c5extras.com

eCommerce Express Custom Gateway Developer Guide

The best way to start is build your payment gateway based on the "Custom Gateway" package.

Package Settings

If you don't know how to create a package yet feel free to look at the Concrete5 tutorial:
http://www.concrete5.org/documentation/developers/system/packages/

Make sure to rename the package directory "`custom_gateway`" as well as the package handle and match it with the package controller and modify the necessary information based on your payment gateway.

Methods:

a. `getPackageName` – This will be used as a reference title to your payment gateway inside the eCommerce Express.
b. `getPackageDescription` – This should contain the description of your package and not the payment description. See `getPaymentDescription`
c. `getPaymentDescription` – This will be used as a reference to your payment gateway description on the "Payments" dashboard page when you click the tooltip.

Dashboard

Controller

It is important that you match the filename and the controller name to your package to prevent confusion. The structure below should be strictly followed in order for your payment gateway to be recognized by eCommerce Express.

```
/packages/custom_gateway/controllers/dashboard/jb_ecommerce_express/gateways/custom_gateway.php
```

As you can see the filename is set to "custom_gateway" which matches the package name. Make sure to rename the package controller accordingly:

In this example below is the name of the controller.

```
DashboardJbEcommerceExpressGatewaysCustomGatewayCont
roller
```

If you have named your gateway to be “paypal_gateway” it should look like this

```
DashboardJbEcommerceExpressGatewaysPaypalGatewayCont
roller
```

Notice the camel case. This is how it should look like if it’s separated by underscores. For more information
http://www.concrete5.org/documentation/developers/pages/mvc-approach/

If you named your controller file correctly it should automatically set the gateway handle for you which is handled by the constructor:

```
public function __construct() {
     $this->gateway = basename(__FILE__, '.php');
}
```

The gateway handle is used as a reference throughout eCommerce Express. On the other hand, the package handle is used as reference throughout the Concrete5 system and both handles should be unique on eCommerce Express and Concrete5 respectively.

Methods

a) `view` – This should work as is and needs no modifications.
b) `saveform_settings` – This takes care of your payment gateway settings which is configured from the view:

   ```
   /packages/custom_gateway/single_pages/dashboard/jb
   _ecommerce_express/gateways/custom_gateway.php
   ```

   The necessary data should be saved on an array:

   ```
   $settingsSaved['enableSandbox']
   $settingsSaved['checkoutUrl']
   $settingsSaved['checkoutUrlSandbox']
   ```

   See the controller for details:
   ```
   /packages/custom_gateway/controllers/dashboard/jb_
   ecommerce_express/gateways/custom_gateway.php
   ```

View

Again it is important that you match the filename of the view and the controller. So in this case:

```
/packages/custom_gateway/single_pages/dashboard/jb_e
commerce_express/gateways/custom_gateway.php
```

You should only concern yourself on the form data and the input IDs. If you look at line 46 on the view file it contains the necessary input elements and the matching attribute names:

```
$form-
>checkbox("enableSandbox",1,$config['enableSandbox'])
$form->text("checkoutUrl", $config['checkoutUrl'])
$form->text("checkoutUrlSandbox",
$config['checkoutUrlSandbox'])
```

Those are some of the examples found in the view file and match it on the controller's array keys. See the methods section of the Controller above for details.

Form Submission

Now here's where the payment gateway gets its cart details and send it through your service provider.

Methods

a) Retrieve the merchant information using the $merchantInfo array:

   Example:

   ```
   $merchantInfo['paymentMethodKey']
   ```

   This returns the User ID registered on the payment gateway.

b) Extend the eCommerce Express options and make sure to prepend it with your gateway handle to make it unique. See line 12-13 on:

   ```
   /packages/custom_gateway/elements/gateways/custom_ga
   teway.php
   ```

c) Moreover, extend the checkout method of your script:

   On line 31:
   ```
   /packages/custom_gateway/elements/gateways/custom_ga
   teway.php
   ```

   The "data" parameter is a JSON object which contains "id" and "inv". The "id" parameter is the unique transaction ID you can use it to send it to your payment gateway if need be. On the other hand, "inv" is the invoice number generated using the time() php function which can be used to uniquely identify each invoice.

   Cart Properties and Options:

   - `this.quantity` – This is the total quantity inside the Cart.

- `this.subtotal` – This is the subtotal Price of the Cart less the Shipping, Discount, and Tax.
- `this.total` – This is the total calculated Price of the Cart items.
- `this.shipping` – The calculated Shipping cost based on the Shipping rate specified on the dashboard settings.
- `this.discount` – The calculated Discount based on the codes that is applied or in the dashboard settings.
- `this.options.currencyCode` – ISO 3 letter currency code. (USD,ZAR,etc)
- `this.options.cancelurl` – This is where the gateway redirects if the transaction is cancelled (it depends on the provider).
- `this.options.returnurl` – This is where the page is redirected after the transaction is completed.
- `this.cart` – This is the container of the shopping Cart. Traversing through the items in the Cart can be done like this:

  ```
  this.cart.each(function () {
  });
  ```

  And under this method you can use these attributes for each item ("this" refers to the current item being traversed):
  - `this.id` – The ID of the item.
  - `this.thumb-id` – The thumbnail ID of the item.
  - `this.title` – The Label of the item.
  - `this.description` – The Description of the item (undefined if not specified)
  - `this.price` – The Price of a single item.
  - `this.weight` – The Weight of a single item.
  - `this.quantity` – The total Quantity of the item being purchased.
  - `this.stock` – The current Stock of an item.

Quick Example

Adding new payment gateway settings field.

Let's say we want to add a Merchant email field.

1. First is to add an input field on the View file:

```
/packages/custom_gateway/single_pages/dashboard/jb_ecomme
rce_express/gateways/custom_gateway.php
```

Add this on line 68:

```
<tr>
```

```
 <td style="width:120px;" valign="top">
  <?php echo t('Merchant Email'); ?>
 </td>
 <td>
  <?php echo $form->text("merchantEmail",
($config['merchantEmail']!='' ? $config['merchantEmail']
: '')); ?>
  <div class="ccm-note">
   <?php echo  t('A copy of orrder transactions will be
sent here.');?>
  </div>
 </td>
</tr>
```

2. Now add the field to be saved on saveform_settings on the Controller.

```
/packages/custom_gateway/controllers/dashboard/jb_ecommer
ce_express/gateways/custom_gateway.php
```

Add this on line 38:

```
$settingsSaved['merchantEmail']=$_POST['merchantEmail'] .
'';
```

3. Onto the gateway script found here:

```
/packages/custom_gateway/elements/gateways/custom_gateway
.php
```

Add this on line 18:

```
merchant_email: '<?php echo
isset($config['merchantEmail']) ?
$config['merchantEmail'] : "";?>',
```

4. Lastly we can now use the `merchant_email` to add it to our form data and be submitted on our payment provider.

Now on line 92 add this:

```
vars.push([ 'merchant_email', config.merchant_email ]);
```

That should be all to get started on creating your own custom gateway. Reminder this gateway doesn't actually work, it should only be used as a reference.