Site Store Pro | WP Cart Pro Plugin Developer Guide

Version 2.0 : Released January 9, 2021

The Site Store Pro shopping cart application and the WPCartPro Wordpress plugin shopping cart can be easily extended by developing new plugins or modifying existing plugins to suit your specific e-commerce site requirements.

The plugins bridge the gap between the core cart features that are released and updated by Site Store Pro and the need to add new functionality to the cart without modifying the core cart files. (Note: if needed, you can also modify the core cart files and record them in your admin as a customization to prevent your modification(s) from being overwritten when you install a core cart update.)

The shopping cart plugins are located in the **/sitestorepro/plugins/** directory of the cart installation.

Due to the mission critical nature of many ecommerce sites that run the Site Store Pro and WPCartPro and to avoid upgrade conflicts that could potentially affect site uptime, the cart plugins are installed by uploading the specific plugin's folder and xml configuration file to **/sitestorepro/plugins/** via FTP to the server/hosting account.

Once the plugin has been uploaded to the install, it will automatically be installed (loaded) in the plugin manager in the admin area, and the merchant can activate/deactivate the plugin, manage the settings for the plugin and, if needed, delete it from the system.

The following categories of plugins can be installed into the cart as of January 8, 2021.

Payment Providers (payment)
Shipping Providers (shipping)
Mailing (Email) List Providers (email)
Vertical Sidebar Menu (sidenav)
Top Navigation Menu (topnav)
Quick Cart Display (quickcart)
Search Bar Display (searchbar)
Home Page | Site Display Elements (display)
Site & Admin Management Tools (tools) – New Feature - January 2021

Plugins with a type of “payment” , “shipping” or “email” have an associated API that sends in valuable data to the plugin automatically. Please see the section at the end of this document called “Plugin API Reference”.

Plugins with the type of “tools” are used for site management such as data backups and can be placed on any site .php file and be triggered automatically. Please see section as the end of this document called “Tools Plugin Reference”.

Plugin Structure

Plugins must follow the structure of :

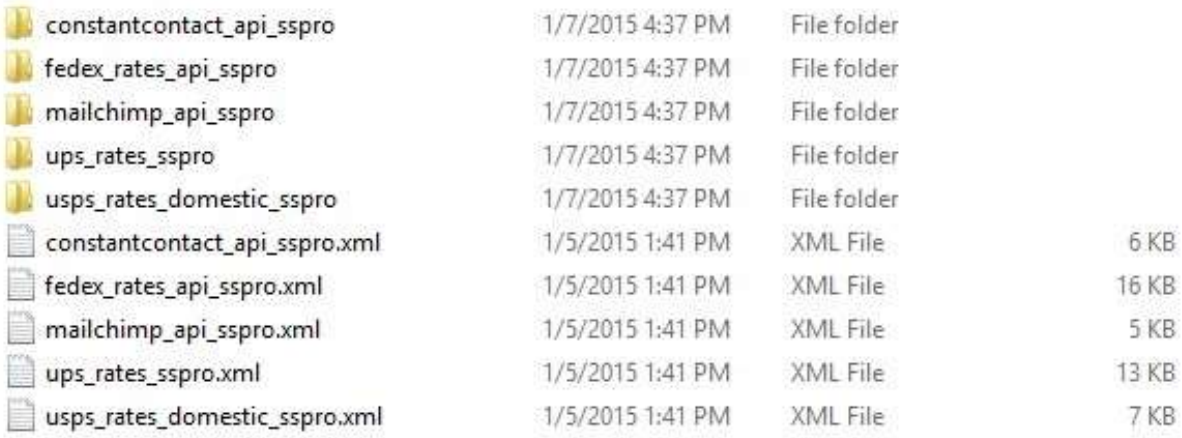
plugin.xml file --
- plugin folder
  >> plugin's primary php class file

The plugin xml file and folder name can be whatever you choose since they are both auto-loaded. However, to avoid conflicts with other developer's plugins, you should name all your files with a unique naming scheme. The plugin xml file, folder and primary PHP class file must all have the same name)before the file extension)

For example, if your company is called XYZ Development and you are creating a plugin for a custom search keyword bar, your file structure may look something like:

/sitestorepro/
  /plugins/
    /keywordsearch_xyzdev/
     keyywordsearch_xyzdev.php
  keywordsearch_xyzdev.xml

Below is as screen shot of a default Site Store Pro install with the standard (included) plugins directory structure. Notice how the plugin’s XML file is the same name as the folder that contains the plugin’s files.



constantcontact_api_sspro	1/7/2015 4:37 PM	File folder	
fedex_rates_api_sspro	1/7/2015 4:37 PM	File folder	
mailchimp_api_sspro	1/7/2015 4:37 PM	File folder	
ups_rates_sspro	1/7/2015 4:37 PM	File folder	
usps_rates_domestic_sspro	1/7/2015 4:37 PM	File folder	
constantcontact_api_sspro.xml	1/5/2015 1:41 PM	XML File	6 KB
fedex_rates_api_sspro.xml	1/5/2015 1:41 PM	XML File	16 KB
mailchimp_api_sspro.xml	1/5/2015 1:41 PM	XML File	5 KB
ups_rates_sspro.xml	1/5/2015 1:41 PM	XML File	13 KB
usps_rates_domestic_sspro.xml	1/5/2015 1:41 PM	XML File	7 KB

PLUGIN XML FILE

The plugin's XML file is installed in the root of the /sitestorepro/plugins/ folder and contains the configuration settings to activate the plugin on the installation.

All fields listed below EXCEPT the <fields></fields> tag are required. If your plugin does not have any runtime configuration settings or input required by the merchant, you can omit the <fields></fields> requirement. Any fields shown below that says "This is a required field" should have data inside the xml tag.

The plugin XML file contains the following fields.

name: This is the name of the plugin that will appear in the plugin directory. This is a required field.

version: This is the release version of the plugin. Changing this value will automatically refresh/update the plugin when it is installed. This is a required field.

type: This is the function type of the plugin. Allowed values are: payment, shipping, email, sidenav, topnav, quickcart, searchbar, display. This is a required field.

author: The developer of the plugin. This is a required field.

primary_filename: **This is the most important field in the xml file**. This field is the name of the xml file, plugin folder and the plugin's php class file that is located in the plugins' folder. This must be a unique name. This is a required field.

install_type: The platform for the plugin: Allowed values are 0, 1 and 2. 0 = All Platforms. 1 = Site Store Pro only. 2 = Wordpress (WPCartPro) only. This is a required field.

description: This field is a short description of the plugin that will appear in the plugin directory. This is a required field.

short_code: This is the friendly name of the plugin that will be used to easily identify the plugin when it is used for display purposes. This is a required field.

activation_required: This field sets if the plugin must be activated with the activation function listed in the plugin's php class file. (i.e. if you are using an external service to activate the plugin for usage via a serial number or activation code) Allowed values are yes or no. This is a required field.

activation_instructions: This is the text that will appear on the activation screen. HTML code must be enclosed within <![CDATA[ ]]> This is a required field if you are requiring activation from a external service.

activation_failed: This is the message that will appear if the plugin activation process failes. HTML code must be enclosed in <![CDATA[…]]> This is a required field if you are requiring activation from a external service.

usage_instructions: This is the instructions that will appear on top of the plugin settings (management) page in the admin ara. HTML code must be enclosed in <![CDATA[…]]> This is a required field.

help_info: this is the info that will appear if the user clicks the help icon on the plugin page.

help_url: this is the url for a additional help page for the plugin.

show_credit_card_form: if set to “yes”, the credit card entry form will appear on the site during checkout. This field is only valid if the plugin type is 'payment'. Allowed values are yes or no. This is a required field even for no payment type plugins.

paypal_standard: if set to yes, the plugin will be classified as a paypal standard plugin. This field is only valid if the plugin type is 'payment'. Allowed values are yes or no. This is a required field even for non payment type plugins.

paypal_advanced: if set to yes, the plugin will be classified as a paypal advanced payments plugin. This field is only valid if the plugin type is 'payment'. Allowed values are yes or no. This is a required field even for non payment type plugins.

paypal_express: if set to yes, the plugin will be classified as a paypal express payments plugin. This field is only valid if the plugin type is 'payment'. Allowed values are yes or no. This is a required field even for non payment type plugins.

amazon_payments: if set to yes, the plugin will be classified as a amazon payments plugin. This field is only valid if the plugin type is 'payment'. Allowed values are yes or no. This is a required field even for non payment type plugins.

custom_plugin_page: if a page (filename) is entered, the system will load the custom admin file instead of the default plugin manager. (This feature has been deprecated as of January 2021. Admin manager pages are not created by having a include file called /admin/include.php in the plugin's folder.

The XML fields below are for the plugin setting fields that control the plugin runtime settings and required settings such as Third Party API username and passwords. The fields below are enclosed in ***<fields></fields>.*** *The entire section can be omitted from your plugin file if your plugin does not require any user interaction or runtime configuration settings.*

field_name: this is unique name of the form field that the user will enter plugin settings. Should all be one word (no spaces or special characters) and lower case is preferred. This is a required.

field_label: this is the label of the form field that the merchant will see on the plugin settings manager page for this field. This is a required.

field_type: this is the form field type: allowed values are; input, radio, checkbox, select, textarea and upload. This is a required.

field_data_format: this is the format of the data that the merchant will enter (used for validation). Allowed values are: string, integer, float, date, file. This is a required.

field_default_value: this is the default value (text/setting) that is loaded into the form field when the merchant first opens the plugin settings.

field_selections: this is a comma separated list used to populate lists (select menus) or radio groups. Only valid if the field_type of the form field is 'select' or 'radio'.

field_min_value: the min value of the form field entry. Only valid if the field_type of the form field is integer or float.

field_max_value: the max value of the form field entry. Only valid if the field_type of the form field is integer or float.

field_wysiwyg_editor: if set to yes and the form field type is a textarea, a HTML editor will appear on the plugin settings page for the field.

field_help_tip: the text that will appear as a help tip for the form field. Text only, not html or special characters.

field_required: sets if the field is required or not. Allowed values are yes or no. This is a required.

field_error: error message that willl be displayed to the user if the field_required value is set to yes and the merchant does not enter in data when submitting the plugin settings. This is a required field if field_required is set to “yes”.

field_html : optional HTML content that can be displayed above the form field on the settings form. HTML code must be enclosed in <![CDATA[…]]>

As an example, below is the xml file for the free plugin that we created for the MailChimp.com API. (For additional reference, please view the files for the plugins in the /sitestorepro/plugins/ folder to use a guide/reference for your custom plugin development.)

```
< ?xml version="1.0" encoding="utf-8"?>
<plugin>
    <name>Mail Chimp API Plugin</name>
    <version>1.0</version>
    <type>email</type>
    <author>Site Store Pro</author>
    <primary_filename>mailchimp_api_sspro</primary_filename>
    <install_type>0</install_type>
    <description>This plugin allows customers to be added directly to your MailChimp Email List.</description>
    <short_code>Mail Chimp API</short_code>
    <activation_required>no</activation_required>
    <activation_instructions></activation_instructions>
    <activation_failed></activation_failed>
    <activation_completed>Your plugin was successfully activated</activation_completed>
    <usage_instructions><![CDATA[<p>The MailChimp API code plugs into the customer account manager module insert and update functions and provides seemless integration between your <a href="customers.php">online store customer database</a> and your <a href="http://www.mailchimp.com" target="_blank">MailChimp Email list</a>. </p><p>The following email fields will be created or updated on your MailChimp.com list automatically if you enable the Mail Chimp API:<br><strong>Email Address<br> First Name<br> Last Name</strong></p><p>If the customer chooses to opt-in during registration or opt-in/opt-out anytime from their account manager, their account will be automatically updated on MailChimp.com.</p><p><strong>You should export your current online store customer email list from your admin and import it to your MailChimp List before or right after you install the MailChimp plug-in</strong>. The plug-in will only synchronize ne emails and when an existing customer updates their opt-in preferences. Your original (past) online store emails must be imported into your MailChimp list unless they have already been imported previously. To export your current opt-in emails from your online store, login to
/estore_admin/, click on the '<a href="reports.php">Reports</a>' tab and then click on 'Export All Customer Emails (CSV)'. The exported CSV file can be easily imported into your MailChimp list from inside your MailChimp Accout Manager (list import functon). </p ><p>Configuration Settings: </p><p><strong>Enable Mail Chimp API</strong>: Check box to turn on the Mail Chimp API. </p <p><strong>MailChimp APIKey </strong>= Mail Chimp API Key Is Available Under Account &amp;gt; API Keys and Info In Your <a href="http://www.mailchimp.com" target="_blank">MAilChimp.Com Account</a><br> <strong>MailChimp Unique ListID</strong> = Mail Chimp UniqueList ID is Available under List Settings
```

```
&gt; List Settings &amp; Unique ID in Your <a href="http://www.mailchimp.com" target="_blank">MailChimp.com Account </a></p>]]></usage_instructions>
    <help_info>This plugin allows you to connect to MailChimp List API. For help, please contact customer support.</help_info>
    <help_url>https://www.sitestorepro.com/support.php</help_url>
    <show_credit_card_form>no</show_credit_card_form>
    <paypal_standard>no</paypal_standard>
    <paypal_advanced>no</paypal_advanced>
    <paypal_express>no</paypal_express>
    <amazon_payments>no</amazon_payments>
    <custom_plugin_page>no</custom_plugin_page>
    <fields>
                <field_name>mail_chimp_apikey</field_name>
                 <field_label>Mail Chimp API Key</field_label>
                 <field_type>input</field_type>
                 <field_data_format>string</field_data_format>
                 <field_default_value></field_default_value>
                 <field_selections></field_selections>
                 <field_min_value></field_min_value>
                 <field_max_value></field_max_value>
                 <field_wysiwyg_editor>no</field_wysiwyg_editor>
                <field_help_tip></field_help_tip>
                 <field_required>yes</field_required>
                 <field_error>Please enter your API Key</field_error>
                 <field_html></field_html>
    </fields>
    <fields>
                 <field_name>mail_chimp_list_id</field_name>
                 <field_label>Mail Chimp List ID</field_label>
                 <field_type>input</field_type>
                 <field_data_format>string</field_data_format>
                 <field_default_value></field_default_value>
                 <field_selections></field_selections>
                 <field_min_value></field_min_value>
                 <field_max_value></field_max_value>
                 <field_wysiwyg_editor>no</field_wysiwyg_editor>
                <field_help_tip></field_help_tip>
                 <field_required>yes</field_required>
                 <field_error>Please enter your List ID</field_error>
                 <field_html></field_html>
    </fields>

</plugin>
```

Plugin API Reference

All plugins will have the following information (variables) available inside the plugin:

$this->plugin_id = Md5 string of the plugin's unique filename)
$this->activation_key = Activation key for paid/externally activated plugins
$this->sspro_db = The mysqli database connection object ($sspro_db).

The plugin API ID, Activation Key and the mysqli DB connection object can be used for advanced plugin functions such as runtime verification of activation status (for paid or expiring plugins) and direct database calls from the plugin

Plugins with the type of "shipping", "email" or "payment" also include powerful type-specific APIs that send in required data to the plugin to use for payment processing, shipping rate calculation or email list subscription.

Plugins can also retrieve their own settings from any external (runtime) file that is used by the plugin by calling the following function: **sspro_plugin_field_value**

The **sspro_plugin_field_value** function allows the plugin developer to retrieve a specific form field setting (value) that was required from the plugins XML file outside of the plugin's main php class file. For example, the developer might need to retrieve specific information from the plugin regarding the location of a third party payment post URL when a payment verification is posted to the site. The format of the function is :

sspro_plugin_field_value(primary_filename, field_name, $sspro_db);

primary_file_name = the name of the plugin's unique file name as set in the XML doc

field_name = the unique field name for a setting for the plugin as set in the XML doc

$sspro_db = the site mysqli database connection object.

APIs For Specific Plugin Types

Below are the three APIs for shipping, email and payment type plugins.

SHIPPING API (type = shipping)

When a plugin is developed for calculating shipping rates, all the data values such as the customer's address, cart weight and order total are sent into the plugin using the $sspro_shipping_api array.

Below are the data variables that are available for shipping type plugins:

$this->CartWeight	= The total cart weight
$this->origin_zipcode	= The merchant's zip code that is set in the web-based admin area
$this->origin_country	= The merchant's country
$this->sspro_ship_address1	= The address where the package will be shipped (line #1)
$this->sspro_ship_address2	= The address where the package will be shipped (line #2)
$this->sspro_ship_city	= The zip or postal code of where the package will be shipped
$this->sspro_ship_zip	= The city where the package will be shipped
$this->sspro_ship_region	= If address is outside North America, this is used as the state/region
$this->sspro_ship_state	= US State or Canadian Province (North American Addresses Only)
$this->sspro_ship_country	= The country where the package will be shipped
$this->sspro_shipping_total	= The total amount of order before shipping is applied.
$this->shippable_items	= the total number of items in the order that are shippable

EMAIL LIST API (type = email)

When a plugin type is set to 'email', the following fields are available

$this->first_name	= The customer or user's first name
$this->last_name	= The customer or user's last name
$this->email	= The customer or user's email address
$this->receive_emails	= If customer/user has opted in or out

PAYMENTS API (type = payment)

When a plugin type is set to payment, the following information is available inside the plugin to process the payment:

$this->INTERNAL_ORDER_ID	= the auto-generated, sequential order id (admin area only)
$this->EXTERNAL_ORDER_ID	= the customer's unique order ID number
$this->order_total	= this is usually the value that you send for authorization

Customer Order (User Account Registration) Information:

$this->customers_email	= The customer's email address
$this->Shipping_First_Name	= The customer's first name
$this->Shipping_Last_Name	= The customer's last name
$this->Shipping_Address1	= The customer shipping address (line 1)
$this->Shipping_Address2	= The customer's shipping address (line 2)
$this->Shipping_City	= The customer's shipping city
$this->Shipping_State	= The customer's state (US and Canada Only)
$this->Shipping_Region	= The customer's region/province (Outside of North America)
$this->Shipping_ZipCode	= The customer's zip code
$this->Shipping_Country	= The customer's country
$this->User_Payment_Token	= The unique security token value for the customer

Payment Provider Plugins Checkout Include Files (Optional):

As of January 2021, you can now include three PHP include files that, if present in the plugin's root directory, will be automatically loaded on the /checkout/order_review.php page. This feature was created to allow the loading of required payment provider initialization data, header Javascript/jQuery scripts and an embedded payment form. The three files must be in the root folder of the plugin's directory to be automatically loaded when the plugin is called on the site's billing page.

The three OPTIONAL plugin include files are :

loader.php: This file loads required php files such as API/SDK files for the payment provider. It will be loaded before the page is displayed to the user and can be used to initialize required php files server-side.

Header.php: This file is automatically loaded in the <head> … <head> area of the HTML and is typically used to add vendor (payment provider) required Javascript to initialize a connection to the payment processor, sent API credentials, etc.

payform.php: If present in the plugin's folder, this file will load in the billing form area of the page. This include file is used to auto load embedded payment forms such as those used by Square, Stripe and PayPal advanced payments. T

Credit Card Order API Information

Note: all values below may not be available. (Depends on your billing form settings and if your plugin is accepting credit card info.)

IMPORTANT: For PCI compliance and data security, you may not save or send credit card or billing related data anywhere EXCEPT directly to the payment process to authorize the sale/order.

Payment information is never stored in the application database and is cleared instantly after the order is submitted securely to the payment processor. You MUST be using https:// (SSL) if you will be accepting credit card payments directly on your website and your site must pass a quarterly PCI compliance SCAN to verify that your hosting account/server is secure. The Site Store Pro cart is certified secure and PCI compliant in its default release configuration but you must verify that your specific server/hosting account setup is secure and compliant prior to accepting credit cards and submitting them from your site to the payment processor through a payment plugin.

Credit card and billing information cannot be saved into the database, session variables or cookies or emailed to the store admin or customer under any circumstances.

The billing API information is only available as temporary PHP server side variables the instant the customer submits their order for authorization/payment. If you save any of the Billing API information into any type of session, cookie or database record, or you email any of the information listed below, you will be in violation of PCI compliance and you will void any support agreement you have with Site Store Pro.

$this->payment_method	= card type for payment (ie. VISA, MASTERCARD, AMEX, etc)
$this->customer_name_on_card	= name on the credit card
$this->credit_card_number	= credit card number
$this->card_expiration_month	= expiration month of the card
$this->card_expiration_year	= expiration year of the card
$this->card_cvv_value	= card verification value (CVV)
$this->card_billing_address	= billing address for card
$this->card_billing_city	= billing city for card
$this->card_billing_zipcode	= billing zip code for card
$this->card_billing_region	= billing region (if outside US)
$this->card_billing_state	= billing state (US)
$this->card_billing_country	= billing country

$this->card_billing_company = company name
$this->cart_billing_phone = billing phone number

Subscription | Recurring Payment API Fields

The following fields (data) may be available if your payment processor supports recurring payments and/or your item/order is set to be a subscription payment.

$this->subscription = any value beside 0 (zero) indicates a subscription order
$this->subscription_interval = weekly, monthly, yearly, etc (set in product manager)
$this->subscription_frequency = number of payments per interval (usually 1)
$this->subscription_payments = of total payments (0 = forever)
$this->subscription_trial = trial period for (0 = none)
$this->subscription_descrip = description of this payment (i.e. Monthly Gym Membership)
$this->subscription_maxfail = max number of failed payment... required for some processor.
$this->subscription_custom1 = custom subscription field
$this->subscription_custom2 = subscription field
$this->subscription_custom3 = custom subscription field

Tools Plugin Reference

As of January 8th, 2021, you can now include site management and utility plugins on your installs that are automatically (or manually) loaded to perform site functions such as database and file backups, auto email senders, folder clean-ups, custom admin reports, etc. The plugins will show up in the admin plugin manager as “Tools” and include a new feature that allows you to have your own custom designed admin management features.

Plugins will the type of “tools” can also have an include file located in the plugin’s directory of /admin/include.php that will automatically load a management page to control the features and runtime of the tool plugin. To prevent direct loading of your plugin’s admin file and for security, the following code should be placed on top of all your plugin .php files:

```
if (!defined('SSPRO_INSTALL_TYPE')) { // kill page load if loaded directly
   die();
}
```

Tool plugins can be triggered on any site php page by adding the following code snippet (shortcode):

<?php sspro_tools(plugin file name, tool type/method, tool options); ?>

The three function arguments are:

- The primary file name of the plugin as specified in your plugin XML file
- The tool's command / method that will be used to initiate a specific function in your plugin class file
- The tool's command options which will be used to perform a specific task of the command/method in the second variable.

The function arguments can either be php variables or hardcoded values encapsulated by single or double quotes. You may pass whatever data you like in the command and option arguments including strings, integers, arrays, etc since you will be processing the argument data inside your class file. However, the first argument, the plugin_file_name, must be sent as a string value.

The sspro_tools function passes data and commands via the above three arguments into a function inside your plugin class file called : tool_set

The tool_set function is where you will preform the bulk of the tool's utility features.

below is a example of the structure of a basic site tools plugins:

```
<?php

if (!defined('SSPRO_INSTALL_TYPE'))
{ // kill page load if loaded directly
   die();
}

class tool_name
{

   /**
    * Generic Site Tool
    *
    * Author: Site Store Pro
    * @param  string $sspro_plugin_id The internal db record ID for this plugin
    * @param  string $sspro_activation_key The activation key for the plugin.
    * @param  array  $sspro_plugin_settings The options/settings that are specified in the admin for this plugin (as defined in plugin's xml file)
    * @param  object $sspro_db This is the mysqli database object connection.
    */
```

```
    public function __construct($sspro_plugin_id, $sspro_activation_key, $sspro_plugin_settings, $sspro_db)
    {

        $this->plugin_id = $sspro_plugin_id; // this is the internal ID for the plugin.. sent in from the plugin's database
record.
        $this->activation_key = $sspro_activation_key; // this is the activation key for the plugin.. sent in from the
plugin's database record.
        /* Plug-In Configuration | Settings */

        if (!empty($sspro_plugin_settings))
        {

                        // plugin settings set in your XML file are loaded here


        }

        $this->sspro_db = $sspro_db; // this is the mysqli database connection object


            }

    //  * @param  string $tool_type is the command used to initiate a specific feature of the tool
    //  * @param  string $tool_options are the specific features in the above command to trigger

    public function tool_set($tool_type, $tool_options)

            {

                        // this loods the tool features and runs them on the site...

    }

              ### End YOUR PLUGIN TOOL FEATURED###


    ############################ The following functions should be included with all plugins
#####################################



    /**
     * Activate Plugin Function (REQUIRED FOR ALL PLUGINS)
     *
     * @param  string  $sspro_activation_key The key that turns on this plugin. Will be random value for plugins that
don't provide it
     * @param  int     $sspro_plugin_id the database record id of the plugin
     *
```

```
     * @return array     $activation_status > active (1 or 0), serialnumber
     */

    public function ActivatePlugin($sspro_plugin_id, $sspro_activation_key)
    {

        // you can add your own activate process here which calls an activation process on your own server via curl,
etc.
        // change the condition below to activate the plugin
        $activation_status = array();
        $activation_status['active'] = 0;
        $activation_status['serialnumber'] = "";

        if (1 == 1)
        { // add your own validation condition (if required)
            $activation_status['active'] = 1; // return 1 for activated, 0 for activation failed
            $activation_status['serialnumber'] = ""; // serial number is not required but can be provided for paid plugins

        }
    }

    private function CheckPluginStatus($sspro_activation_key)
    {

        // if you want to validate that the user is licensed to use the plugin every time it is run, you can do that here
        // however, it is not recommended to do check every time plugin is called if it might affect page load time, etc
        $plugin_valid = false;

        if (1 == 1)
        { // add your own validation condition (if required)
            $activation_status['active'] = 1; // return 1 for activated, 0 for activation failed
            $activation_status['serialnumber'] = ""; // serial number is not required but can be provided for paid plugins

        }

        return $activation_status;

        return $plugin_valid;
    }

}
```