



WebMatrix Helpers

PayPal Helper

Contents

Introduction.....	1
Getting Started in 60 Seconds	2
More info.....	4
Helper Reference.....	5

Introduction

Microsoft WebMatrix provides an easy way to get started with web development, and together with new Razor syntax for ASP.NET Web Pages it includes everything you need to get your web site up, running and integrated with many other sites and networks, in a short period of time. The WebMatrix helpers are designed to make your life easier when creating web sites. They provide you a simple and consistent way of performing common web development tasks that otherwise would require a great deal of custom coding. With a few lines of code you should be able to secure your web site using membership, store information in Windows Azure Storage, integrate your site with Facebook, among others things.

The PayPal helper allows you to integrate PayPal payments within your WebMatrix website or e-commerce application. With a few lines of code, you'll enable your Web site customers to click on a payment button to pay for their purchases with their PayPal accounts.

The helper supports both the PayPal [Button Manager API](#) and the [Adaptive Payments API](#). Using the Button Manager API you will be able to create (and manage) PayPal buttons like *Add To Cart* or *Buy Now*, that will let your customers purchase single or multiple items, among other things. With the Adaptive Payments API, you can process from your WebMatrix web site PayPal transactions, in both simple and complex scenarios. You can build Web applications that handle payments, preapprovals for payments, and refunds.



Getting Started in 60 Seconds

The helper consists mainly of a **PayPal.dll** library, among other supporting libraries (paypal_base.dll, log4net.dll), located into the **Bin** folder of your WebMatrix site. To use the PayPal helper against the PayPal Sandbox follow these steps (the PayPal sandbox allows you to perform operations without real money exchange):

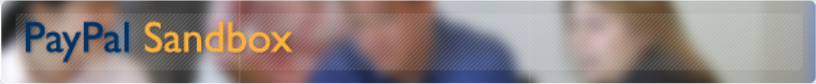
1. If you do not already have one, create an account in [PayPal Sandbox Test Environment](#).
2. Once logged into the Sandbox, enter the **API Credentials** section and make note of the following values:
 - a. API Username
 - b. API Password
 - c. Signature

Sandbox

- Home
- Test Accounts
- Test Email
- API Credentials**
- Test Tools

Additional resources

- Documentation
- PayPal Developer Network
- Customer Support



API Credentials

You must have credentials to test APIs for Website Payments Pro and Express Checkout in the Sandbox. In most cases, you will use API signatures and not download certificates.

The test accounts identified below are enabled for API access.

Note: These credentials will not work outside the Sandbox. You will need new credentials from paypal.com to go live.

Test Account	Date Created
Test Account: test_1287595157_bin@usfirstworks.net	Oct. 20, 2010 11:26:06 PDT
API Username: test_1287595157_bin_api@usfirstworks.net API Password: 1287595157 Signature: AaBmFwqSLA-1Wb0Cm0JF-gdPMA-0Gy7P8C0B8K0QK2ayCnUGSp	

3. Enter the **Test Accounts** section and create a PayPal Business account (to represent a seller) on the PayPal Sandbox. To do this, you can follow the instructions from this [PayPal Documentation page](#). Make note of the **Login e-mail** of the new Business account.

4. Add the following lines to the **_AppStart.cshtml** page of your WebMatrix site (you'll have to create if it doesn't exist) to initialize the Managed Buttons and Adaptive Payments APIs of the PayPal Helper. Replace the placeholders with your PayPal API credentials.

Razor

```
@{
    PayPal.Profile.Initialize(
        "{yourAPIUsername}",
        "{yourAPIPassword}",
        "{yourSignature}",
        "sandbox");

    // General Adaptive Payments' properties
    PayPal.Profile.Language = "en_US";
    PayPal.Profile.CancelUrl = "http://www.mystore.com/ohtoobad.cshtml";
    PayPal.Profile.ReturnUrl = "http://www.mystore.com/thanks.cshtml";
    PayPal.Profile.IpnUrl = "http://www.mystore.com/notifications.cshtml";
    PayPal.Profile.CurrencyCode = "USD";
}
```

5. Add the highlighted lines from below in the page where you want to show a **PayPal Add To Cart** button. The `"{sellerEmail}"` placeholder should be changed by the e-mail of the seller account you've configured previously. **Notice that this code is for sample purposes only and will create a PayPal button each time the page is rendered;** on your implementation you can store the button's *WebSiteCode* value in a database.

Razor

```
@{
    var paypalButton = PayPal.ButtonManager.AddToCartButton.Create(
        business : "{sellerEmail}",
        itemName : "My Product",
        amount : "99.99");

    HtmlString paypalButtonHtml = new HtmlString(paypalButton.WebSiteCode);
}
<!DOCTYPE html>
<html>

    <head>
        ...
    </head>
    <body>
        ...
        @paypalButtonHtml
        ...
    </body>
</html>
```

Now, if you login to PayPal (remember to use your Sandbox account if you have been using that) and click the **Profile** link and then **My Saved Buttons** under **Selling Preferences**, you will be able to see the button you have just created.

6. Now let's test the Adaptive Payments API, by executing a chained payment. Suppose that you have a list of suppliers that you want to pay whenever an item is sold through your online application. Create a new web page and add the following highlighted code. Replace the "{storeEmail}" and "{supplierEmail}" placeholders with two different PayPal sandbox (or real) accounts.

Razor

```
@{
    var productPrice = 99.99m;
    var supplierAmount = decimal.Round(productPrice * .10m, 2);

    var MyStore = new PayPal.AdaptivePayments.Receiver();
    MyStore.amount = productPrice;
    MyStore.email = "{storeEmail}";

    var Suppliers = new List<PayPal.AdaptivePayments.Receiver>();
    var Supplier = new PayPal.AdaptivePayments.Receiver();
    Supplier.email = "{supplierEmail}";
    Supplier.amount = supplierAmount;
    Suppliers.Add(Supplier);

    var response = PayPal.AdaptivePayments.ChainedPay.Execute(MyStore,
Suppliers, "", "Test Payment", "127.0.0.1", "MyDevice");

    response.Redirect();
}

<!DOCTYPE html>
<html>

    <head>
        ...
    </head>
    <body>
        ...
    </body>
</html>
```

You prepare the payment by setting up a single instance of a "receiver" which is the store that is receiving the payment. After that, you setup a list of receivers, which represents our suppliers (or other people that we need to pay after the transaction); each one will receive 10% of the product price.

More info

Learn more on WebMatrix, ASP.NET Web Pages and the Razor Syntax with [the WebMatrix tutorials](#).

PayPal Documentation and Manuals can be found on the [PayPal X Developer Network](#).

Helper Reference

Button Manager API

The PayPal Button Manager API support of the PayPal Helper lets you programmatically create, manage, edit, and delete Website Payment Standard buttons, which are the same kind of buttons that you can create from the PayPal Profile. Hosted buttons created by this API reside on PayPal and can use all features, including inventory management.

You can use the Helper to create the following types of buttons:

- Buy Now: Sell single items in various quantities
- Add to Cart: Sell multiple items at one time
- Donate: Accept monetary contributions
- Subscribe: Sign people up for recurring payments

All the calls to the Button Manager API, should be preceded by the **PayPal.ButtonManager** namespace.

Method	Description
static PayPal.Profile.Initialize (string apiUsername, string apiPassword, string apiSignature, string environment, [string applicationId])	Initializes the PayPal both for using the Button Manager API and the Adaptive Payments API Helper. The environment can be 'sandbox' or 'production'. If applicationId is not provided, the helper will use a default testing value provided by PayPal (ONLY for testing purposes).
static ButtonManegerResponse AddToCartButton.Create (string Business, string ItemName, string Amount)	Creates an Add To Cart button for one single item with the minimum number of parameters.
static ButtonManegerResponse AddToCartButton.Create (HtmlButtonVariables buttonVariables, [ButtonOption[] buttonOptions], [string buttonImage], [string buttonImageUrl], [string buttonSubType], [string buttonCountry], [string buttonLanguage])	Creates a fully configurable Add To Cart button.
static ButtonManegerResponse AddToCartButton.Update (String hostedButtonId, HtmlButtonVariables buttonVariables, ButtonOption[] buttonOptions)	Updates an Add To Cart button.
static void AddToCartButton.Delete (String hostedButtonId)	Deletes an Add To Cart button.
static ButtonManegerResponse BuyNowButton.Create (string Business, string ItemName, string Amount)	Creates a Buy Now button for one single item with the minimum number of parameters.
static ButtonManegerResponse BuyNowButton.Create (HtmlButtonVariables buttonVariables, [ButtonOption[] buttonOptions], [string buttonImage], [string buttonImageUrl], [string buttonSubType], [string buttonCountry], [string	Creates a fully configurable Buy Now button.

buttonLanguage])	
static ButtonManagerResponse BuyNowButton.Update (String hostedButtonId, HtmlButtonVariables buttonVariables, ButtonOption[] buttonOptions)	Updates a Buy Now button.
static void BuyNowButton.Delete (String hostedButtonId)	Deletes a Buy Now button.
static ButtonManagerResponse DonateButton.Create (string Business, string ItemName, [string Amount])	Creates a Donate button with the minimum number of parameters.
public static ButtonManagerResponse DonateButton.Create (HtmlButtonVariables buttonVariables, [ButtonOption[] buttonOptions], [string buttonImage], [string buttonImageUrl], [string buttonSubType], [string buttonCountry], [string buttonLanguage])	Creates a fully configurable Donate button.
static ButtonManagerResponse DonateButton.Update (String hostedButtonId, HtmlButtonVariables buttonVariables, ButtonOption[] buttonOptions)	Updates a Donate button.
static void DonateButton.Delete (String hostedButtonId)	Deletes a Donate button.
static ButtonManagerResponse SubscribeButton.Create (string Business, string ItemName, a3, p3, t3)	Creates a Subscribe button with the minimum number of parameters.
static ButtonManagerResponse SubscribeButton.Create (HtmlButtonVariables buttonVariables, [ButtonOption[] buttonOptions], [string buttonImage], [string buttonImageUrl], [string buttonSubType], [string buttonCountry], [string buttonLanguage])	Creates a fully configurable Subscribe button.
static ButtonManagerResponse SubscribeButton.Update (String hostedButtonId, HtmlButtonVariables buttonVariables, ButtonOption[] buttonOptions)	Updates a Subscribe button.
static void SubscribeButton.Delete (String hostedButtonId)	Deletes a Subscribe button.

Type	Description
ButtonManagerResponse	<p>After you've executed the create button methods, you can retrieve its data from the properties of this object:</p> <ul style="list-style-type: none"> • WebSiteCode: The button HTML code to render on your Web page. • EmailLink: Code for email links and links in other documents that support external links. • HostedButtonId: The ID of the button.
HtmlButtonVariables	<p>Html Button Variables specify information about the product or service for Buy Now and Add to Cart buttons, or they specify information about a contribution for Donate buttons. They also control how PayPal responds when people click the buttons and how they interact with special PayPal features.</p> <p>For a complete documentation of the variables you can use to configure the PayPal</p>

	Buttons see the following link: https://cms.paypal.com/us/cgi-bin/?cmd=_render-content&content_ID=developer/e_howto_html_Appx_websitestandard_htmlvariables
ButtonOptions	Using this parameter you can specify different options for the product or service, which will be displayed as a combo box next to the PayPal button. For each option that you create you should specify a name and a price.

The following images showcases how you should specify the **HtmlButtonVariables** and the **ButtonOptions** in an AddToCart button. This button includes menus specified with the ButtonOptions that allow you to specify the color and size; each color is associated with a specific price.



```

BMResponse resp = AddToCartButton.Create(
    new HtmlButtonVariables {
        Business = "msft_1284494024_biz@yahoo.com",
        ItemName = "Wireless Mouse",
        Amount = "199.99",
        CurrencyCode = "USD"
    },
    ButtonOptions.Create(
        ButtonOption.Create("color", ButtonOptionItems.Create(
            ButtonOptionItem.Create("Red", "11.50"),
            ButtonOptionItem.Create("Blue", "12.50")
        )),
        ButtonOption.Create("size", ButtonOptionItems.Create(
            ButtonOptionItem.Create("Small", "11.50"),
            ButtonOptionItem.Create("Medium", "12.50")
        ))
    )
);
  
```

For more information on the Button Manager API see the following articles:

- Website Payments Standard Button Manager API Overview: <https://www.x.com/docs/DOC-1108>
- NVP Button Manager API Administration Guide: https://cms.paypal.com/cms_content/US/en_US/files/developer/PP_WPSButtonMgrAPINV P.pdf

Adaptive Payments API

PayPal Adaptive Payments API support in the PayPal helper provides several methods, enabling you to build an application that handles payments, preapprovals for payments, and refunds. You can also retrieve Foreign Exchange conversion rates for a list of amounts. Adaptive Payments provides several methods of payment:

- **Simple Payments:** Simple payments enable a sender to send a single payment to a single receiver. This is the traditional way that payments are made. For example, there is the obvious scenario of a one-to-one transaction such as a simple shopping cart application with a single seller.
- **Parallel Payments:** Parallel payments enable a sender to send a single payment to multiple receivers. For example, your application might be a shopping cart that enables a buyer to pay for items from several merchants with one payment. Your shopping cart allocates the payment to merchants that actually provided the items. PayPal then deducts money from the sender's account and deposits it in the receivers' accounts.

- **Chained payments:** Chained payments enable a sender to send a single payment to a primary receiver; the primary receiver keeps part of the payment and pays secondary receivers the remainder. For example, your application could be an online travel agency that handles bookings for airfare, hotel reservations, and car rentals. The sender sees only you as the primary receiver. You allocate the payment for your commission and the actual cost of services provided by other receivers. PayPal then deducts money from the sender's account and deposits it in both your account and the secondary receivers' accounts.

Additionally there are three kinds of payment approvals:

- **Explicit approval payments,** in which the sender logs into paypal.com to approve each payment. Explicitly approving payments is the traditional way to pay with PayPal. This method is the only option unless the sender has set up a preapproval agreement or you, the API caller, are also the sender.
- **Preapproved payments,** in which a sender logs into PayPal and sets up preapprovals that approve future payments; for example, for a specific vendor. The sender logs into paypal.com once to set up the preapproval. After the preapproval is set up, payments are considered approved, and specific approval is unnecessary.
- **Implicit approval payments,** in which your application is both the sender of a payment and the caller of the Adaptive Payments Pay API. In this case, the payment is drawn from your own account, which eliminates the need for approval.

All the calls to the Adaptive Payments API, should be preceded by the **PayPal.AdaptivePayments** namespace.

Property	Description
static string PayPal.Profile.Language	The RFC 3066 language in which error messages are returned; by default it is en_US.
static string PayPal.Profile.CancelUrl	The URL to which the sender's browser is redirected if the sender cancels the approval for the payment after logging in to paypal.com to approve the payment. Specify the URL with the HTTP or HTTPS. Maximum length: 1024 characters.
static string PayPal.Profile.ReturnUrl	The URL to which the sender's browser is redirected after approving a payment on paypal.com. Specify the URL with the HTTP or HTTPS designator. Maximum length: 1024 characters.
static string PayPal.Profile.IpnUrl	The URL to which you want all IPN messages for the payments to be sent. Maximum length: 1024 characters.
static string PayPal.Profile.CurrencyCode	The code of the currency used for the

	payments. Check the Adaptive Payments Guide for a list of supported currency codes.
--	---

Method	Description
static PayPal.Profile.Initialize (string apiUsername, string apiPassword, string apiSignature, string environment, [string applicationId])	Initializes the PayPal both for using the Button Manager API and the Adaptive Payments API Helper. The environment can be 'sandbox' or 'production'. If applicationId is not provided, the helper will use a default testing value provided by PayPal (ONLY for testing purposes).
static PayResponse ChainedPay.Execute (Receiver primaryReceiver, List<Receiver> secondaryReceivers, string senderEmail, string memo, string userIp, string deviceId, [string cancelUrl], [string returnUrl], [string ipnUrl], [string currencyCode], [string language])	Executes an explicit chained pay.
static ConvertCurrencyResponse ConvertCurrency.Execute (CurrencyType[] baseAmountList, string[] convertToCurrencyList, [string language])	Executes a ConvertCurrency operation to request the current foreign exchange (FX) rate for a specific amount and currency.
static PayResponse ImplicitChainedPay.Execute (Receiver primaryReceiver, List<Receiver> secondaryReceivers, string memo, string userIp, string deviceId, [ipnUrl], [currencyCode], [string language])	Executes an implicit chained pay.
static PayResponse ImplicitParallelPay.Execute (List<Receiver> receivers, string memo, string userIp, string deviceId, [string ipnUrl], [string currencyCode], [string language])	Executes an implicit parallel pay.
static PayResponse ImplicitSimplePay.Execute (string receiverEmail, decimal amount, string memo, string userIp, string deviceId, [string ipnUrl], [string currencyCode], [string language])	Executes an implicit simple pay.
static PayResponse ParallelPay.Execute (List<Receiver> receivers, string senderEmail, string memo, string userIp, string deviceId, [string cancelUrl], [string returnUrl], [string ipnUrl], [string currencyCode], [string language])	Executes an explicit parallel pay.
static PayResponse PreapprovedChainedPay.Execute (Receiver primaryReceiver, string preapprovalkey, List<Receiver> secondaryReceivers, string senderEmail, string memo, string userIp, string deviceId, [string ipnUrl], [string currencyCode], [string language])	Executes a preapproved chained pay.
static PayResponse PreapprovedParallelPay.Execute (List<Receiver> receivers, string preapprovalkey, string senderEmail, string memo, string userIp, string deviceId, [string ipnUrl], [string currencyCode], [string language])	Executes a preapproved parallel pay.
static PayResponse PreapprovedSimplePay.Execute (string	Executes a preapproved simple pay.

receiverEmail, decimal amount, string preapprovalKey ,string senderEmail, string memo, string userIp, string deviceId,[string ipnUrl],[string currencyCode],[string language])	
static RefundResponse RefundCompletePayment.Execute (string payKey,string trackingId,[string language])	Executes a refund operation to refund an entire payment.
static RefundResponse RefundPartialPayment.Execute (Receiver[] receivers,string payKey,string trackingId,[string currencyCode],[string language])	Executes a refund operation to refund a payment to certain specific receivers. In a refund, the terms sender and receiver refer to sender and receivers of the original payment. When making a refund, the sender's account receives the refund and the receivers' accounts are the source of the refund. Refunds are made from one or more receivers to a sender.
static RefundResponse RefundTransaction.Execute (string transactionId,[string language])	Executes a refund operation to refund a specific transaction.
static PayResponse SimplePay.Execute (string receiverEmail,decimal amount,string senderEmail,string memo,string userIp,string deviceId,[string cancelUrl],[string returnUrl],[string ipnUrl],[string currencyCode],[string language])	Executes an explicit simple pay.
static PreapprovalResponse SimplePreapproval.Execute (string senderEmail, DateTime startingDate,decimal maxTotalAmountForAllPayments, decimal maxAmountPerPayment, int maxNumberOfPayments, DateTime endingDate, string memo, string userIp, string deviceId,[string cancelUrl],[string returnUrl],[string ipnUrl],[string currencyCode],[string language])	Executes a preapproval operation to set up an agreement between yourself and a sender for making payments on the sender's behalf. You set up a preapprovals for a specific maximum amount over a specific period of time and, optionally, by any of the following constraints: the number of payments, a maximum per payment amount, for each payment request

For more information on PayPal Adaptive Payments API see the following articles:

- Adaptive Payments Guide: https://cms.paypal.com/cms_content/US/en_US/files/developer/PP_AdaptivePayments.pdf
- How To Make a Parallel Payment Using NVP: <https://www.x.com/people/travis/blog/2010/08/12/how-to-make-a-parallel-payment-using-nvp>
- How To Make a Chained Payment Using NVP: <https://www.x.com/people/travis/blog/2010/08/17/how-to-make-a-chained-payment-using-nvp>
- 3 Types of Adaptive Payments: <https://www.x.com/people/travis/blog/2010/08/11/3-types-of-adaptive-payments>