

```
1 <?php
2 /*
3 * The software contained in this file is distributed WITHOUT ANY WARRANTY;
4 * without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
5 * PARTICULAR PURPOSE. THERE IS NO WARRANTY FOR THE SOFTWARE, TO THE EXTENT
6 * PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE
7 * COPYRIGHT HOLDERS PROVIDE THE SOFTWARE "AS IS" WITHOUT WARRANTY OF ANY
8 * KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
9 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
10 * THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH
11 * YOU. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL
12 * NECESSARY SERVICING, REPAIR OR CORRECTION. IN NO EVENT UNLESS REQUIRED BY
13 * APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, BE
14 * LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
15 * CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
16 * SOFTWARE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED
17 * INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE
18 * SOFTWARE TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER HAS BEEN
19 * ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
20 *
21 * Author: Brenton Camac (b1@camacit.com)
22 * Copyright (c) 2009, Camac IT Ltd, All Rights Reserved.
23 *
24 * This software is developed by Camac IT Ltd and the permitted uses of
25 * it are subject to the terms and conditions found in the attached
26 * Software License Agreement (software_license_agreement.pdf)
27 */
28
29
30 /**
31 * This class allows Suma to be customized in terms of what messages and forms
32 * are displayed to the user. It also provides function stubs that you may
33 * extend to be able to receive notification of important events during Suma's
34 * operation - such as when a new user has subscribed, or when a payment attempt
35 * has failed.
36 *
37 * ----- IMPORTANT! ----- READ THIS -----
38 *
39 * Suma will only call these functions when they are found in a file named
40 * suma_customization.php within Suma's plugin directory.
41 *
42 * Hence, TO USE THESE FEATURES, you will need to RENAME THIS FILE to
43 * suma_customization.php in Suma's plugin directory.
44 *
45 * Then, uncomment only those functions you wish to provide custom behavior
46 * for. If a function remains commented, Suma will retain its default behavior.
```

suma_customization-sample.php

```
47 *
48 * Future updates of Suma will likely OVERWRITE the suma_customization-sample.php
49 * file with new function definitions and comments so don't keep your edits
50 * there/here.
51 *
52 * -----
53 *
54 */
55 if (!class_exists('SumaCustomization')) {
56     class SumaCustomization {
57         /**
58          * Called by Suma when Wordpress is performing its init callbacks.
59          * Use this function as a convenient way to register any action hooks or filters
60          */
61         static function init() {
62             }
63
64         /**
65          * The following functions are called when restricted content is encountered
66          * that can not be displayed to the current user.
67          */
68
69         /**
70          * Called when Wordpress is to display a Post (or Page) to a User but
71          * the Post's access controls forbid the specified User from accessing it
72          * according to rules defined for Suma by the administrator.
73          *
74          * @param $post the Wordpress Post or Page
75          * @param $visibility the Suma-defined access control level associated
76          * with this Post or Page. One of 'subscribers', 'public', 'contentClasses'
77          * @param $subscriptionLink a URL to the Suma Subscription/Registration
78          * form hosted on your website.
79          * @return String of HTML. See function for example
80          */
81         static function accessToContentDenied($post, $visibility, $subscriptionLink) {
82             // Suma expects this function to return an HTML fragment explaining to the user
83             // that they do not have the requisite subscription to access this content.
84             // For example, the default used by Suma is:
85             return "<a class='SumaPostContent' onfocus='if(this.blur)this.blur();' href='$subscriptionLink'>Viewing the remainder of this
86 article requires a Subscription</a>";
87         }
88     }
89
90     static function accessToExcerptDenied($post, $visibility, $subscriptionLink) {
```

suma_customization-sample.php

```
92     // Suma expects this function to return an HTML fragment explaining to the user
93     // that they do not have the requisite subscription to access this excerpt.
94     // For example, the default used by Suma is:
95     return "<a class='SumaPostContent' onfocus='if(this.blur)this.blur();' href='$subscriptionLink'>Viewing this excerpt requires a
Subscription</a>";
96 }
97
98 static function accessToCommentTextDenied($post, $visibility, $subscriptionLink) {
99     // Suma expects this function to return an HTML fragment explaining to the user
100    // that they do not have the requisite subscription to access this comment text.
101    // For example, the default used by Suma is:
102    return "<a class='SumaPostContent' onfocus='if(this.blur)this.blur();' href='$subscriptionLink'>Viewing this comment requires a
Subscription</a>";
103 }
104
105 static function accessToCommentExcerptDenied($post, $visibility, $subscriptionLink) {
106    // Suma expects this function to return an HTML fragment explaining to the user
107    // that they do not have the requisite subscription to access this comment excerpt.
108    // For example, the default used by Suma is:
109    return "<a class='SumaPostContent' onfocus='if(this.blur)this.blur();' href='$subscriptionLink'>Viewing this excerpt requires a
Subscription</a>";
110 }
111
112 /*****
113  * The following functions relate to the subscription/registration process
114  *****/
115
116 /**
117  * The following function is called by Suma when listing a Subscription
118  * Plan choice on the first registration page. It provides an opportunity
119  * to specify a html link for that text: e.g. to provide more information
120  * about the plan.
121  *
122  * @param $planID Integer planID of the Subscription Plan
123  * @param $desc String description of the plan as presented to the user on
124  * the form.
125  * @return String href value or null if none
126  */
127 static function getPlanDescriptionHREF($planID, $description) {
128 }
129
130
131 /**
132  * The following function is called by Suma when there are no Subscription
133  * Plans defined and Suma needs to display the first page of the
134  * registration process. To illustrate, the example below shows what amounts
```

suma_customization-sample.php

```
135     * to the same as Suma's current default behavior.
136     * @return String HTML
137     */
138     static function getNoPlansDefinedMsg() {
139         return '<div class="SumaFieldSetRow">
140             <p class="SumaFormErrors">(No subscription plans are available at this time. Please check back later.)</p>
141             </div>
142             <br class="SumaClear"/>';
143     }
144
145     /**
146     * This function is called when Suma is generating the first page of the
147     * four-page registration process. It provides you an opportunity to
148     * insert custom HTML code between the "Account Details" section and the
149     * "Accept Terms of Use" section. This can be used, for instance, if you
150     * need to add a field to ask the user where they heard about your website,
151     * or to collect other survey type information.
152     *
153     * Note. This function is called when the first registration page is
154     * fetched (ie. user's browser performed a GET) AND also if that page
155     * needs to be re-displayed as happens when the page has been submitted
156     * (ie. the user's browser performed a POST) but the form's submitted values
157     * failed to pass all validation tests. Therefore, if the current method
158     * is POST (ie. $_SERVER['REQUEST_METHOD'] == 'POST'), you will probably
159     * want to initialize the fields you are generating with the values supplied
160     * by the user (extract from $_POST array) so that the information they
161     * submitted in the form is still there when the form is re-displayed to them.
162     *
163     * @param $planID Integer planID of the Subscription Plan this user has
164     * selected, or null if the page is being requested for the first time
165     * (ie. is the result of a GET request)
166     * @param $desc String description of the plan as presented to the user
167     * on the form.
168     * @param $role Wordpress role that this user will be associated with should
169     * the registration process create a new Wordpress account for them, or null
170     * if the page is being requested for the first time (ie. is the result of
171     * a GET request)
172     * @return String HTML should be a fragment suitable for inclusion in the
173     * registration form, or null if none
174     */
175     static function getExtraRegistrationFieldSet($planID, $desc, $role) {
176     }
177
178     /**
179     * This function is called when Suma is to generate an extra first page
180
```

```

181 * of the four part registration process (the payment info stage). It
182 * can be considered a part-2 to the first page of the registration
183 * process (user info) and is useful in those situations where the
184 * information you wish to display would be too much to add to the first
185 * page or if the fields you wanted to display somehow depended upon what
186 * information was provided on the first page (e.g. if the user had
187 * selected a particular type of subscription plan).
188 *
189 * @param $planID Integer planID of the Subscription Plan this user has
190 * selected, or null if the page is being requested for the first time
191 * (ie. is the result of a GET request)
192 * @param $desc String description of the plan as presented to the user
193 * on the form.
194 * @param $role Wordpress role that this user will be associated with
195 * should the registration process create a new Wordpress account for them,
196 * or null if the page is being requested for the first time (ie. is the
197 * result of a GET request)
198 * @return String HTML should be a fragment suitable for inclusion in the
199 * registration form, or null if none
200 */
201 static function getExtraRegistrationPage($planID, $desc, $role) {
202 }
203
204
205 /**
206 * This function is called when a user has posted (submitted) the first
207 * page of the registration process. It provides you an opportunity to
208 * validate any HTML fields you may have defined via the
209 * subscriptionExtraInfo() function above. If this function returns a
210 * non-null string to Suma, then Suma will interpret that to be an error
211 * message to be displayed to the user and will not let the user proceed
212 * to the next page of the registration process (the Provide Payment
213 * Info page). If there are no validation errors in the HTML fields you
214 * have defined this function should return null to allow the user to
215 * proceed with the registration process. It is recommended that you NOT
216 * save this information at this stage since the registration process has
217 * not yet been completed (and may be abandoned before being completed).
218 * Better to wait for the subscriptionCompleted() function call to save
219 * any info you have collected from the user (see below).
220 *
221 * @return String validation error message (if any) that is to be
222 * displayed on the form to the user, or null if none. WARNING: User
223 * registration will not proceed until Suma receives a null return value
224 * from this function!
225 */
226

```

```
227     static function validateExtraRegistrationFieldSet() {
228         return null;
229     }
230
231     static function validateExtraRegistrationPage() {
232         return null;
233     }
234
235
236     /**
237     * This function is called when a user has completed Step3 (Confirm Order
238     * Details) of the registration process and Suma has completed its attempt
239     * to create a new Recurring Payment profile for the user and a new
240     * Wordpress user account (unless the user is subscribing from an existing
241     * Wordpress account). In other words, this is the post-registration
242     * notification. The only part of the registration left will be for Suma
243     * to log the user into their new account (if it created a new account for
244     * them) - see Step 4 of the registration process.
245     *
246     * This function provides you an opportunity to save any custom information
247     * you may have collected from the user during the registration process
248     * through the use of the subscriptionExtraInfo() and validateExtraInfo()
249     * functions above.
250     *
251     * @param $userID int Wordpress userID if they already have an account or
252     * if Suma created one for them during registration. Might be null if the
253     * registration process failed to create a new account
254     * @param $username string username that was supplied during the registration
255     * process. May or may not belong to an account; depends on success or
256     * failure of the registration process.
257     * @param $role Role Wordpress role associated with this subscription plan
258     * @param $emailAddress string emailAddress supplied by the user during the
259     * registration process
260     * @param $planID int planID of the Subscription Plan the user selected during
261     * the registration process
262     * @param $planDesc string description component of the Subscription Plan the
263     * user selected during the registration process
264     * @param $paymentMethod string method of payment used during the registration
265     * process, one of HostCreditCard (completed credit card form on your website)
266     * or ExpressCheckout (PayPal), or null if no payment method was used (i.e. guest membership).
267     */
268     static function subscriptionCompleted($userID, $username, $role, $emailAddress, $planID, $planDesc, $paymentMethod = null) {
269     }
270
271
272     /**
```

suma_customization-sample.php

```
273 * Allows you to specify where a new successfully registered subscriber's
274 * browser is redirect to when they have completed their registration
275 * and click on Continue.
276 *
277 * By default, Suma will redirect their browser to the website's home page.
278 * If the return value is null, then Suma's default is used.
279 *
280 * @param int postID of Page or Post which the user tried to access and which
281 * led them to subscribe in order to access it (or null if none)
282 * @return string URL of target landing page
283 */
284 static function afterRegistrationPage($postID) {
285 }
286
287 /*****
288 * The following functions relate to events being received from PayPal
289 * about transitions in the Subscribers' Payment Profile. For subscription
290 * profiles without any payment method associated with them (i.e. Guest
291 * Accounts) some of these notifications are fired nonetheless.
292 *****/
293
294 /**
295 * See PayPal IPN documentation for the conditions under which this
296 * notification is issued. This method is only called after Suma has
297 * verified the authenticity of the notification, performed any necessary
298 * actions of its own, and logged the event in the Administration Log.
299 *
300 * @param $ppParams array of PayPal name/values
301 * @param $profile array of Suma name/values based on the Suma_Profiles table
302 * schema, and will reflect any updates resulting from this notification.
303 */
304 static function subscriptionModificationNotification($ppParams, $profile) {
305 }
306
307 /**
308 * See PayPal IPN documentation for the conditions under which this
309 * notification is issued. This method is only called after Suma has
310 * verified the authenticity of the notification, performed any necessary
311 * actions of its own, and logged the event in the Administration Log.
312 *
313 * Although caused by the same conditions that lead to subscriptionCompleted()
314 * above being called, the information supplied is very different. In this
315 * function, the information is coming from PayPal, and therefore will only
316 * be triggered when the user's registration resulted in a subscription: ie.
317 * would not occur for a guest membership. Furthermore, in this situation
318 * there is no access to the $_SESSION data which your code might have saved
```

suma_customization-sample.php

```
319     * via the subscriptionExtraInfo() during a user's registration process.
320     *
321     * @param $ppParams array of PayPal name/values
322     * @param $profile array of Suma name/values based on the Suma_Profiles table
323     * schema, and will reflect any updates resulting from this notification.
324     */
325     static function subscriptionSignupNotification($ppParams, $profile) {
326     }
327
328     /**
329     * See PayPal IPN documentation for the conditions under which this
330     * notification is issued. This method is only called after Suma has
331     * verified the authenticity of the notification, performed any necessary
332     * actions of its own, and logged the event in the Administration Log.
333     *
334     * @param $ppParams array of PayPal name/values
335     * @param $plan array of Suma name/values based on the Suma_Plans table schema
336     * @param $profile array of Suma name/values based on the Suma_Profiles table
337     * schema, and will reflect any updates resulting from this notification.
338     */
339     static function initialPaymentNotification($ppParams, $plan, $profile) {
340     }
341
342     /**
343     * See PayPal IPN documentation for the conditions under which this
344     * notification is issued. This method is only called after Suma has
345     * verified the authenticity of the notification, performed any necessary
346     * actions of its own, and logged the event in the Administration Log.
347     *
348     * @param $ppParams array of PayPal name/values
349     * @param $profile array of Suma name/values based on the Suma_Profiles table
350     * schema, and will reflect any updates resulting from this notification.
351     */
352     static function paymentNotification($ppParams, $profile) {
353     }
354
355     /**
356     * See PayPal IPN documentation for the conditions under which this
357     * notification is issued. This method is only called after Suma has
358     * verified the authenticity of the notification, performed any necessary
359     * actions of its own, and logged the event in the Administration Log.
360     *
361     * @param $ppParams array of PayPal name/values
362     * @param $profile array of Suma name/values based on the Suma_Profiles table
363     * schema, and will reflect any updates resulting from this notification.
364     */
```

```
365 static function endOfTermNotification($ppParams, $profile) {
366 }
367
368 /**
369  * See PayPal IPN documentation for the conditions under which this
370  * notification is issued. This method is only called after Suma has
371  * verified the authenticity of the notification, performed any necessary
372  * actions of its own, and logged the event in the Administration Log.
373  *
374  * @param $ppParams array of PayPal name/values
375  * @param $profile array of Suma name/values based on the Suma_Profiles table
376  * schema, and will reflect any updates resulting from this notification.
377  */
378 static function paymentSkippedNotification($ppParams, $profile) {
379 }
380
381 /**
382  * See PayPal IPN documentation for the conditions under which this
383  * notification is issued. This method is only called after Suma has
384  * verified the authenticity of the notification, performed any necessary
385  * actions of its own, and logged the event in the Administration Log.
386  *
387  * @param $ppParams array of PayPal name/values
388  * @param $profile array of Suma name/values based on the Suma_Profiles table
389  * schema, and will reflect any updates resulting from this notification.
390  */
391 static function paymentFailedNotification($ppParams, $profile) {
392 }
393
394 /**
395  * See PayPal IPN documentation for the conditions under which this
396  * notification is issued. This method is only called after Suma has
397  * verified the authenticity of the notification, performed any necessary
398  * actions of its own, and logged the event in the Administration Log.
399  *
400  * @param $ppParams array of PayPal name/values
401  * @param $profile array of Suma name/values based on the Suma_Profiles table
402  * schema, and will reflect any updates resulting from this notification.
403  */
404 static function processCancelledNotification($ppParams, $profile) {
405     // $user_info = get_userdata($userID);
406     // $loginName = $user_info->user_login;
407 }
408
409 /*****
410  * The following functions relate to information displayed to logged in
```

suma_customization-sample.php

```
411     * users with Subscriptions regardless of the state of their Subscription:
412     * e.g. expired, current, etc.
413     *****/
414
415     /*
416     * Suma calls this function when it needs to generate the
417     * list of links to display within Suma's Login Widget to a subscriber
418     * (who is logged in, obviously). This function allows you to add your
419     * own links to that list. The list of links you provide should
420     * be packaged into an array where the key of the array is the
421     * label of the link and the value is the URL of the link.
422     *
423     * @param $userID int Wordpress userID of the user making the page request
424     * @return array of [label]=>URL to be included in the list of links
425     * presented to the user in their Suma Login widget panel
426     */
427     static function accountWidgetLinks($userID) {
428     }
429
430     /**
431     * The following function is called upon by Suma to generate the user's
432     * Account page.
433     *
434     * @param $userdata the global Wordpress variable containing user
435     * data of the current user
436     * @param $errors an array of key->message, or true if the user
437     * account was updated
438     * @param $targetOfPostURL the URL that the HTML form should POST to in
439     * order to be processed
440     * @return String of HTML
441     */
442     static function userAccountPage($userdata, $errors, $targetOfPostURL) {
443     }
444 }
445 }
446 ?>
```