOX_FIELD}
    </p>
    {/if}
{/data_CUSTOM_CHECKBOX_FIELD}
```

CUSTOM_RADIO_FIELD (Pro Feature)

```
{data_CUSTOM_RADIO_FIELD}
    {if data_CUSTOM_RADIO_FIELD:count == 1}
    <p style="display: inline-flex;">
        {/if}
        <input type="radio" name="CUSTOM_RADIO_FIELD" va
        <label for="test_{data_CUSTOM_RADIO_FIELD:count}
        {if data_CUSTOM_RADIO_FIELD:count == data_CUSTOM
        {error:CUSTOM_RADIO_FIELD}
    </p>
    {/if}
{/data_CUSTOM_RADIO_FIELD}
```

CUSTOM_MULTI_SELECT_FIELD (Pro Feature)

```
{data_CUSTOM_MULTI_SELECT_FIELD}
{if data_CUSTOM_MULTI_SELECT_FIELD:count == 1}
<p style="display: inline-flex; ">
    <select multiple name="CUSTOM_MULTI_SELECT_FIELD[]">
        {/if}
        <option value="{CUSTOM_MULTI_SELECT_FIELD_value}
        {if data_CUSTOM_MULTI_SELECT_FIELD:count == data
    </select>
    {error:CUSTOM_MULTI_SELECT_FIELD}
</p>
{/if}
{/data_CUSTOM_MULTI_SELECT_FIELD}
```

CUSTOM_FILE_FIELD (Pro Feature)

```
<p>
    Old file name: {CUSTOM_FILE_FIELD}<br>
    <input type="file" name="CUSTOM_FILE_FIELD">
    {error:CUSTOM_FILE_FIELD}
</p>
```

Example (For <= EE5):

```
{exp:smart_members:register
     allowed_groups="6|7|5"
```

```
    rule:username="required|valid_email|min_length[5]"
    rule:password="required|matches[password_confirm]|mi
    rule:password_confirm="required|min_length[5]"
    rule:group_id="required"
    rule:state="required"
    rule:first_name="required"

    attr:id="registration_id"
    attr:class="registration_class"
    attr:name="registration-form"
    attr:data-id="registration_data_id_attr"

    return="smart-members/edit-profile"
    enable_recaptcha="yes"
    error_reporting="inline"
    wrap_errors="<span class='error-inline'>|</span>"

    on_submit="registration()"
}
    <p>
        <input type="text" name="CUSTOM_FIELD" placehold
        {error:CUSTOM_FIELD}
    </p>
    {data_group_id}
    {if data_group_id:count == 1}
    <p>
```

```
        <select name="group_id">
            <option value=""></option>
            {/if}
            <option value="{group_id_value}">{group_id_l
            {if data_group_id:count == data_group_id:tot
        </select>
        {error:group_id}
    </p>
    {/if}
    {/data_group_id}
    <p>
        <input type="text" name="username" placeholder="
        {error:username}
    </p>
    <p>
        <input type="text" name="email" placeholder="Ema
        {error:email}
    </p>

    <p>
        <input type="password" name="password" placehold
        {error:password}
    </p>
    <p>
        <input type="password" name="password_confirm">
        {error:password_confirm}
    </p>
    <p>
        <input type="file" name="avatar_filename" />
        {error:avatar_filename}
    </p>
    {if captcha}
    <p>
        <label for="captcha">Please enter in the word yo
        {captcha}
        <input type="text" name="captcha" id="captcha" /
        {error:captcha}
    </p>
    {if:elseif recaptcha}
    <p style="margin: 0;">
        <label for="recaptcha">Click the checkbox</label
        {recaptcha}
        {error:recaptcha}
    </p>
    {/if}
    <p>
        <input type="checkbox" name="accept_terms" value
        {error:accept_terms}
    </p>
    <div class="register-register-click cf">
       <input type="submit" class="register" value="Regi
    </div>
{/exp:smart_members:register}
```

Example (For EE6):

```
{exp:smart_members:register
allowed_groups="6|7|5"

rule:username="required|valid_email|min_length[5]"
rule:password="required|matches[password_confirm]|min_
rule:password_confirm="required|min_length[5]"
rule:role_id="required"
rule:state="required"
rule:first_name="required"

attr:id="registration_id"
attr:class="registration_class"
attr:name="registration-form"
attr:data-id="registration_data_id_attr"

return="smart-members/edit-profile"
enable_recaptcha="yes"
error_reporting="inline"
wrap_errors="<span class='error-inline'>|</span>"

on_submit="registration()"
}
<p>
<input type="text" name="CUSTOM_FIELD" placeholder="CU
{error:CUSTOM_FIELD}
</p>
```

```
{data_group_id}
{if data_group_id:count == 1}
<p>
<select name="role_id">
<option value=""></option>
{/if}
<option value="{group_id_value}">{group_id_label}</opt
{if data_group_id:count == data_group_id:total_results
</select>
{error:role_id}
</p>
{/if}
{/data_group_id}
<p>
<input type="text" name="username" placeholder="Userna
{error:username}
</p>
<p>
<input type="text" name="email" placeholder="Email Add
{error:email}
</p>

<p>
<input type="password" name="password" placeholder="Pa
{error:password}
</p>
<p>
```

```
<input type="password" name="password_confirm">
{error:password_confirm}
</p>


<p>
<input type="file" name="avatar_filename" />
{error:avatar_filename}
</p>
{if captcha}
<p>
<label for="captcha">Please enter in the word you see
{captcha}
<input type="text" name="captcha" id="captcha" />
{error:captcha}
</p>
{if:elseif recaptcha}
<p style="margin: 0;">
<label for="recaptcha">Click the checkbox</label>
{recaptcha}
{error:recaptcha}
</p>
{/if}
<p>
<input type="checkbox" name="accept_terms" value="y" /
{error:accept_terms}
</p>
<div class="register-register-click cf">
```

```
    <input type="submit" class="register" value="Register'
    </div>
    {/exp:smart_members:register}
```

Login

Login module is use for login the registered member. ExpressionEngine also provides default Login module. We are giving the Login module to provide the additional features the addon provides such as Inline error reporting, Recaptcha, Login with either Username or Email address etc.

Tag for Login module will look like this.

```
{exp:smart_members:login} Content data {/exp:smart_membe
```

Parameters

1. allowed_groups
2. rule:FIELD_NAME
3. attr:ATTRIBUTES
4. return
5. error_reporting
6. wrap_errors
7. on_submit
8. secure_action
9. secure_return

10. enable_recaptcha

Fields

1. group_id
2. username
3. email
4. password
5. auto_login
6. captcha

Following Parameters can be use in Login form

allowed_groups

Allowed groups parameter is used to give options to member for select any group from them. Use input field of group_id to enter the group id from defined allowed groups.
For EE6, you will pass the primary role ID to this parameter.

Example:

```
allowed_groups = "5|6|7"
```

rule:FIELD_NAME

Rule parameter is use to give custom rules to fields separate by pipe (|).

Example:

```
rule:username   = "required"
rule:email      = "required|valid_email|is_unique[members
```

attr:ATTRIBUTES

Parameter to add attributes in form. We can add classes, ids etc.

Example:

```
attr:id = "form_id"

attr:class = "form_class"

attr:name = "form_name"

attr:data-id = "form_data_id"
```

return

Return to any specific page after successful submission of form.

Example:

```
return = "smart-members/profile"
```

error_reporting

Error reporting format is defined by this parameter. It can be either “inline” or “outline”.

“Inline” error reporting will show the error in same page.
“outline” error reporting will show error in EE gray box in new page.

Example:

```
error_reporting="inline"
```

wrap_errors

Use this parameter to wrap forms error in any span or div if set error_reporting=”inline”.

Example:

```
wrap_errors="<span class='error-inline'>|</span>"
```

on_submit

This parameter allows us to call any Javascript function on submit of form

Example:

```
on_submit="call_me( )"
```

secure_action

Secure action will post the data on secure site i.e., https

Example:

```
secure_action="yes"
```

secure_return

Secure return will return the page after submit of form on secure site i.e., https

Example:

```
secure_return="yes"
```

enable_recaptcha

This parameter enables recaptcha instead of normal captcha.

1. Scenario EE2:
    1. If this parameter is set and API key and SECRET is not passed for recaptcha in backend, The normal captcha will show.
    2. If recaptcha API key and SECRET is passed in backend and this parameter not set:
        1. If the page is registration page and you have set captcha as required from member preferences, normal captcha will show.
        2. If the page is not registration page, neither captcha or recaptcha will show.
    3. If recaptcha API key and SECRET is passed in backend and this parameter is set:
        1. If the page is registration page and you have set captcha as required from member preferences, The normal captcha will not show and recaptcha will override the settings.
        2. If the page is not registration page, recaptcha will show.

2. Scenario EE3:
    1. Same scenario like EE2. Only change is, there is backend member preference settings in EE3 that allows to not enter any captcha if member is logged in. So if you set this parameter and API key and SECRET is also passed, If you are logged in this recaptcha or captcha will not show until you set “Require CAPTCHA while logged in?” to “Yes” from backend member settings > captcha settings.

Example:

```
enable_recaptcha="yes"
```

Different methods that can be use by user are given below.

Input

```
<input type="email" name="username" placeholder="Email A
{error:username} //if error_reporting="inline"
```

Select Dropdown: (For member group)

Notes: You will use the below script for EE6.

```
{data_group_id}
{if data_group_id:count == 1}
<select name="group_id">
    <option value=""></option>
    {/if}
    <option value = "{group_id_value}" > {group_id_label
    {if data_group_id:count == data_group_id:total_resul
</select>
{error:group_id} //if error_reporting="inline"
{/if}
{/data_group_id}
```

Captcha

```
{if captcha}
<p>
    <label for="captcha">Please enter in the word you se
    {captcha}
    <input type="text" name="captcha" id="captcha" />
    {error:captcha}
</p>
{if:elseif recaptcha}
<p style="margin: 0;">
    <label for="recaptcha">Click the checkbox</label>{re
    {error:recaptcha}
</p>
{/if}
```

Auto Login

```
<input type="checkbox" name="auto_login" value="y" /> Re
{error:auto_login} //if error_reporting="inline"
```

Example:

```
{exp:smart_members:login
    rule:username="required|min_length[5]"
    rule:password="required"

    allowed_groups="1|5|7"

    attr:id="login_id"
    attr:class="login_class"
    attr:name="login-form"
    attr:data-id="login_datta_id_attr"

    return="/smart-members/edit-profile"

    error_reporting="inline"
    wrap_errors="<span class='error-inline'>|</span>"

    enable_recaptcha="yes"

    on_submit="login()"
}
    {data_group_id}
    {if data_group_id:count == 1}
    <p>
        <select name="group_id">
            <option value=""></option>
            {/if}
            <option value="{group_id_value}">{group_id_l
            {if data_group_id:count == data_group_id:tot
        </select>
        {error:group_id}
    </p>
    {/if}
    {/data_group_id}
```

```
<p>
    <input type="text" name="username" class="text r
    {error:username}
</p>
<p>
    <input type="password" name="password" class="pa
    {error:password}
</p>
{if captcha}
<p>
    <label for="captcha">Please enter in the word yo
    {captcha}
    <input type="text" name="captcha" id="captcha" /
    {error:captcha}
</p>
{if:elseif recaptcha}
<p style="margin: 0;">
    <label for="recaptcha">Click the checkbox to com
    {recaptcha}
    {error:recaptcha}
</p>
{/if}
<label for="auto_login">
    <input type="checkbox" name="auto_login" id="aut
    {error:auto_login}
</label>
```

```
    <div class="login-login-click cf">
        <input type="submit" class="login-popup-btn logi
    </div>
{/exp:smart_members:login}
```

Social Login

Social Login module is use for login/register a member with your site. A Social login module will react as per settings you have made in backend.

Tag for Social Login module will look like this.

```
{exp:smart_members:social_login} Content data {/exp:smar
```

Parameters

1. rule:FIELD_NAME
2. attr:ATTRIBUTES
3. return
4. no_email_return
5. error_reporting
6. wrap_errors
7. on_submit
8. secure_action
9. secure_return
10. enable_recaptcha
11. popup

12. remember_me

13. providers

Fields

1. providers

Following Parameters can be use in Social Login form

rule:FIELD_NAME

Rule parameter is use to give custom rules to fields separate by pipe (|).

Example:

```
rule:username  = "required"
rule:email     = "required|valid_email|is_unique[members
```

attr:ATTRIBUTES

Parameter to add attributes in form. We can add classes, ids etc.

Example:

```
attr:id = "form_id"
attr:class = "form_class"
attr:name = "form_name"
attr:data-id = "form_data_id"
```

return

Return to any specific page after successful submission of form.

Example:

```
return = "smart-members/profile"
```

no_email_return

Return to a decided page if Social API not return email address of user.

Example:

```
no_email_return = "smart-members/member-form"
```

error_reporting

Error reporting format is defined by this parameter. It can be either “inline” or “outline”.

“Inline” error reporting will show the error in same page.
“outline” error reporting will show error in EE gray box in new page.

Example:

```
error_reporting="inline"
```

wrap_errors

Use this parameter to wrap forms error in any span or div if set error_reporting=”inline”.

Example:

```
wrap_errors="<span class='error-inline'>|</span>"
```

on_submit

This parameter allows us to call any Javascript function on submit of form

Example:

```
on_submit="call_me( )"
```

secure_action

Secure action will post the data on secure site i.e., https

Example:

```
secure_action="yes"
```

secure_return

Secure return will return the page after submit of form on secure site i.e., https

Example:

```
secure_return="yes"
```

enable_recaptcha

This parameter enables recaptcha instead of normal captcha.

1. Scenario EE2:
    1. If this parameter is set and API key and SECRET is not passed for recaptcha in backend, The normal captcha will show.
    2. If recaptcha API key and SECRET is passed in backend and this parameter not set:
        1. If the page is registration page and you have set captcha as required from member preferences, normal captcha will show.
        2. If the page is not registration page, neither captcha or recaptcha will show.
    3. If recaptcha API key and SECRET is passed in backend and this parameter is set:
        1. If the page is registration page and you have set captcha as required from member preferences, The normal captcha will not show and recaptcha will override the settings.
        2. If the page is not registration page, recaptcha will show.
2. Scenario EE3:
    1. Same scenario like EE2. Only change is, there is backend member preference settings in EE3 that allows to not enter any captcha if member is logged in. So if you set this parameter and API key and SECRET is also passed, If you are logged in this recaptcha or captcha will not show until you set “Require CAPTCHA while logged in?” to “Yes” from backend member settings > captcha settings.

Example:

```
enable_recaptcha="yes"
```

popup

Popup parameter will decide your request will sent to social API from same page or send in popup

Example:

```
popup="yes"
```

remember_me

remember_me parameter will allow user to save session for long to remember the user so user can avoid login again and again.

Example:

```
remember_me="yes"
```

providers

providers parameter will decide the provider you want to show in providers list to allow user to login with. You can ignore the paramter if you want to let allow all the active providers to allow to login with.

Example:

```
providers = "facebook|twitter|google"
```

Different methods that can be use by user are given below.

Select Dropdown (For Providers listing)

```
{providers}

   {if providers:count == 1}

   <p>

       <select name="providers">

       <option value=""></option>
```

```
        {/if}
        <option value="{provider_name}">{provider_label}<
        {if providers:count == providers:total_results}
        </select>
        {error:providers}
    </p>
    {/if}
{/providers}
```

Captcha

```
{if captcha}
<p>
    <label for="captcha">Please enter in the word you se
    {captcha}
    <input type="text" name="captcha" id="captcha" />
    {error:captcha}
</p>
{if:elseif recaptcha}
<p style="margin: 0;">
    <label for="recaptcha">Click the checkbox</label>{re
    {error:recaptcha}
</p>
{/if}
```

Example:

```
{exp:smart_members:social_login
    providers = "facebook|twitter|google"
    attr:id = "sl_id"
    attr:class = "sl_class"
    attr:name = "sl-form"
    attr:data-id = "sl_datta_id_attr"

    rule:providers = "required"

    error_reporting="inline"
    wrap_errors="<span class='error-inline'>|</span>"

    on_submit = "sl()"

    return = "smart-members/profile"
    no_email_return = "smart-members/edit-profile"

    remember_me = "yes"
    secure_action="yes"
    secure_return="yes"

    enable_recaptcha = "yes"
    popup = "yes"
}
        {providers}
            {if providers:count == 1}
            <p>
                <select name="providers">
```

```
                    <option value=""></option>
                    {/if}
                    <option value="{provider_name}">{prov
                {if providers:count == providers:total_re
                </select>
                {error:providers}
            </p>
            {/if}
        {/providers}

        <div class="login-register-click cf">
            <input type="submit" class="login-popup-btn r
        </div>
{/exp:smart_members:social_login}
```

Edit profile

Edit profile form is used to update the member static data as well as custom data. A member can also edit their profile username, email and password with this form.

Normal static fields and custom fields doesn't require current password to update the data.

Dependent fields for login and email such as username, email and password fields needs current password to be entered to update the fields.

Tag for Edit profile module will look like this.

```
{exp:smart_members:edit} Content data {/exp:smart_member
```

Parameters

1. allowed_groups
2. rule:FIELD_NAME
3. attr:ATTRIBUTES
4. return
5. error_reporting
6. wrap_errors
7. on_submit
8. secure_action
9. secure_return
10. enable_recaptcha
11. member_id
12. allowed_admin_groups

Fields

1. group_id (For <= EE5)
2. role_id (For EE6)
3. username
4. email
5. password
6. password_confirm
7. avatar_filename
8. photo_filename
9. sig_img_filename

10. yahoo_im
11. url
12. location
13. occupation
14. interests
15. aol_im
16. msn_im
17. icq
18. bio
19. signature
20. captcha
21. CUSTOM_FIELD
22. CUSTOM_CHECKBOX_FIELD (Pro Feature)
23. CUSTOM_RADIO_FIELD (Pro Feature)
24. CUSTOM_MULTI_SELECT_FIELD (Pro Feature)
25. CUSTOM_FILE_FIELD (Pro Feature)

Following Parameters can be use in Edit Profile form

allowed_groups

Allowed groups parameter is used to give options to member for select any group from them. Use input field of group_id to enter the group id from defined allowed groups.

For EE6, you will pass the primary role ID to this parameter.

Example:

```
allowed_groups = "5|6|7"
```

rule:FIELD_NAME

Rule parameter is use to give custom rules to fields separate by pipe (|).

Example:

```
rule:username  = "required"
rule:email     = "required|valid_email|is_unique[members
```

attr:ATTRIBUTES

Parameter to add attributes in form. We can add classes, ids etc.

Example:

```
attr:id = "form_id"
attr:class = "form_class"
attr:name = "form_name"
attr:data-id = "form_data_id"
```

return

Return to any specific page after successful submission of form.

Example:

```
return = "smart-members/profile"
```

error_reporting

Error reporting format is defined by this parameter. It can be either “inline” or “outline”.

“Inline” error reporting will show the error in same page.
“outline” error reporting will show error in EE gray box in new page.

Example:

```
error_reporting="inline"
```

wrap_errors

Use this parameter to wrap forms error in any span or div if set error_reporting=”inline”.

Example:

```
wrap_errors="<span class='error-inline'>|</span>"
```

on_submit

This parameter allows us to call any Javascript function on submit of form

Example:

```
on_submit="call_me( )"
```

secure_action

Secure action will post the data on secure site i.e., https

Example:

```
secure_action="yes"
```

secure_return

Secure return will return the page after submit of form on secure site i.e., https

Example:

```
secure_return="yes"
```

enable_recaptcha

This parameter enables recaptcha instead of normal captcha.

1. Scenario EE2:
    1. If this parameter is set and API key and SECRET is not passed for recaptcha in backend, The normal captcha will show.
    2. If recaptcha API key and SECRET is passed in backend and this parameter not set:
        1. If the page is registration page and you have set captcha as required from member preferences, normal captcha will show.

        2. If the page is not registration page, neither captcha or recaptcha will show.
    3. If recaptcha API key and SECRET is passed in backend and this parameter is set:
        1. If the page is registration page and you have set captcha as required from member preferences, The normal captcha will not show and recaptcha will override the settings.
        2. If the page is not registration page, recaptcha will show.
2. Scenario EE3:
    1. Same scenario like EE2. Only change is, there is backend member preference settings in EE3 that allows to not enter any captcha if member is logged in. So if you set this parameter and API key and SECRET is also passed, If you are logged in this recaptcha or captcha will not show until you set “Require CAPTCHA while logged in?” to “Yes” from backend member settings > captcha settings.

Example:

```
enable_recaptcha="yes"
```

member_id

Use this parameter if you want to edit some other member’s information instead of current logged in member. You can pass member_id of member whose data you want to modify and can modify just like you edit your own profile.

If you want to edit username, email or password, You need to have a field name current_password. You need to enter password of current logged in member in this field.

Example:

```
member_id="25"
```

```
member_id="{segment_3}"
```

allowed_admin_groups

This parameter play main role if you want to modify member data of other users. You can restrict modifiers by their group. So if you want to have only super admin, site admins and modifiers to edit information of other members, Just pass their group_id in this parameter and they will be able to edit all information for particular member.

For EE6, you will pass the primary role ID to this parameter.

Note: That can be a major risk if you give any random people to change data of other members. It is requested to use this functionality wisely.

Example:

```
allowed_admin_groups="1"
```

```
allowed_admin_groups="1|6|7"
```

Different methods that can be use by user are given below.

Input

```
<input type="email" name="username" placeholder="Email A
{error:username} //if error_reporting="inline"
```

Textarea

```
<textarea name="signature"></textarea>
{error:signature} //if error_reporting="inline"
```

File field

```
<input type="file" name="avatar_filename" />
{error:avatar_filename} //if error_reporting="inline"
```

Select Dropdown: (For member group) (For <= EE5)

```
{data_group_id}
{if data_group_id:count == 1}
<select name="group_id">
    <option value=""></option>
    {/if}
    <option value = "{group_id_value}" {if group_id == g
    {if data_group_id:count == data_group_id:total_resul
</select>
{error:group_id} //if error_reporting="inline"
{/if}
{/data_group_id}
```

Select Dropdown: (For member primary role) (For EE6)

```
  {data_group_id}
  {if data_group_id:count == 1}
  <select name="role_id">
  <option value=""></option>
  {/if}
  <option value = "{group_id_value}" {if group_id == gro
  {if data_group_id:count == data_group_id:total_results
  </select>
  {error:role_id} //if error_reporting="inline"
  {/if}
  {/data_group_id}
```

Select Dropdown: (For custom field)

```
{data_CUSTOM_FIELD}
{if data_CUSTOM_FIELD:count == 1}
<select name="CUSTOM_FIELD">
<option value=""></option>
    {/if}
    <option value = "{CUSTOM_FIELD_value}"  {if CUSTOM_F
    {if data_CUSTOM_FIELD:count == data_CUSTOM_FIELD:tot
</select>
{error:CUSTOM_FIELD} //if error_reporting="inline"
{/if}
{/data_CUSTOM_FIELD}
```

Captcha

```
{if captcha}
<p>
    <label for="captcha">Please enter in the word you se
    {captcha}
    <input type="text" name="captcha" id="captcha" />
    {error:captcha}
</p>
{if:elseif recaptcha}
<p style="margin: 0;">
    <label for="recaptcha">Click the checkbox</label>{re
    {error:recaptcha}
</p>
{/if}
```

CUSTOM_CHECKBOX_FIELD (Pro Feature)

```
{data_CUSTOM_CHECKBOX_FIELD}
    {if data_CUSTOM_CHECKBOX_FIELD:count == 1}
    <p style="display: inline-flex; ">
        {/if}
        <input type="checkbox" name="CUSTOM_CHECKBOX_FIE
        <label for="test_{data_CUSTOM_CHECKBOX_FIELD:cou
        {if data_CUSTOM_CHECKBOX_FIELD:count == data_CUS
        {error:CUSTOM_CHECKBOX_FIELD}
    </p>
    {/if}
{/data_CUSTOM_CHECKBOX_FIELD}
```

CUSTOM_RADIO_FIELD (Pro Feature)

```
{data_CUSTOM_RADIO_FIELD}
    {if data_CUSTOM_RADIO_FIELD:count == 1}
    <p style="display: inline-flex;">
        {/if}
        <input type="radio" name="CUSTOM_RADIO_FIELD" va
        <label for="test_{data_CUSTOM_RADIO_FIELD:count}
        {if data_CUSTOM_RADIO_FIELD:count == data_CUSTOM
        {error:CUSTOM_RADIO_FIELD}
    </p>
    {/if}
{/data_CUSTOM_RADIO_FIELD}
```

CUSTOM_MULTI_SELECT_FIELD (Pro Feature)

```
{data_CUSTOM_MULTI_SELECT_FIELD}
{if data_CUSTOM_MULTI_SELECT_FIELD:count == 1}
<p style="display: inline-flex; ">
    <select multiple name="CUSTOM_MULTI_SELECT_FIELD[]">
        {/if}
        <option value="{CUSTOM_MULTI_SELECT_FIELD_value}
        {if data_CUSTOM_MULTI_SELECT_FIELD:count == data
    </select>
    {error:CUSTOM_MULTI_SELECT_FIELD}
</p>
{/if}
{/data_CUSTOM_MULTI_SELECT_FIELD}
```

CUSTOM_FILE_FIELD (Pro Feature)

```
<p>
    Old file name: {CUSTOM_FILE_FIELD}<br>
    <input type="file" name="CUSTOM_FILE_FIELD">
    {error:CUSTOM_FILE_FIELD}
</p>
```

Example (For <= EE5):

```
{exp:smart_members:edit
    allowed_groups="6|7|5"

    rule:username="required|valid_email|min_length[5]"
    rule:password="required|matches[password_confirm]|mi
    rule:password_confirm="required|min_length[5]"

    rule:group_id="required"
    rule:state="required"
    rule:first_name="required"

    attr:id="edit_profile_id"
    attr:class="edit_profile_class"
    attr:name="edit_profile-form"
    attr:data-id="edit_profile_data_id_attr"

    return="smart-members/edit-profile"
    enable_recaptcha="yes"
    error_reporting="inline"
    wrap_errors="<span class='error-inline'>|</span>"
    on_submit="edit_profile()"
}
    <p>
        <input type="text" name="CUSTOM_FIELD" placehold
        {error:CUSTOM_FIELD}
    </p>

    {data_group_id}
    {if data_group_id:count == 1}
    <p>
        <select name="group_id">
            <option value=""></option>
            {/if}
            <option value="{group_id_value}" {if group_i
```

```
                {group_id_label}
            </option>
            {if data_group_id:count == data_group_id:tot
        </select>
        {error:group_id}
    </p>
    {/if}
    {/data_group_id}
    <p>
        <input type="text" name="username" value="{usern
        {error:username}
    </p>
    <p>
        <input type="text" name="email" value="{email}">
        {error:email}
    </p>
    <p>
        <input type="password" name="current_password">
        {error:current_password}
    </p>
    <p>
        <input type="password" name="password">
        {error:password}
    </p>
    <p>
        <input type="password" name="password_confirm">
        {error:password_confirm}
    </p>
    <p>
        {if avatar_filename}
        <img src="{avatar_url}{avatar_filename}" height=
        <input type="file" name="avatar_filename" />
        {error:avatar_filename}
    </p>
    {if captcha}
    <p>
        <label for="captcha">Please enter in the word yo
        {captcha}
        <input type="text" name="captcha" id="captcha" /
        {error:captcha}
    </p>
    {if:elseif recaptcha}
    <p style="margin: 0;"><label for="recaptcha">Click t
        {recaptcha}
        {error:recaptcha}
    </p>
    {/if}
    <div class="edit-profile-click cf">
        <input type="submit" class="edit" value="Update 
    </div>
{/exp:smart_members:edit}
```

Example (For EE6):

```
{exp:smart_members:edit
    allowed_groups="6|7|5"

    rule:username="required|valid_email|min_length[5]"
    rule:password="required|matches[password_confirm]|mi
    rule:password_confirm="required|min_length[5]"
    rule:role_id="required"
    rule:state="required"
    rule:first_name="required"

    attr:id="edit_profile_id"
    attr:class="edit_profile_class"
    attr:name="edit_profile-form"
    attr:data-id="edit_profile_data_id_attr"

    return="smart-members/edit-profile"
    enable_recaptcha="yes"
    error_reporting="inline"
    wrap_errors="<span class='error-inline'>|</span>"
    on_submit="edit_profile()"
}
    <p>
        <input type="text" name="CUSTOM_FIELD" placehold
        {error:CUSTOM_FIELD}
    </p>

    {data_group_id}
```

```
{if data_group_id:count == 1}
<p>
    <select name="role_id">
        <option value=""></option>
        {/if}
        <option value="{group_id_value}" {if group_i
           {group_id_label}
        </option>
        {if data_group_id:count == data_group_id:tot
    </select>
    {error:role_id}
</p>
{/if}
{/data_group_id}
<p>
    <input type="text" name="username" value="{usern
    {error:username}
</p>
<p>
    <input type="text" name="email" value="{email}">
    {error:email}
</p>
<p>
    <input type="password" name="current_password">
    {error:current_password}
</p>
<p>
```

```
        <input type="password" name="password">
        {error:password}
    </p>
    <p>
        <input type="password" name="password_confirm">
        {error:password_confirm}
    </p>
    <p>
        {if avatar_filename}
        <img src="{avatar_url}{avatar_filename}" height=
        <input type="file" name="avatar_filename" />
        {error:avatar_filename}
    </p>
    {if captcha}
    <p>
        <label for="captcha">Please enter in the word yo
        {captcha}
        <input type="text" name="captcha" id="captcha" /
        {error:captcha}
    </p>
    {if:elseif recaptcha}
    <p style="margin: 0;"><label for="recaptcha">Click t
        {recaptcha}
        {error:recaptcha}
    </p>
    {/if}
    <div class="edit-profile-click cf">
```

```
            <input type="submit" class="edit" value="Update
        </div>
{/exp:smart_members:edit}
```

Remove Image

Member can remove their images from "Edit member form". This section contains removing of default images i.e., "Avatar file, Photo fIle and Signature image file".

You can remove these images in 2 way.

Parameters

1. By submit the form:
2. By input field:

By submit the form

You need to pass an submit button named the file.

Examples:To remove Avatar file pass this "name" attribute in submit button: (You can pass anything in "value" according to your form.)

```
<input type="submit" name="remove_avatar" value="Remove
<input type="submit" name="remove_avatar" value="Submit"
<input type="submit" name="remove_avatar" value="Edit Pr
```

To remove Photo file pass this "name" attribute in submit button: (You can pass anything in "value" according to your form.)

```
<input type="submit" name="remove_photo" value="Remove P
<input type="submit" name="remove_photo" value="Submit"
```

```
<input type="submit" name="remove_photo" value="Edit Pro
```

To remove Signature file pass this “name” attribute in submit button: (You can pass anything in “value” according to your form.)

```
<input type="submit" name="remove_sig_img" value="Remove
<input type="submit" name="remove_sig_img" value="Submit
<input type="submit" name="remove_sig_img" value="Edit P
```

When you submit the form with this input type, it will remove the file when you submit the form (Only file entry will remove, Images will still there in image upload folder till user not upload another image.)

By input field

You can also pass a hidden value or radio button or javascript named with attribute remove name to remove that image.

Examples:

```
<input type='hidden' name='remove_avatar'>
<input type='hidden' name='remove_photo'>
<input type='hidden' name='remove_sig_img'>
```

or

```
<input type="radio" name="remove_avatar" value="remove"/
<input type="radio" name="remove_photo"  value="remove"/
<input type="radio" name="remove_sig_img" value="remove"
```

Below is the example of JQuery remove:

Example:

```
{exp:smart_members:edit
    return="smart-members/edit-profile"
    password_required="no"
    error_reporting="inline"
    wrap_errors="<span class='error-inline'>|</span>"
}
    <p>
        {if avatar_filename}
            <div>
                <img src="{avatar_url}{avatar_filename}"
                <a href="javascript:void(0);" class="remo
                    Remove Avatar
                </a><br>
            </div>
        {/if}
        <input type="file" name="avatar_filename" />
        {error:avatar_filename}
    </p>
    <p>
        {if photo_filename}
            <div>
                <img src="{photo_url}{photo_filename}" he
                <a href="javascript:void(0);" class="remo
                    Remove Photo
                </a><br>
            </div>
        {/if}
        <input type="file" name="photo_filename" />
        {error:photo_filename}
    </p>
    <p>
        {if sig_img_filename}
            <div>
                <img src="{sig_img_url}{sig_img_filename}
                <a href="javascript:void(0);" class="remo
                    Remove Signature
                </a><br>
            </div>
        {/if}
        <input type="file" name="sig_img_filename" />
        {error:sig_img_filename}
    </p>

    <div class="login-edit-click cf">
        <input type="submit" value="Update Account Inform
    </div>
{/exp:smart_members:edit}
```

JQuery:

```
<script type="text/javascript">
$(document).ready(function() {
    $(document).on('click', '.remove_sm_pic', function(ev
        event.preventDefault();
        /* Act on the event */
        if(typeof($(this).attr('add_field_attr')) !== "un
        {
            if($(this).parent('div').find("input[name="+
            {
                $(this).parent('div').hide()
                $(this).parent('div').append("<input type
            }
        }
        else
        {
            console.log('Cannnot remove')
        }
    });
});
</script>
```

View Profile

View profile module is to list the member(s) data. You can access each and every data except password with this module.

Tag for View profile module will look like this.

```
{exp:smart_members:profile} Content data {/exp:smart_mem
```

Parameters

1. member_id
2. group_id
3. not_member_id
4. not_group_id
5. limit
6. order_by
7. sort

Fields

1. aol_im
2. avatar_url
3. avatar_filename
4. avatar_height
5. avatar_width
6. bday_d
7. bday_m
8. bday_y
9. bio
10. email
11. group_id (For <= EE5)
12. role_id (For EE6)
13. icq
14. interests
15. join_date
16. last_activity

17. last_bulletin_date
18. last_comment_date
19. last_entry_date
20. last_email_date
21. last_forum_post_date
22. last_view_bulletins
23. last_visit
24. location
25. member_id
26. msn_im
27. notepad
28. notepad_size
29. occupation
30. private_messages
31. photo_url
32. photo_filename
33. photo_height
34. photo_width
35. screen_name
36. signature
37. sig_img_url
38. sig_img_filename
39. sig_img_width
40. sig_img_height
41. total_comments

42. total_entries
43. total_forum_topics
44. total_forum_posts
45. username
46. url
47. yahoo_im
48. CUSTOM_FIELD

Labels

1. aol_im_label
2. avatar_filename_label
3. avatar_height_label
4. avatar_width_label
5. bday_d_label
6. bday_m_label
7. bday_y_label
8. bio_label
9. email_label
10. group_id_label (For <= EE5)
11. role_id_label (For EE6)
12. icq_label
13. interests_label
14. join_date_label
15. last_activity_label
16. last_bulletin_date_label

17. last_comment_date_label
18. last_entry_date_label
19. last_email_date_label
20. last_forum_post_date_label
21. last_view_bulletins_label
22. last_visit_label
23. location_label
24. member_id_label
25. msn_im_label
26. notepad_label
27. notepad_size_label
28. occupation_label
29. private_messages_label
30. photo_filename_label
31. photo_height_label
32. photo_width_label
33. screen_name_label
34. signature_label
35. sig_img_filename_label
36. sig_img_width_label
37. sig_img_height_label
38. total_comments_label
39. total_entries_label
40. total_forum_topics_label
41. total_forum_posts_label

42. username_label
43. url_label
44. yahoo_im_label
45. CUSTOM_FIELD_label

To list all the fields above there is single field. It will list all the static fields which is not empty as well as all the custom fields created by user.

1. sm_list_all_fields
    1. Parameters (It doesn't contain any parameter)
    2. Fields
        1. field_label
        2. field_value
        3. field_sort_name
        4. field_db_name
        5. sm_list_all_fields:count
        6. sm_list_all_fields:total_results

Example:

```
{sm_list_all_fields}
        {field_label} : {field_value}
{/sm_list_all_fields}
```

Following Parameters can be use in View Profile page

member_id

Member ID of user you want to get the data of.
Not to use this parameter, Leave this parameter blank or pass "CURRENT_MEMBER" in parameter will extract profile data of current member.
Pass "ALL_MEMBERS" in parameter if you want to fetch all the members.

Example:

```
member_id = ""
member_id = "CURRENT_MEMBER"
member_id = "ALL_MEMBERS"
member_id = "53"
member_id = "50|51|52|53"
```

group_id

Member(s) of particular group you want to get data of.
Using this parameter will filter the output data with member group. If member group doesn't exists in the groups passed in parameter, it will not show the data of member.
Passing nothing in this parameter will bring every groups except "banned (2)", "Guests (3)" and "Pending (4)".

For EE6, you can use this same parameter to pass the member primary role ID. You can also use the 'role_id' parameter instead of the 'group_id' parameter if you want it for the EE6.

Example:

```
group_id = "5"
group_id = "5|6|7"
```

not_member_id

You can pass the ID of the member(s) with pipe (|) separated to ignore record of that member.

For EE6, you will pass the primary role ID to this parameter.

Example:

```
not_member_id = "2"
not_member_id = "2|4"
```

not_group_id

You can pass the ID of the group with pipe (|) separated to ignore record of that group.

For EE6, you will pass the primary role ID to this parameter.

Example:

```
not_group_id = "5"

not_group_id = "5|6"
```

limit

You can Limit output rows by any value you want.

Example:

```
limit = "50"
```

order_by

This parameter use to start sorting via any proper field. You can use both default member field or custom member field to group by the result.

Example:

```
order_by = "member_id"

order_by = "first_name"
```

sort

You can sort the output by ASC or DESC.

Example:

```
sort = "asc"

sort = "desc"
```

How to render custome member fields

We have developed custem member field types like checkbox, dropdown, radio button, file, etc user can render this custom member fields with some tags like below

If user want to render checkbox and multi-select dropdown then use below code.

Example:

```
{CUSTM_MEMBER_FIELD_SHORT_NAME}
        {CUSTM_MEMBER_FIELD_SHORT_NAME:label} : {CUSTM_ME
{CUSTM_MEMBER_FIELD_SHORT_NAME}
```

If user want to render other then checkbox and multi-select dropdown (means single value custom field) then use below code.

Example:

```
{CUSTM_MEMBER_FIELD_SHORT_NAME}
```

OR

```
{CUSTM_MEMBER_FIELD_SHORT_NAME:label} : {CUSTM_MEMBER_FIE
```

Example (For <= EE5):

```
 {exp:smart_members:profile
     member_id="ALL_MEMBERS"

     not_group_id="1|6"
     not_member_id="14|1"
 }
     {if no_results} <p>no data found!!</p> {/if}

     <p>Your details of member id {member_id} are as foll

     <p>All Fields:</p>
```

```
    <ul>
        {sm_list_all_fields}
        <li><b>{field_label} : </b> {field_value}</li>
        {/sm_list_all_fields}
    </ul>

    <p>Fields of interest:</p>
    <ul>
        <li> <b> {group_id_label}        : </b> {group_id
        <li> <b> {username_label}        : </b> {username
        <li> <b> {screen_name_label}     : </b> {screen_n
        <li> <b> {email_label}           : </b> {email}
    </ul>

{/exp:smart_members:profile}
```

Example (For EE6):

```
{exp:smart_members:profile
    member_id="ALL_MEMBERS"

    not_group_id="1|6"
    not_member_id="14|1"
}
    {if no_results} <p>no data found!!</p> {/if}

    <p>Your details of member id {member_id} are as foll
    <p>All Fields:</p>
    <ul>
        {sm_list_all_fields}
        <li><b>{field_label} : </b> {field_value}</li>
        {/sm_list_all_fields}
    </ul>

    <p>Fields of interest:</p>
    <ul>
        <li> <b> {role_id_label}       : </b> {role_id}
        <li> <b> {username_label}       : </b> {username
        <li> <b> {screen_name_label}    : </b> {screen_n
        <li> <b> {email_label}          : </b> {email}
    </ul>

{/exp:smart_members:profile}
```

Forgot Password

Forgot Password module is use to send URL of reset password with key to reset the password of user.
Parameter of reset password template as well as forgot password email settings can be sent from backend smart members setting or can pass in this tag as parameter.

Tag for View profile module will look like this.

```
{exp:smart_members:forgot_password}
        ... Content data ...
{/exp:smart_members:forgot_password}
```

Parameters

1. rule:FIELD_NAME
2. attr:ATTRIBUTES
3. return
4. error_reporting
5. wrap_errors
6. on_submit
7. secure_action
8. secure_return
9. reset_password_template
10. email:subject
11. email:template
12. email:word_wrap
13. email:mailtype
14. enable_recaptcha

Fields

1. email
2. captcha

Following Parameters can be use in Forgot Password form

rule:FIELD_NAME

Rule parameter is use to give custom rules to fields separate by pipe (|).

Example:

```
rule:username   = "required"
rule:email      = "required|valid_email|is_unique[members
```

attr:ATTRIBUTES

Parameter to add attributes in form. We can add classes, ids etc.

Example:

```
attr:id = "form_id"
attr:class = "form_class"
attr:name = "form_name"
attr:data-id = "form_data_id"
```

return

Return to any specific page after successful submission of form.

Example:

```
return = "smart-members/profile"
```

error_reporting

Error reporting format is defined by this parameter. It can be either “inline” or “outline”.
“Inline” error reporting will show the error in same page.
“outline” error reporting will show error in EE gray box in new page.

Example:

```
error_reporting="inline"
```

wrap_errors

Use this parameter to wrap forms error in any span or div if set error_reporting=”inline”.
If error_reporting is set to inline and not defined this parameter, It will take span to display errors.

Example:

```
wrap_errors="<span class='error-inline'>|</span>"
```

on_submit

This parameter allows us to call any Javascript function on submit of form.

Example:

```
on_submit="call_me( )
```

secure_action

Secure action will post the data on secure site i.e., https.

Example:

```
secure_action="yes"
```

secure_return

Secure return will return the page after submit of form on secure site i.e., https.

Example:

```
secure_return="yes"
```

reset_password_template

Pass template path of reset password page. URL of this page will pass in forgot password email with reset token to user.
(Note: This can be passed from template as parameter or can set from backend member settings. If passed from backend don’t pass this parameter.)

Example:

```
reset_password_template="smart-members/reset-password"
```

email:subject

Pass forgot password email subject in this parameter. Pass the value in this parameter will show as a subject of email.
(Note: This can be passed from template as parameter or can set from backend member settings. If passed from backend don’t pass this parameter.)

Example:

```
email:subject = "Reset password request | {site_name}"
```

email:template

Pass forgot password email template path in this parameter. Template passed in this parameter will send as forgot password email body to member.
(Note: This can be passed from template as parameter or can set from backend member settings. If passed from backend don’t pass this parameter.)

Example:

```
email:template = "smart-members/email-forgot-password"
```

email:word_wrap

Pass the word wrap settings in this parameter.
(Note: This can be passed from template as parameter or can set from backend member settings. If passed from backend don’t pass this parameter.)

Example:

```
email:word_wrap = "yes"
```

email:mailtype

Pass email type in this parameter. Choosing right parameter is very important. If we choose “html”, the normal text will execute in same line and it will not look proper in email. Same as if we choose “text” as parameter, It will not consider HTML tags such as <p> <b> etc. and the tags will appear in mail body.
(Note: This can be passed from template as parameter or can set from backend member settings. If passed from backend don’t pass this parameter.)

Example:

```
email:mailtype = "text"

email:mailtype = "html"
```

enable_recaptcha

This parameter enables recaptcha instead of normal captcha.

1. Scenario EE2:
    1. If this parameter is set and API key and SECRET is not passed for recaptcha in backend, The normal captcha will show.
    2. If recaptcha API key and SECRET is passed in backend and this parameter not set:
        1. If the page is registration page and you have set captcha as required from member preferences, normal captcha will show.
        2. If the page is not registration page, neither captcha or recaptcha will show.
    3. If recaptcha API key and SECRET is passed in backend and this parameter is set:
        1. If the page is registration page and you have set captcha as required from member preferences, The normal captcha will not show and recaptcha will override the settings.
        2. If the page is not registration page, recaptcha will show.
2. Scenario EE3:
    1. Same scenario like EE2. Only change is, there is backend member preference settings in EE3 that allows to not enter any captcha if member is logged in. So if you set this parameter and API key and SECRET is also passed, If you are logged in this recaptcha or captcha will not show until you set “Require CAPTCHA while logged in?” to “Yes” from backend member settings > captcha settings.

Example:

```
enable_recaptcha="yes"
```

Different methods that can be use by user are given below.

Input

```
<input type="email" name="username" placeholder="Email A
{error:username} //if error_reporting="inline"
```

Captcha

```
{if captcha}
<p>
    <label for="captcha">Please enter in the word you se
    {captcha}
    <input type="text" name="captcha" id="captcha" />
    {error:captcha}
</p>
{if:elseif recaptcha}
<p style="margin: 0;">
    <label for="recaptcha">Click the checkbox</label>{re
    {error:recaptcha}
</p>
{/if}
```

Example:

```
{exp:smart_members:forgot_password
    attr:id="fp_id"
    attr:class="fp_class"
    attr:name="fp-form"
    attr:data-id="fp_data_id_attr"

    return="smart-members/send-forg-mail"

    error_reporting="inline"
    wrap_errors="<span class='error-inline'>|</span>"

    on_submit="fp()"
    enable_recaptcha="yes"

    reset_password_template="smart-members/reset-passwor

    email:subject="Reset password request"
    email:template="smart-members/forgot-password-email-
    email:word_wrap="yes"
    email:mailtype="html"
}
    <p>
        <input type="email" name="email">
        {error:email}
    </p>

    {if captcha}
    <p>
        <label for="captcha">Please enter in the word yo
        {captcha}
        <input type="text" name="captcha" id="captcha" /
```

```
            {error:captcha}
        </p>
        {if:elseif recaptcha}
        <p style="margin: 0;">
            <label for="recaptcha">Click the checkbox</label
            {recaptcha}
            {error:recaptcha}
        </p>
        {/if}
        <p>
            <input type="submit" class="forgot-pass" value="
        </p>
{/exp:smart_members:forgot_password}
```

Reset Password

Reset Password module is use to set new password with help of token send by user in mail through forgot password email.

Tag for Reset password module will look like this.

```
{exp:smart_members:reset_password}
       ... Content data ...
{/exp:smart_members:reset_password}
```

Parameters

1. rule:FIELD_NAME

Fields

1. password
2. password_confirm
3. captcha

Following Parameters can be use in Forgot Password form

rule:FIELD_NAME

Rule parameter is use to give custom rules to fields separate by pipe (|).

Example:

```
rule:username   = "required"
rule:email      = "required|valid_email|is_unique[members
```

attr:ATTRIBUTES

Parameter to add attributes in form. We can add classes, ids etc.

Example:

```
attr:id = "form_id"

attr:class = "form_class"

attr:name = "form_name"

attr:data-id = "form_data_id"
```

return

Return to any specific page after successful submission of form.

Example:

```
return = "smart-members/profile"
```

error_reporting

Error reporting format is defined by this parameter. It can be either “inline” or “outline”.
“Inline” error reporting will show the error in same page.
“outline” error reporting will show error in EE gray box in new page.

Example:

```
error_reporting="inline"
```

wrap_errors

Use this parameter to wrap forms error in any span or div if set error_reporting=”inline”.
If error_reporting is set to inline and not defined this parameter, It will take span to display errors.

Example:

```
wrap_errors="<span class='error-inline'>|</span>"
```

on_submit

This parameter allows us to call any Javascript function on submit of form.

Example:

```
on_submit="call_me( )
```

secure_action

Secure action will post the data on secure site i.e., https.

Example:

```
secure_action="yes"
```

secure_return

Secure return will return the page after submit of form on secure site i.e., https.

Example:

```
secure_return="yes"
```

enable_recaptcha

This parameter enables recaptcha instead of normal captcha.

1. Scenario EE2:
    1. If this parameter is set and API key and SECRET is not passed for recaptcha in backend, The normal captcha will show.
    2. If recaptcha API key and SECRET is passed in backend and this parameter not set:
        1. If the page is registration page and you have set captcha as required from member preferences, normal captcha will show.
        2. If the page is not registration page, neither captcha or recaptcha will show.
    3. If recaptcha API key and SECRET is passed in backend and this parameter is set:
        1. If the page is registration page and you have set captcha as required from member preferences, The normal captcha will not show and recaptcha will override the settings.
        2. If the page is not registration page, recaptcha will show.

2. Scenario EE3:
    1. Same scenario like EE2. Only change is, there is backend member preference settings in EE3 that allows to not enter any captcha if member is logged in. So if you set this parameter and API key and SECRET is also passed, If you are logged in this recaptcha or captcha will not show until you set “Require CAPTCHA while logged in?” to “Yes” from backend member settings > captcha settings.

Example:

```
enable_recaptcha="yes"
```

reset_code

Reset code is the token sent in forgot password email to user. You can simply set it parameterize from URL segment.

Example:

```
reset_code = "{segment_3}"
```

Different methods that can be use by user are given below.

Input

```
<input type="email" name="password" placeholder="Passwor
{error:password} //if error_reporting="inline"
```

Captcha

```
{if captcha}
<p>
    <label for="captcha">Please enter in the word you se
    {captcha}
    <input type="text" name="captcha" id="captcha" />
    {error:captcha}
```

```
</p>
{if:elseif recaptcha}
<p style="margin: 0;">
    <label for="recaptcha">Click the checkbox</label>{re
    {error:recaptcha}
</p>
{/if}
```

Example:

```
{if segment_3 != "reset-success"}
       {exp:smart_members:reset_password
               reset_code="{segment_3}"
               rule:password="required|matches[password_
               rule:password_confirm="required|min_lengt

               attr:id="rp_id"
               attr:class="rp_class"
               attr:name="rp-form"
               attr:data-id="rp_data_id_attr"

               return="smart-members/reset-password/rese
               enable_recaptcha="yes"

               error_reporting="inline"
               wrap_errors="<span class='error-inline'>|
               on_submit="rp()"
       }
```

```
        {if no_results}<h4>The reset token provided is in
        {/if}
        <p>
                <input type="password" name="password">{e
        </p>

        <p>
                <input type="password" name="password_con
        </p>

        {if captcha}
        <p>
                <label for="captcha">Please enter in the 
                {captcha}
                <input type="text" name="captcha" id="cap
                {error:captcha}
        </p>
        {if:elseif recaptcha}
        <p style="margin: 0;">
                <label for="recaptcha">Click the checkbox
                {recaptcha}
                {error:recaptcha}
        </p>
        {/if}
        <p>
                <input type="submit" class="reset" value=
        </p>
        {/exp:smart_members:reset_password}

{if:else}
        <p> Password successfully updated. You can now <a
{/if}
```

Delete Member

Delete member module is use to allow member to delete himself from membership account of site. If one will delete his/her account, all the entries or data he/she has entered in the site will also deletes.

Any one can delete his/her account from site except super admin.

To delete the member user only needs his/her account password.

Tag for Delete member module will look like this.

```
{exp:smart_members:delete}
   ... Content data ...
{/exp:smart_members:delete}
```

Parameters

1. rule:FIELD_NAME
2. attr:ATTRIBUTES
3. return
4. error_reporting
5. wrap_errors

6. on_submit
7. secure_action
8. secure_return
9. enable_recaptcha

Fields

1. password
2. captcha

Following Parameters can be use in Forgot Password form

rule:FIELD_NAME

Rule parameter is use to give custom rules to fields separate by pipe (|).

Example:

```
rule:username  = "required"
rule:email     = "required|valid_email|is_unique[members
```

attr:ATTRIBUTES

Parameter to add attributes in form. We can add classes, ids etc.

Example:

```
attr:id = "form_id"
attr:class = "form_class"
attr:name = "form_name"
attr:data-id = "form_data_id"
```

return

Return to any specific page after successful submission of form.

Example:

```
return = "smart-members/profile"
```

error_reporting

Error reporting format is defined by this parameter. It can be either “inline” or “outline”.
“Inline” error reporting will show the error in same page.
“outline” error reporting will show error in EE gray box in new page.

Example:

```
error_reporting="inline"
```

wrap_errors

Use this parameter to wrap forms error in any span or div if set error_reporting=”inline”.
If error_reporting is set to inline and not defined this parameter, It will take span to display errors.

Example:

```
wrap_errors="<span class='error-inline'>|</span>"
```

on_submit

This parameter allows us to call any Javascript function on submit of form.

Example:

```
on_submit="call_me( )
```

secure_action

Secure action will post the data on secure site i.e., https.

Example:

```
secure_action="yes"
```

secure_return

Secure return will return the page after submit of form on secure site i.e., https.

Example:

```
secure_return="yes"
```

enable_recaptcha

This parameter enables recaptcha instead of normal captcha.

1. Scenario EE2:
    1. If this parameter is set and API key and SECRET is not passed for recaptcha in backend, The normal captcha will show.
    2. If recaptcha API key and SECRET is passed in backend and this parameter not set:
        1. If the page is registration page and you have set captcha as required from member preferences, normal captcha will show.
        2. If the page is not registration page, neither captcha or recaptcha will show.
    3. If recaptcha API key and SECRET is passed in backend and this parameter is set:
        1. If the page is registration page and you have set captcha as required from member preferences, The normal captcha will not show and recaptcha will override the settings.
        2. If the page is not registration page, recaptcha will show.
2. Scenario EE3:
    1. Same scenario like EE2. Only change is, there is backend member preference settings in EE3 that allows to not enter any captcha if member is logged in. So if you set this parameter and API key and SECRET is also passed, If you are logged in this recaptcha or captcha will not show until you set “Require CAPTCHA while logged in?” to “Yes” from backend member settings > captcha settings.

Example:

```
enable_recaptcha="yes"
```

Different methods that can be use by user are given below.

Input

```
<input type="password" name="password" placeholder="Pass
{error:password} //if error_reporting="inline"
```

Captcha

```
{if captcha}
<p>
    <label for="captcha">Please enter in the word you se
    {captcha}
    <input type="text" name="captcha" id="captcha" />
    {error:captcha}
</p>
{if:elseif recaptcha}
<p style="margin: 0;">
    <label for="recaptcha">Click the checkbox</label>{re
    {error:recaptcha}
</p>
{/if}
```

Example:

```
{exp:smart_members:delete
       attr:id="dm_id"
       attr:class="dm_class"
       attr:name="dm-form"
       attr:data-id="dm_data_id_attr"

```

```
        return="smart-members/index"

        error_reporting="inline"
        wrap_errors="<span class='error-inline'>|</span>"

        on_submit="dm()"
        enable_recaptcha="yes"
}
         <p>
                <input type="password" name="password" pl
                {error:password}
         </p>
         {if captcha}
         <p>
                <label for="captcha">Please enter in the
                {captcha}
                <input type="text" name="captcha" id="cap
                {error:captcha}
         </p>
         {if:elseif recaptcha}
         <p style="margin: 0;">
                <label for="recaptcha">Click the checkbox
                {recaptcha}
                {error:recaptcha}
         </p>
         {/if}
         <p>
                <input type="submit" class="delete" value
         </p>
{/exp:smart_members:delete}
```

Logout

Logout module provides a user to successfully destroy the session and logging out from the site.

Tag for logout created in 2 ways. One is with closing tag and another one is without closing tag.

With closing tag, content code will look something like this:

```
{exp:smart_members:logout return='smart-members/index'}
   <p> <a href="{url}">Logout</a> </p>
{/exp:smart_members:logout}
```

Without closing tag, content code will look like this:

```
<a href="{exp:smart_members:logout return='smart-members
```

Parameters

1. return
2. secure_return

Following Parameters can be use in Forgot Password form

return

Return to any specific page after successful submission of form.

Example:

```
return = "smart-members/profile"
```

secure_return

Secure return will return the page after submit of form on secure site i.e., https.

Example:

```
secure_return="yes"
```