

VevoCart

Version 4.1

Payment Gateway Integration Guide

VevoCart.com

support@vevocart.com

March 18, 2010

Table of Contents

1.	Prologue.....	2
2.	Source Code Structure	2
3.	Payment Module Class Hierarchies.....	3
3.1	PaymentMethod	3
3.2	HostedPaymentMethod	4
3.3	OnWebsitePaymentMethod.....	4
4.	User Controls for Express Checkout Button	5
5.	User Controls for Admin Configuration	5
5.1	Admin User Control Base Class	5
5.2	User Control Layout (.ascx File)	6
5.3	User Control Code-Behind (ascx.cs File)	6
6.	Database Structure	7
6.1	PaymentOption Table	7
6.2	PaymentOptionLocale Table.....	10
6.3	vevopay_PaymentOption Table.....	10
6.4	SQL Script to Insert New Payment	11
7.	Configuration Parameters.....	12
8.	How to Integrate a New Payment Gateway	12

1. Prologue

VevoCart 4.1 introduced a flexible payment framework that developers can add new payment gateways easily.

The framework is designed to allow non-intrusive payment gateway addition. The payment gateway can be packaged and distributed in the way that the clients just can copy the files and run database upgrade script. There would be no need to edit the file.

Chapters 2 to 7 provide necessary technical background to understand the structure of payment processing in VevoCart.

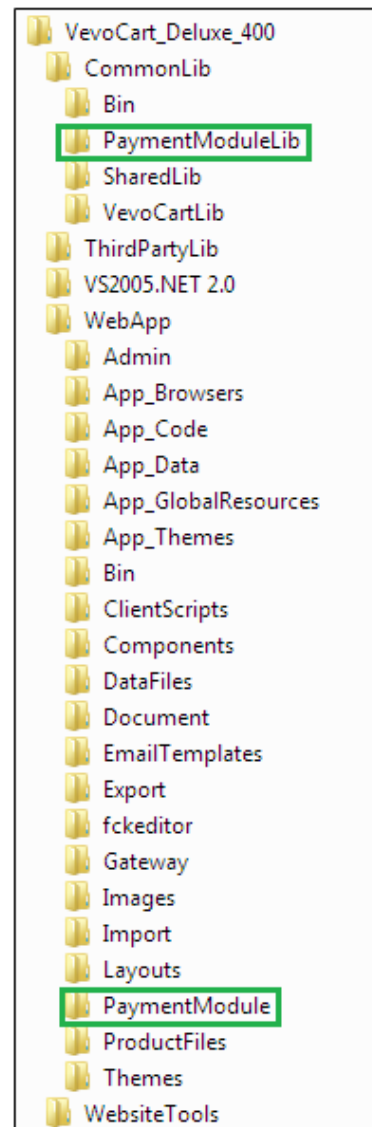
Chapter 8 outlines the necessary steps to add a new payment gateway to VevoCart.

2. Source Code Structure

2.1 Non-PCI Version

VevoCart 4.1 Non-PCI Version has separated source code that connects to payment gateways into a module called “Payment Module”.

- The Vevo Payment module resides in “PaymentModule” folder inside the main VevoCart website. The database tables that Payment Module uses begin with “vevopay_”.
- The Payment Module also has several libraries (DLL). The DLL file names start with “Vevo.PaymentApp”, which is also the namespace. Developers can find the source code of Payment Module DLL in “CommonLib\PaymentModuleLib” folder.



2.2 PCI Version

VevoCart 4.1 PCI Version has separate source code which connects to payment gateways into a new web application called “VevoPay”.

- The VevoPay solution resides in “VevoCart_xxx_PCI\VevoPay_100” folder where xxx is the VevoCart version (e.g. VevoCart_Deluxe_410_PCI).
- The VevoPay database should be separate from VevoCart database. And all of database table that begin with “vevopay_” are located in VevoPay database.
- The VevoPay also has several libraries (DLL). The DLL file names start with “Vevo.PaymentApp”, which is also the namespace. Developers can find the source code of VevoPay DLL in “VevoPayLib” folder.

3. Payment Module (VevoPay) Class Hierarchies

Payment Module libraries are inside “CommonLib\PaymentModuleLib” folder (“VevoPayLib” in PCI Version).

The list of all available payment options are inside “vevopay_PaymentOption” database table. For Payment Module, there is a class called *PaymentOption* in CommonLib\PaymentModuleLib\Vevo.PaymentApp.Domain\ folder (“VevoPayLib\Vevo.PaymentApp.Domain” in PCI Version). The objects of this class represent each row in the database.

The PaymentOption class has a method called “CreatePaymentMethod”. This method will create a new derived class of PaymentMethod class based on the class name set in “ProviderClassName” column in PaymentOption database.

Vevo Payment Module delegates the payment processing to payment method objects in Payment Module. The base class of all payments is *PaymentMethod*. This abstract class provides properties and methods to define the behavior of particular payment gateway.

You do not need to change PaymentOption class. However, you will need to add new rows in the database and create a derived class of PaymentMethod.

3.1 PaymentMethod

The class PaymentMethod is the base class of all payment classes. It has several abstract subclasses. These subclasses provide functionalities that you can inherit to create your own payment classes.

You usually do not inherit from this class directly. Your payment class will inherit from a sub-class, such as HostedPaymentMethod instead.

The class also defines some common functionality that can be overridden in the sub-classes. Here are some properties and methods that you may need to override.

Property Name	Description
Name	Inherit to return the Name of your payment option. This is the unique key for each payment option.

3.2 HostedPaymentMethod

This class represents the hosted payment solutions, such as PayPal Standard and 2Checkout.

An example of the concrete class of this type is *PayPalStandardPaymentMethod*, whose source code is inside
CommonLib\PaymentModuleLib\Vevo.PaymentApp.Payments\PayPalStandard\PayPalStandardPaymentMethod.cs.

This class defines new abstract methods that the sub-class must implement.

Method Name	Description
GetPostedUrl	Inherit to collect the Posted Url of your payment gateway.
CreatePostParameters	Inherit to collect the Parameters which you need to post to payment gateway.

3.3 OnWebsitePaymentMethod

This class represents to payment gateways that allow customers to enter credit card information on your own website (without switching to payment gateway's website), such as PayPal WebsitePayments Pro US, eWay, etc.

To implement this class, you will need to handle the credit card numbers and pass that information to payment gateways. The methods of information passing are different for each payment gateway. You will need to refer to your payment gateway manual for more information.

An example of the concrete class of this type is *AuthorizeNetPaymentMethod*, whose source code is inside
CommonLib\PaymentModuleLib\Vevo.PaymentApp.Payments\AuthorizeNet\AuthorizeNetPaymentMethod.cs.

This class defines new abstract properties and methods that the sub-class must implement.

Method Name	Description
ProcessPayment	<p>Inherit to process your payment to the payment gateway. This usually involves sending credit card information to the payment processor website. Return false if the credit card charge has failed.</p> <p>Also need to return the object of the type <i>ProcessPaymentResult</i>, which contains the error code and the log information that will be logged into PaymentLog database table of the main VevoCart.</p>

4. User Controls for Express Checkout Button

This section is only applicable for payment gateways that have their own checkout scenario. Some payment gateways, such as Google Checkout, allows user to bypass regular checkout process by clicking their Checkout Button. These payment methods usually inherit from *AnonymousPaymentMethod* base class.

The checkout button is implemented as a user control. It usually resides in WebApp\Gateway\Buttons folder. For example, Google Checkout button is implemented in WebApp\Gateway\Buttons\GoogleCheckoutButton.ascx.

This user control needs to inherit from a base class called *Vevo.WebUI.Payments.BaseButtonPaymentMethod*. The regular implementation would be transferring user to the payment gateway website, along with other information required by the payment gateway.

5. User Controls for Admin Configuration

By default, the Admin interface already provides basic configurations for payment methods, such as “Enabled” flag and payment image paths. However, each payment method has its own settings such as user name or access key. Therefore, each payment method needs a user control to allow additional settings to be set in Admin website.

In the PayPal Website Payments Standard configuration below, the highlighted section is specific for only PayPal Standard and created by the user control of this payment method (WebApp\Admin\Gateway\AdminPayPal.ascx), while the rest of the screens are standard for every payment methods.

Enabled

No

Display Name

PayPal Payment

+

(+)

Description

This is PayPal payment

(+)

Payment Image

Upload...

PayPal email account

Vevo@systems.com

Sandbox mode (Test mode)

☒

Update

5.1 Admin User Control Base Class

The user control needs to be inherited from the class *Vevo.AdminAdvancedBaseGatewayUserControl*. This base class provides two abstract methods that you need to override.

Method Name	Description
Refresh	This method will be called when the user control is loaded the first time. It should populate settings from database to text boxes, check boxes, etc.
Save	This method will be called when the user click "Update" button to save settings into the database. It should store the data to be used later during payment processing.

5.2 User Control Layout (.ascx File)

To display the first line in the highlighted area above (PayPal email account), you may enter the code similar to below:

```
<div class="CommonRowStyle">
  <div class="Label">
    <asp:Label ID="lcPayPalEmail" runat="server" meta:
resourcekey="lcPayPalEmail" />
  </div>
  <asp:TextBox ID="uxPayPalEmailText" runat="server"
    Width="250px" CssClass="fl TextBox" />
  <div class="Clear">
  </div>
</div>
```

5.3 User Control Code-Behind (ascx.cs File)

Usually, the values of payment gateway settings are stored in Configuration database table. In the code-behind file, you may load the configuration with the command such as:

```
public override void Refresh()
{
    ...
    uxPayPalEmailText.Text =
DataContext.Configurations.GetValue( "PaymentByPayPalEmail" );
    ...
}
```

Below is the example of the code to write values that merchant has entered back to database. The call to *System.Load()* function is necessary to ensure that the values that you have just saved will be used by the system.

```
public override void Save()
{
    DataContext.ConfigurationRepository.UpdateValue(
        DataContext.Configurations[ "PaymentByPayPalEmail" ],
        uxPayPalEmailText.Text );
    ...
    SystemConfig.Load();
}
```

6. Database Structure

To allow merchants to configure the payment method in the backend, the payment method information must be included in *PaymentOption*, *PaymentOptionLocale*, *vevopay_PaymentOption* database table.

6.1 PaymentOption Table

The structure of *PaymentOption* table is shown below. The items in **bold** can be edited from the Payment menu in the Admin website.

Column Name	Type	Description
Name	nvarchar(50)	The name of this payment method. This is the primary key of the table.
PaymentImage	nvarchar(100)	The path to the payment image that will be shown in the payment selection page. This is a relative path from the web application directory.
IsEnabled	Bit	Boolean value whether this payment is enabled.
Currencies	nvarchar(255)	The list of supported currencies of this payment method, separated by commas. The example is "USD, GBP" to support US Dollars and British Pounds. This is the list of currencies that will be shown in the payment gateway currency drop-down in the Admin Payment menu.
ShowPayment	bit	Flag whether this payment method will be shown in the payment selection page. <i>ShowPayment</i> and <i>ShowButton</i> flags should not be set to "True" at the same time.
ShowButton	bit	Flag whether this payment method will be shown as alternative checkout button in the shopping cart page. Currently, only Google Checkout and PayPal Express Checkout fit this category.
ButtonUserControl	nvarchar(255)	Path to user control for displaying checkout button. This field is necessary only if <i>ShowButton</i> flag is true.
AdminUserControl	nvarchar(255)	Path to user control to be used in Admin website. This user control is for allowing customers to enter additional configuration that is specific to the payment gateway. Preceding the path with "[Admin]" tag will put this user control in the folder for back office (e.g. Advanced).

SortOrder	int	The sort order of payment method when displaying in payment selection page during the checkout.
ProviderClassName	nvarchar(255)	<p>Set which Class that payment method should be called.</p> <ul style="list-style-type: none"> For <i>Hosted Solution</i> (similarly to PayPal Standard), enter "Vevo.Domain.Payments.RedirectPaymentMethod, Vevo.Domain" here. For <i>Integrated Solution</i> (similarly to PayPal Website Payments Pro), enter "Vevo.Domain.Payments.OnWebsitePaymentMethod, Vevo.Domain" here.
SystemUseOnly	bit	Flag whether this payment method is used internally by the system and will not be displayed for selection (e.g. ZeroPaymentMethod).
AuthenticationConfigs	nvarchar(MAX)	<p>The list of configurations that are required in this payment method, separated by commas. These configurations are the ones inside "Configuration" database table.</p> <p>For example: "LinkPointStoreNumber, LinkPointHost, LinkPointPort, LinkPointMode".</p>
CanUseRecurring	bit	Flag whether this payment method can process recurring billing. Currently, only PayPal Website Payments Pro can.
IsCreditCardNumberSaved	bit	Flag whether this payment method saves credit card number in to your database for later viewing in the Admin website. Currently, only Offline Credit Card Payment does this.
SupportedCreditCardValues	nvarchar(MAX)	<p>The value list of supported credit card of this payment method, separated by commas. One example is "Visa,MasterCard" to support Visa and Master Card.</p> <p>The values in this field may be different for each payment gateway requirements and implementations. For example, MasterCard in protX use "MC" but most other gateways use "MasterCard".</p>

SupportedCreditCards	nvarchar(MAX)	<p>The list of supported credit card of this payment method, separated by commas. One example is "Visa, MasterCard".</p> <p>This is the list of credit cards that will be shown in the credit card drop-down when the customer enters credit card information in the DirectPaymentSale page. This list should have the same order of credit cards as SupportedCreditCardValues.</p>
PaymentMethodSelectionAllowed	bit	Flag whether this payment method can be selected in the payment selection page. This flag should always be true, except ZeroPaymentMethod.
AnonymousCheckoutAllowed	bit	Flag whether this payment method is the anonymous checkout (e.g. PayPal Express, Google Checkout).
BillingAddressRequired	bit	Flag whether this payment method requires billing address
CreditCardRequired	bit	Flag whether this payment method needs credit card number enter on the store website.
ShownAsCreditCard	bit	If this flag is true, this payment method will be shown as "Credit Card" instead of payment method name in the Order Summary page and order confirmation email.
Cvv2Required	bit	Flag wheter this payment method requires CVV2 to be entered by the customers. This flag should be set to true only if CreditCardRequired field is set to true.

***Note:** Data fields in bold can be edited in the Payment menu of Admin website.

6.2 PaymentOptionLocale Table

The structure of *PaymentOptionLocale* table is shown below. The items in **bold** can be edited from the Payment menu in the Admin website.

Column Name	Type	Description
Name	nvarchar(50)	The name of this payment method. This is the one of primary key for this table. The value in this field must match the "Name" property in your payment class.
CultureID	int	The Culture ID of the language to be used with this data row. This is the one of primary key for this table.
DisplayName	nvarchar(255)	The name of this payment method. This is the payment name to be displayed in the payment selection page during the checkout.
Description	ntext	The description of this payment method. This description will be displayed in the payment selection page during the checkout.

***Note:** Data fields in bold can be edited in the Payment menu of Admin website

6.3 vevopay_PaymentOption Table

The structure of *vevopay_PaymentOption* table is shown below.

Column Name	Type	Description
Name	nvarchar(100)	The name of this payment method. This is the primary key of the table. The value in this field must match the "Name" property in your payment class.
Currencies	nvarchar(255)	The list of supported currencies of this payment method for validation purpose, separated by commas. The example is "USD, GBP" to support US Dollars and British Pounds. The value should be the same as "Currencies" field in <i>PaymentOption</i> table.
ProviderClassName	nvarchar(255)	The class name that will handle this payment method. The class must be inherited from PaymentMethod base class. For example,

		“Vevo.PaymentApp.PayPalProUK.PayPalProUKPaymentMethod, VevoPaymentApp” for PayPal Website Payments Pro UK or “Vevo.PaymentApp.PayPalStandard.PayPalStandardPaymentMethod, VevoPaymentApp” for PayPal Website Payments Standard.
--	--	--

6.4 SQL Script to Insert New Payment

One example of SQL Server script to insert PayPal Website Payment Standard into “PaymentOption”, “PaymentOptionLocale”, and “vevopay_PaymentOption” (For “vevopay_PaymentOption” table will be located at “VevoPay” database in PCI Version) database table would be:

```

INSERT INTO [PaymentOption]
([Name],[PaymentImage],[IsEnabled],[Currencies],[ShowPayment],[
ShowButton],[ButtonUserControl],[AdminUserControl],[SortOrder],[
ProviderClassName],[SystemUseOnly],[AuthenticationConfigs],[
CanUseRecurring],[IsCreditCardNumberSaved],[
SupportedCreditCardValues],[SupportedCreditCards],[
PaymentMethodSelectionAllowed],[AnonymousCheckoutAllowed],[
BillingAddressRequired],[CreditCardRequired],[ShownAsCreditCard]
,
[Cvv2Required])
VALUES(N'PayPal',N'',0,N'AUD,CAD,CHF,CZK,DKK,EUR,GBP,HKD,HUF,JPY,
NOK,NZD,PLN,SEK,SGD,USD',1,0,N'',
N'[Admin]/Gateway/AdminPayPal.ascx',5,
N'Vevo.Domain.Payments.RedirectPaymentMethod,
Vevo.Domain',0,N'PaymentByPayPalEmail,PaymentByPayPalEnvironment'
,
0,0,N'',N'',1,0,0,0,0,0);

INSERT INTO [paymentoptionlocale]
([Name],[CultureID],[DisplayName],[Description])
VALUES(N'PayPal',1,N'PayPal Payment',N'This is PayPal payment');

INSERT INTO [vevopay_PaymentOption]
([Name],[Currencies],[ProviderClassName])
VALUES(N'PayPal',
N'AUD,CAD,CHF,CZK,DKK,EUR,GBP,HKD,HUF,JPY,NOK,NZD,PLN,SEK,SGD,USD
',
N'Vevo.PaymentApp.PayPalStandard.PayPalStandardPaymentMethod,
Vevo.PaymentApp');

```

7. Configuration Parameters

As shown in previous chapters, the *PaymentOption* database table provides basic attributes of the payment method that the store will be used.

In addition, each payment gateway has its own set of parameters, such as user name, or access key. These additional parameters are normally saved in Configuration and ConfigurationValue database tables.

To create these configurations, it is advisable to put it in *Page_Load* event of the Admin User Control in chapter 5. You will need to check first whether this configuration already exists by calling *SystemConfig.ContainsKey* function.

```
private static void CreateDisplayConfiguration()
{
    CreateConfigConditionally(
        ValueType.Single,
        CultureIDs.English,
        "PaymentByPayPalEmail",
        "paypal@test.com",
        "Payment", "Payment", "The description here" );
}
```

This will create a configuration item called “PaymentByPalPalEmail” with the value of “paypal@test.com”. You only need to change these two values when calling *ConfigurationAccess.Create* function.

The last parameter is for description of this configuration item. However, at this moment, this description is not yet shown in the GUI.

8. How to Integrate a New Payment Gateway

For integrate new Payment Gateway to VevoCart 4.0. You will need a developer with C# and ASP.NET skills.

VevoCart 4.0 has separated payment functionality into its own module which is in the “PaymentModule” subfolder of VevoCart website. The database tables that begin with “vevopay_” are also part of Payment Module.

There are few places you need to modify integrate a new payment gateway. Below are the steps to add a new payment gateway.

- 1) You will need to first understand the flow of your payment gateway. There are two main solutions:
 - *Hosted Solution* is similar to PayPal Website Payments Standard. The customers will be redirected to the third party payment website and enter credit card information there.
 - *Integrated Solution* (also called On-Website in the source code) is similar to Authorize.Net AIM and PayPal Website Payments Pro. The customers will enter credit card information on your website instead of on the payment gateway website.

- 2) Add new entries in both “PaymentOption” and “PaymentOptionLocale” tables in the database. Depending on whether your payment gateway is similar to *Hosted Solution* (PayPal Standard) or *Integrated Solution* (PayPal Website Payments Pro US), you can copy the values from “PayPal” or “PayPal Pro US” entry respectively. You may see a sample SQL script in “Database Structure” (chapter 6) above.

Here are some important fields in PaymentOption table that needs to be set according to the payment gateway. For more information on how to fill these values, please see chapter 6 above.

- AdminUserControl
 - AuthenticationConfigs
 - ProviderClassName
 - Currencies
- 3) Add a new entry in the “vevopay_PaymentOption” database table. You will also need to create a new class corresponding to the value in “ProviderClassName” field. For example, if your payment name is “Newpay”, you may create a new class called “Vevo.PaymentApp.Payments.Newpay.NewpayPaymentMethod”. You can see the implementation of existing payment methods in “Vevo.PaymentApp.Payments” project when you open with the solution.
 - 4) Implement the new payment method class (e.g. NewpayPaymentMethod) from the step 3) above. You need to inherit from an existing base payment method class, depending on the type of your gateway solution.
 - If your payment gateway solution is *Hosted Solution* (similar to PayPal Standard), you can inherit your class from “Vevo.PaymentApp.Domain.HostedData.HostedPaymentMethod” class. Then, override the methods called “GetPostedUrl” and “CreatePostParameters” to process your payment.
 - If your payment gateway solution is *Integrated Solution* (similar to PayPal Website Payments Pro), you can inherit your class from “Vevo.PaymentApp.Domain.OnWebsiteData.OnWebsitePaymentMethod” class. Then, override the method called “ProcessPayment” to process your credit card information.