



[Web Development](#) » [ASP.NET](#) » [General](#)

License: [The Code Project Open License \(CPOL\)](#)

C#, .NET (.NET 2.0, .NET 3.0, .NET 3.5), ASP.NET, Architect, Dev

Introduction to PayPal for C# - ASP.NET developers



By [Predrag Tomasevic](#)

Posted: **6 Oct 2009**

Views: **678**

Bookmarked: **10 times**

Overview that presents all PayPal integration options, targeted especially at C# developers

2 votes for this article.  Popularity: 1.38 Rating: **4.60** out of 5  1 2 3 4 5

[Download source - 170 KB](#)

Index

- [PayPal Introduction](#)
- [Getting started with PayPal](#)
- [Setting up test account](#)
- [Website Payments Standard \(HTML\)](#)
- [PostPayment processing](#)
 - [AutoReturn](#)
 - [Direct Payment \(PDT\)](#)
 - [Instant Payment Notification \(IPN\)](#)
- [PayPal API](#)
 - [Express Checkout](#)
 - [Direct Payment \(Website Payments Pro\)](#)
- [Conclusion](#)
- [History](#)

Introduction

PayPal is probably one of the first things that gets mentioned once discussion on online payments starts. It's not so without reason – in 2008 PayPal moved over 60 billion dollars between accounts which is, you'll agree, respectable amount. And also, all trends show that this growth will continue – with huge number of new accounts (over 184 million accounts in 2008 compared to 96.2 million in 2005), with new platform named [PayPal X](#) and with more cool applications that involve paying (like [Twitpay](#)) you can bet that PayPal is here to stay. So, how can you join whole PayPal Development movement?

Unfortunately, I would say – not so easily. When I first started with PayPal integration - it was hard, really hard. If you wish to see what I mean, just jump to [PayPal Developer Center](#). There is no way you'll easily fish out what you need from that site if you are PayPal newbie; simply - there are too many links, too many resources and too many mixings of important and not-so-important information. So, how should you start?

Getting started with PayPal

To those who really want to get into PayPal and are willing to shell out some buck, I would recommend [Pro PayPal E-Commerce](#) book - that's how I eventually got into understanding concepts behind PayPal integration. For those who are not so eager to pay – don't worry, that's why this article is here... I'll go over most of the stuff that book covers, but in more brief and concise manner.

First and foremost - understanding what kinds of integration PayPal offers is, I would say, most important thing in order to successfully start your development journey. Common mistake, that happened to me also, is to start at once with PayPal API and Express Checkout. I mean it's natural - we are developers and when they tell us to integrate with something, the first thing we look for is SDK & API... PayPal API comes up as a result... we say "That's it" to ourselves... and start working. The problem is – majority of payment scenarios can be handled with way simpler approach - HTML forms that are part of Website Payments Standard.

So, without further ado, here is classification of PayPal integrations:

- Website Payments Standard (HTML)
- Postpayment Processing
 - AutoReturn
 - Payment Data Transfer (PDT)
 - Instant Payment Notification (IPN)
- PayPal API
 - Express Checkout
 - Direct Payment (Website Payments Pro)
- Payflow Gateway

Items in classification are also ordered in a way I would suggest for everyone to follow. So, if you are new to PayPal – first learn all of the options that you have with Website Payments Standard (HTML). Then, if you need to add some basic post-payment processing, see if Auto-Return or PDT will solve your problem... if not, IPN is more robust option you have at your disposal.

Next level would involve PayPal API and implementing Express Checkout, which is the most flexible PayPal integration solution. And finally, if you long for ability to directly process credit cards on your website, you'll pay monthly fee to PayPal and implement Direct Payment (effectively getting what is called Website Payments Pro).

Last item from our classification - Payflow Gateway is, on the other hand, different beast. It doesn't "update the stack" in a way previously mentioned technologies do. It is solution aimed specifically at those businesses that have/want Internet Merchant Account (IMA) and just need payment gateway. In order to keep article consistent I'll skip explaining details of Payflow Gateway. However, if you have any question related to it, feel free to leave message in comments and I'll try to answer.

That said; let's get to setting up testing PayPal account and then we'll delve deeper into describing mentioned integrations.

Setting up test account

Word of notice – you'll want to follow this step even if you already have live PayPal account. There are two reasons for using test accounts:

- you don't want to test and play with real money
- you want to have access to different [types of PayPal accounts](#)
 - Personal account – most people have these; just account that allows you to use PayPal when paying for stuff online. Theoretically, you can use Personal account to accept money; just know that you'll be severely constrained – there is \$500 receiving limit per month and you are only able to accept one time payments using Website Payments Standard (HTML). The big advantage of Personal account is that you don't need to pay any transaction fee when receiving money. Note however, that if you receive more than \$500 in one month you'll be prompted to either upgrade to Premier/Business account or reject payment.
 - Premier account – step up from personal account; for anyone that wants to run personal online business. This type of account has all of the integration options (accepting credit cards, recurring payments, PayPal API). However, most people skip directly from Personal to Business account as Premier account has same [transaction fees](#) (in most cases 2.9% + \$0.30 per transaction) while lacking reporting, multi-user access and other advanced merchant services of Business account.
 - Business account – it has all of the features of Premier account plus few more (ability to operate under your business's name is one of them). If you are developing website that needs to accept payments in 99% of situations you'll go with this type of account.

To start, visit PayPal Sandbox and sign-up for new account. The process is straightforward and most developers should have no trouble finishing it. However, here are the pictures that will help you navigate through process:



Signing up for sandbox account



Filling in details of your sandbox account

Once done with entering details for your sandbox account, you'll need to check email you provided in order to complete registration. After that, you'll be able to login and start creating sandbox PayPal accounts. Clicking on **Test Accounts** (menu on the left), and then Create Account: **Preconfigured** - will get you form like the one on image below:



Creating a Sandbox Test Account

Clarification of Account Type radio buttons: by selecting Buyer you'll create Personal account and by selecting Seller you'll create Business account. For testing most integration scenarios you'll need both accounts so be sure to create them. Here is what you should eventually have on your screen after you click on **Test Accounts**.



Overview of your testing accounts

Checking radio button next to any of the accounts from list and clicking on Enter Sandbox Test Site should bring up Sandbox PayPal site which will allow you to login and administer your account in same way as with regular PayPal account. The only difference is that you'll have huge PayPal Sandbox header and text that displays email address of your developer account. To see what I'm talking about, check image below:



Administering PayPal Sandbox account

Last but not least - in order to use your Sandbox account for testing you need to be logged in with your [developer account](#). If you are not logged in and you follow some payment link you'll get following screen:



Login to use the PayPal Sandbox features

Website Payments Standard (HTML)

In this section I'll provide you with number of examples that will show how to create your own HTML form for receiving money over PayPal. You'll see how to use different variables in order to influence payment details. Before we delve into details, let's take a look at two most basic variables:

- form's action attribute - in most cases it should be **https://www.paypal.com/cgi-bin/webscr**. If you are using Sandbox for testing payments you'll change it to **https://www.sandbox.paypal.com/cgi-bin/webscr** - effectively you just insert word **sandbox** into url (this is also true for some other integrations; e.g. PayPal API). For coming examples, I won't be using Sandbox url because most of you would just get that "Login to use the PayPal Sandbox features" screen (look up for image).
- form's business child - I'll use **youremailaddress@yourdomain.com** for most examples; if you copy-paste code, you'll want to replace that with email of your PayPal account.

Basic payment

OK, let's say you have opened PayPal account and you just wish to be able to accept \$10 payment for painting you are selling through your site. Just insert following HTML into your page and you are set to go:

```
<form action="https://www.paypal.com/cgi-bin/webscr" method="post">
  <input type="hidden" name="cmd" value="_xclick" />
  <input type="hidden" name="business" value="youremailaddress@yourdomain.com" />
  <input type="hidden" name="item_name" value="My painting" />
  <input type="hidden" name="amount" value="10.00" />
  <input type="submit" value="Buy!" />
</form>
```

Shipping & handling

Next thing that comes to mind is that you'll wish to add shipping and/or handling fees to your form. It's easy - just add more parameters:

```
<form action="https://www.paypal.com/cgi-bin/webscr" method="post">
  <input type="hidden" name="cmd" value="_xclick" />
  <input type="hidden" name="business" value="youremailaddress@yourdomain.com" />
  <input type="hidden" name="item_name" value="My painting" />
  <input type="hidden" name="amount" value="10.00" />
  <input type="hidden" name="shipping" value="3.00" />
  <input type="hidden" name="handling" value="2.00" />
  <input type="submit" value="Buy with additional parameters!" />
</form>
```

Donations & Textual links

If you aren't selling anything but rather accepting donations for some cause - you'll just need to change value of **cmd** variable to **_donations**. If we combine this with common requirement to have hyperlink instead of button - we get following URL (of course you can use this method of URL creation for other PayPal payment types):

```
https://www.paypal.com/cgi-bin/webscr?
cmd=_donations&business=youremailaddress@yourdomain.com&item_name=Save Polar Bears!
&amount=10.00
```

RESULT:

[Save Polar Bears!](https://www.paypal.com/cgi-bin/webscr?cmd=_donations&business=youremailaddress@yourdomain.com&item_name=Save Polar Bears!&amount=10.00)

Cart system

If you have bunch different products to offer and you just want simple cart system without implementing anything PayPal has you covered. Basically, you'll just play with cmd variable while keeping rest of the form same as for Basic Payment. Let's see how you should do this for two products; one priced at \$10 without shipping fees and one priced at \$5 with \$1 shipping fee. We will also need View Cart button:

```
My Cart Item 1:
<form action="https://www.paypal.com/cgi-bin/webscr" method="post" target="paypal">
  <input type="hidden" name="cmd" value="_cart">
  <input type="hidden" name="add" value="1">
  <input type="hidden" name="business" value="youremailaddress@yourdomain.com">
  <input type="hidden" name="item_name" value="My Cart Item 1">
  <input type="hidden" name="amount" value="10.00">
  <input type="hidden" name="shopping_url" value="http://www.yourwebsite.com/shoppingpage.html">
  <input type="hidden" name="return" value="http://www.yourwebsite.com/success.html">
```

```



```

Recurring payments

If you are selling monthly service rather than product, you'll be interesting in recurring payment options PayPal provides. Again, it's playing with different variables that have different meaning. Let's say you wish to set 3 day free trial after which user will have to pay you \$10.00 per month to keep using service. Following HTML form should do the trick:

```

<form action="https://www.paypal.com/cgi-bin/webscr" method="post">
  <input type="hidden" name="cmd" value="_xclick-subscriptions"/>
  <input type="hidden" name="business" value="youremailaddress@yourdomain.com"/>
  <input type="hidden" name="item_name" value="Something"/>
  <input type="submit" value="Subscribe!" />

  <input type="hidden" name="a1" value="0"/>
  <input type="hidden" name="p1" value="3"/>
  <input type="hidden" name="t1" value="D"/>
  <input type="hidden" name="a3" value="10.00"/>
  <input type="hidden" name="p3" value="1"/>
  <input type="hidden" name="t3" value="M"/>
  <input type="hidden" name="src" value="1"/>
  <input type="hidden" name="srt" value="0"/>
  <input type="hidden" name="sra" value="1"/>
</form>

```

HTML variables & Resources

After reading previous example you may be wondering what certain variables do (a1? p1? srt?). Luckily, PayPal provides "[HTML Variables for Website Payments Standard](#)" page on which you can read about any variable that you are interested in.

Also, another great resource (to which I wish someone pointed me when I first started with PP integration) is "[skier's PayPal examples](#)". You'll find example for almost any payment scenario you can think of - so instead of cluttering this article with more examples, I'll encourage you to visit that page should you wish to implement some more advanced PayPal HTML form.

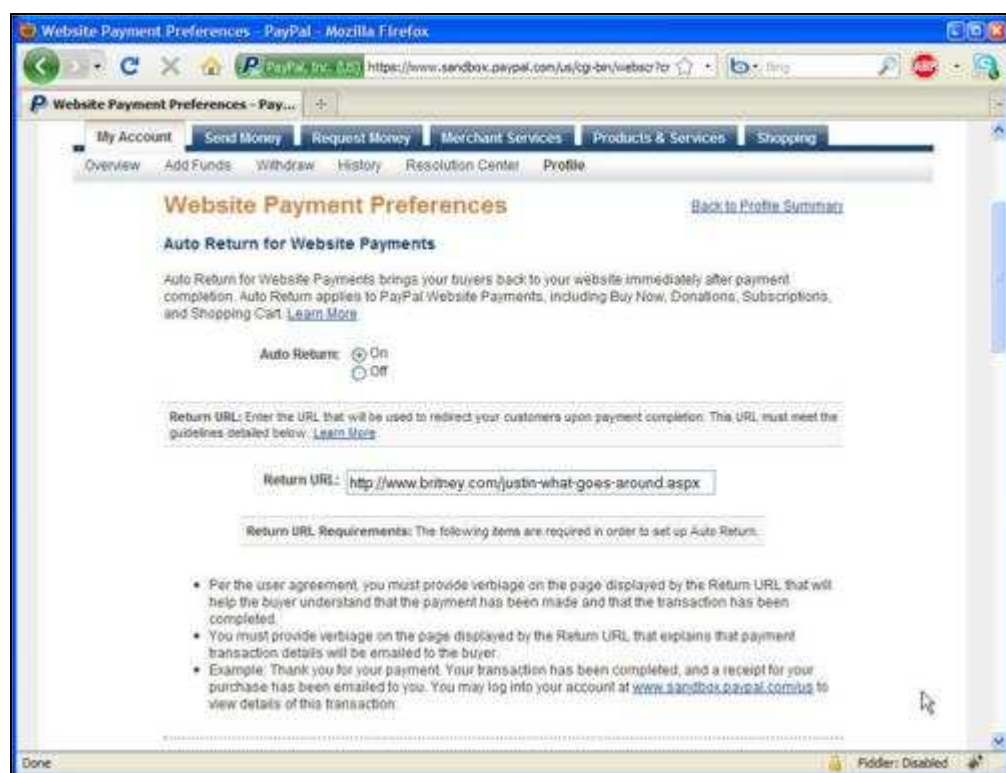
PostPayment processing

When you setup your PayPal HTML form the first question you'll probably ask is - after user pays can I have some post-payments processing logic? It is not so without reason; there are numerous post-payments scenarios we can think of - from sending simple "Thank you" email to updating site database and allowing user access to restricted resources for which he paid. Depending on your knowledge and desired level of robustness for post-processing logic, there are three ways you can go; and the good thing is you can combine them.

AutoReturn

AutoReturn is the simplest PostPayment processing solution that you have - after user pays he is *automatically** redirected to specified page on your website on which you can display some confirmation text. If you carefully went through "HTML Variables for Website Payments Standard" you know that you can use **return** variable to specify AutoReturn url in HTML form. If you wish to have default AutoReturn url, follow these steps:

1. Log in to your Premier or Business account
2. Click the Profile subtab
3. Click Website Payment Preferences in the Selling Preferences column
4. Click the On radio button next to the Auto Return label
5. Enter the URL where you want your users to return in the text box labeled Return URL
6. Click the Save button at the bottom of the page



Providing AutoReturl url in your PayPal profile

Know that if you have both AutoReturn url in your profile and provide **return** variable in your HTML form, the **return** variable will overwrite profile url value.

Now, when your return page is hit, you'll be getting variables that should allow you to customize page display and log payment:

- tx - Transaction ID
- st - Payment status
- amt - Payment amount
- cc - Currency code

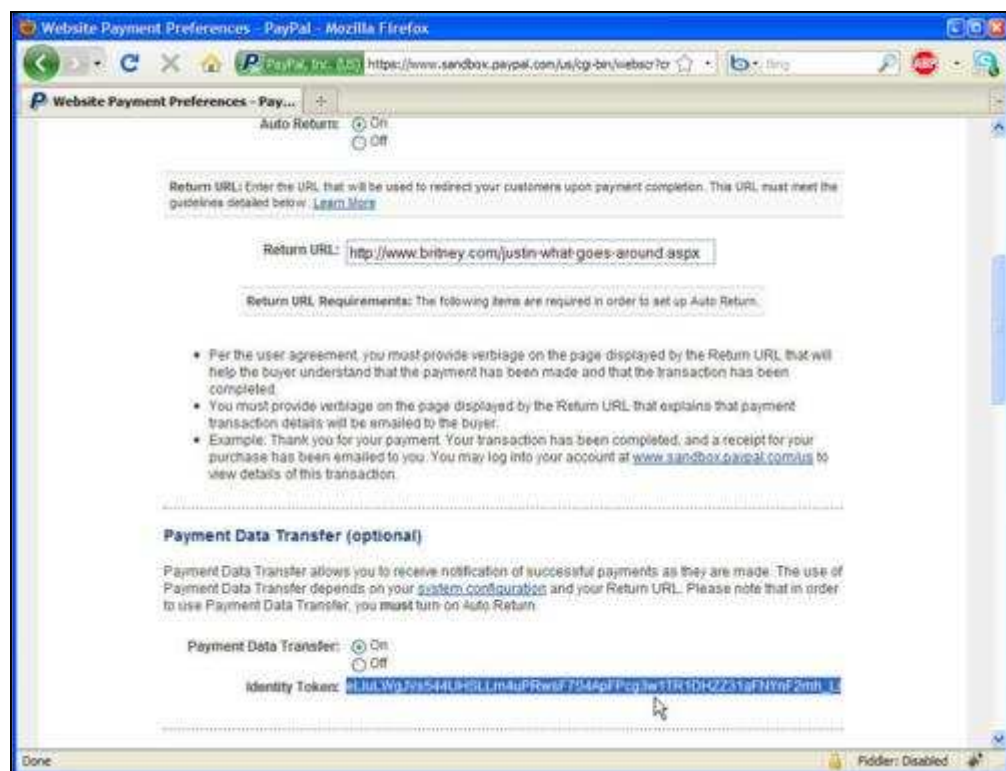
Before closing this section, one more thing. The reason why I've italicized word "automatically" in first sentence is: if user uses credit card to pay you, he won't be automatically redirected to your return url; he'll rather need to click on "Return to Merchant" button. If this sounds weird to you [know that you're not alone](#); however, because of legal issues PayPal refused and still refuses to change the way credit card payments are handled with AutoReturn.

Payment Data Transfer (PDT)

After looking over the list of variables that AutoReturn provides, you probably wondered - can I get more details about transaction that occurred? This is exactly where PDT jumps in - building on AutoReturn functionality. For that reason, you'll need to enable both AutoReturn and then PDT in your profile; here is how to do that:

1. Log in to your Premier or Business account
2. Click the Profile sub tab
3. Click Website Payment Preferences in the Selling Preferences column
4. Click the On radio button next to the Auto Return label
5. Enter the URL of the script that will process the PDT HTTP request sent from PayPal
6. Under Payment Data Transfer, click the On radio button
7. Click Save.

After following these steps, you should get PDT Identity Token that is needed for querying PayPal. If you don't copy-paste token after clicking Save, know that you can always see it in your Website Payment Preferences:



Payment Data Transfer and Identity Token

Now that you have Identity Token you can query PayPal for more details after your return url has been hit. Here is how things flow when utilizing PDT:

1. User pays and is redirected to your AutoReturn page, for example:
`http://www.yourdomain.com/Thanks.aspx?tx=[TransactionID]`
2. From code-behind of Thanks.aspx you'll parse tx value and make HTTP POST to **`https://www.paypal.com/cgi-bin/webscr`** with following parameters: **`cmd=_notify-synch&tx=[TransactionID]&at=[PDTIdentityToken]`**. (If you are using Sandbox you'll of course make HTTP POST to **`https://www.sandbox.paypal.com/cgi-bin/webscr`**)
3. PayPal will respond to your HTTP POST in following format:
SUCCESS
`first_name=Firstname`
`last_name=Lastname`
`payment_status=Completed`
`payer_email=firstname%40lastname.com`
`payment_gross=50.00`
`mc_currency=USD`
`custom=Custom+value+you+passed+with+your+HTML+form`
`etc.`
4. Do with data whatever you wish

In code I'm attaching with article under PDT directory you'll find example aspx and classes that will help you out in following previously mentioned flow.

And, as with AutoReturn, one notice before closing subject - take a look at **custom** variable in PayPal's response to your HTTP POST. You'll probably want to utilize this variable as it allows you to pass some information from your payment page to your post-processing page without presenting it to user. To name one use, in some of my PayPal implementations I track user with it - when he started payment process and when/if he finished it.

Instant Payment Notification (IPN)

One big shortcoming of PDT is that it is user-driven process, meaning - if user closes browser after performing payment and before being redirected to your site- you'll lose opportunity to run your post processing logic. That's why you are advised to combine PDT with IPN for any serious integration with PayPal.

IPN is a back-end mechanism that makes HTTP POSTs to your page, notifying you of important events. It is used not only for PostPayment processing, but also for things that come after, like handling user cancelation of recurring payments.

Being back-end technology it is somewhat harder to implement and debug than PDT. There are couple of things you should be aware of before starting to implement IPN:

- IPN messages can be delayed sometimes. I know, I know... that beats word "Instant" in IPN, but that's how things are.
- There is known history of problems with IPN service; two latest incidents happened on [October 2nd 2009](#) (2 hour delay) and on [September 6th 2009](#) (6 hour delay).
- Whenever you have problems with IPN be sure to check [Live Status](#) page and see if there is incident notification before digging into debugging and changing your script. There is also similar page for [Sandbox Status](#).

Before being able to receive IPN messages you'll need to activate this service; follow these steps:

1. Log in to your Premier or Business account
2. Click the Profile sub tab
3. Click Instant Payment Notification in the Selling Preferences column.
4. Click the 'Edit IPN Settings' button to update your settings.
5. Select 'Receive IPN messages (Enabled)' and enter URL of your IPN handler.
6. Click Save, and you should get a message that you have successfully activated IPN.



Activating Instant Payment Notification

As with AutoReturn, you can overwrite IPN handler url set in profile in individual forms by adding **notify_url** variable (see [HTML Variables reference](#)). Know that this will influence not only initial IPN message but all future messages related to that transaction (they will all go to **notify_url**).

To handle IPN messages you'll need to create HTTP handler somewhere on your website. When significant event occurs (e.g. user performs payment) following flow takes place:

1. PayPal will send HTTP POST to your IPN handler with number of variables
2. After receiving HTTP POST and parsing it, you need to submit complete body of the message back to **<https://www.paypal.com/cgi-bin/webscr>** (or **<https://www.sandbox.paypal.com/cgi-bin/webscr>** for Sandbox account). When you are doing this be sure to send message back in exact format in which you received it; the only thing you are allowed to do is adding **cmd=_notify-validate**. This is all done in order to verify that HTTP POST was authentic and sent from PayPal
3. PayPal will respond with either VERIFIED or INVALID. After you receive this response, be sure to send 200 OK to prevent additional attempts from PayPal to send IPN. If you don't close loop with 200 OK, PayPal will start resending IPN (starting from 4 seconds and doubling - 8 seconds, 16 seconds, 32 seconds... up to 4 days).

If you received INVALID response that could mean two things:

- Someone tried to send malicious message to your IPN handler
- Your implementation isn't perfect.

In case of malicious message you're on your own (log IP, take appropriate action), but for imperfect implementations visit this [IPN troubleshooting topic](#) on PayPal Developer forums; it is full of useful tips that should help you solve INVALID responses.

Also, before going online with your IPN handler, be sure to test it thoroughly with [Instant Payment Notification \(IPN\) simulator](#). As IPN handler will work in background you'll want to test it as much as you can before going live and relying on its processing.

Another thing you'll want to visit if you are implementing IPN is [ScriptGenerator2 page](#)... it can quickly generate IPN handler in language of your choice. Funny thing - they are missing generator for ASP.NET/C#; for that check out IPN directory in code-archive I've attached to this article and [these Code Samples](#).

Finally, on PayPal Developer Center there is this nice [page that lists most of the IPN and PDT variables](#). I'm saying most because there are some variables missing on it (check comments on [this page](#)), but list is far beyond being useless.

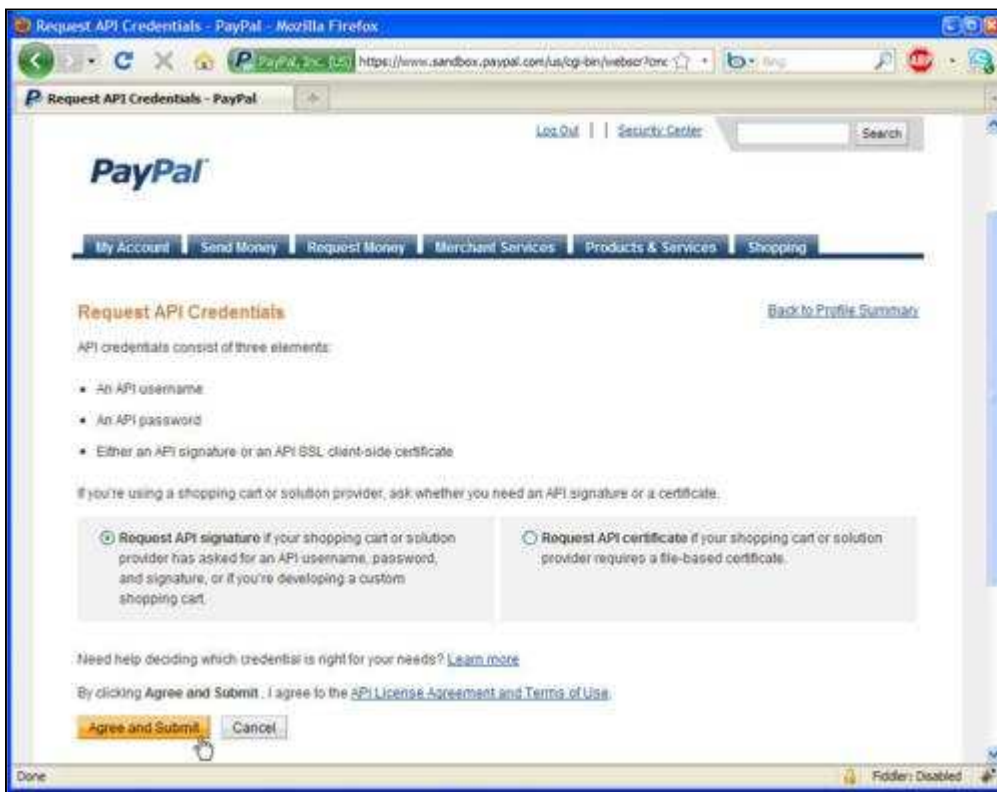
PayPal API

As said in introduction, most developers, even if they have no previous experience with PayPal, start directly with API. Google search brings up either [API Reference](#) or [SDKs and Downloads](#) page and then browsing through [Documentation](#) starts. I don't want to say that there is something wrong with using PayPal API for payments; I want to say that in most of the cases it's not necessary to go down that path.

PayPal API is much more than just a mechanism for payment - if you look at provided [API Reference](#) page, you'll see that there are lots of methods not tied directly to "user performing payment". You can use API to browse through history of your transactions, issue refund or update recurring payments profile. So how do you start using it?

First and foremost, you'll need to enable API access in your account; follow these steps:

1. Log in to your Premier or Business
2. Click the Profile sub tab.
3. Click the API Access link under the Account Information header.
4. Click the link that says Request API Credentials / View API Certificate.
5. Select either API Signature or API Certificate.



Step 5 from activating API Access

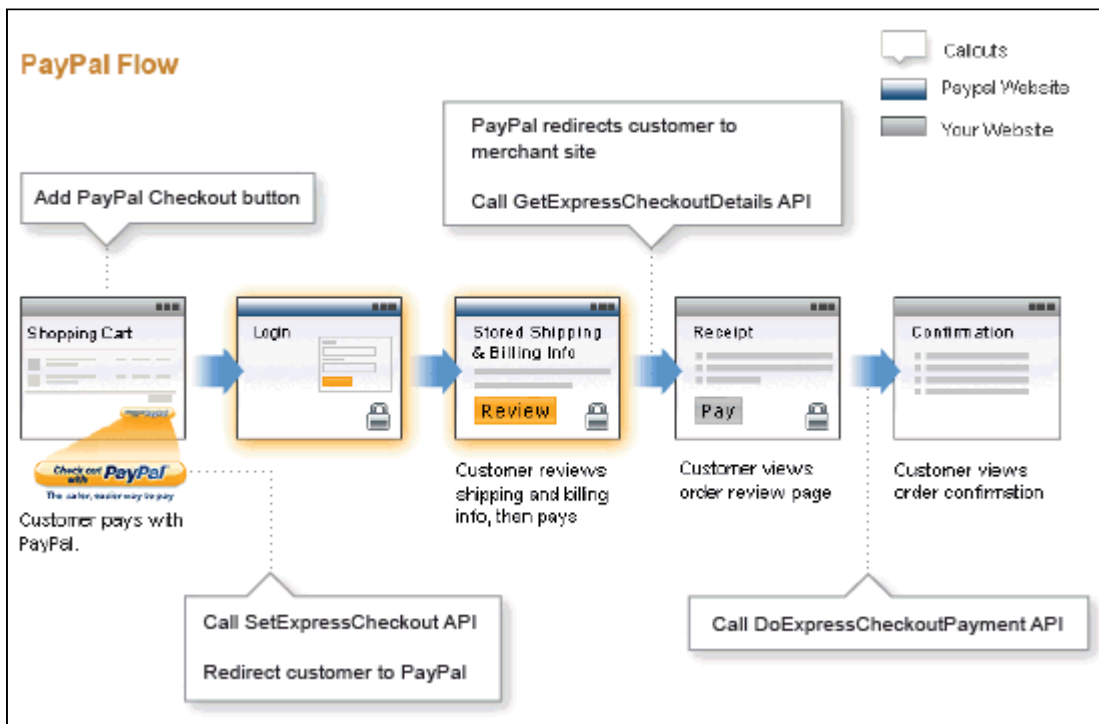
I recommend that you select API Signature and examples that follow will assume you made this choice. There is nothing wrong with selecting API Certificate; I just find it more demanding from setup perspective.

Now that you have credentials to make API calls, how do you perform them? The approach that will work equally well with all platforms is to [download SDK](#), target [appropriate API endpoint](#), and start making HTTP calls with either Name-Value pairs or SOAP.

However, for .NET developers I recommend different approach. Considering that Visual Studio has awesome WSDL parser, I urge you to just add Web Service Reference to <https://www.paypal.com/wSDL/PayPalSvc.wsdl>. After few moments you'll have up-to-date class ready to serve you with all benefits of strong typing - no building of HTTP requests, no copy-pasting field names and no cumbersome parsing of responses. You have same thing available for Sandbox at url <https://www.sandbox.paypal.com/wSDL/PayPalSvc.wsdl>.

Express Checkout

Express Checkout is the most flexible PayPal integration solution. User is redirected to PayPal just for authentication and confirmation that he wants to pay for your services and after that everything is done on your website; you'll make calls to PayPal API in background. Following picture describes process (taken from [this page](#)):



Express Checkout flow

1. You'll add PayPal Checkout button that invokes [SetExpressCheckout](#) method of PayPal API after it is clicked.
 1. If you are invoking this method for one time payment, it'll be valid if you include only required fields. Setting NOSHIPPING variable to 1 is important if you are selling some online service (it'll help you skip Shipping info page).
 2. If you are invoking this method in order to set recurring payments, be sure to set L_BILLINGTYPE0 to RecurringPayments and L_BILLINGAGREEMENTDESCRIPTION0 to valid description of your service.
2. [SetExpressCheckout](#) will return 20 char token that will uniquely identify your transaction. This token is valid for 3 hours. After you receive it, redirect user to [https://www.paypal.com/cgi-bin/webscr?cmd=_express-checkout&token=\[TOKEN\]](https://www.paypal.com/cgi-bin/webscr?cmd=_express-checkout&token=[TOKEN]) (you can guess what is url for sandbox, right?)
3. User will review payment information and if everything is OK enter his login credentials. After this PayPal will redirect him to url you specified with RETURNURL when you called SetExpressCheckout
4. When your RETURNURL is hit you need to invoke [GetExpressCheckoutDetails](#) method and see details of actual transaction; verify that everything is in order.
5. Now, all that is left is to commit transaction. Depending on what you did in step 1, there are two things that can be done.
 1. For one time payments you'll just invoke [DoExpressCheckoutPayment](#) and forward appropriate variables.
 2. For recurring payments you'll invoke [CreateRecurringPayments](#) method. It is required that you include DESC field and match it to value entered in L_BILLINGAGREEMENTDESCRIPTION0 when you called SetExpressCheckout.

In a nutshell - that's it. Again, I provided code examples that follow previously specified flow in archive accompanying this article (under API directory). If you wish, you can also use [PayPal Express Checkout Integration Wizard](#) for generating reference code.

Direct Payment (Website Payments Pro)

Most developers aren't aware that PayPal platform can be used for just Credit Card processing. This part of PayPal API is called Direct Payment and when combined with Express Checkout (which only services customers with PayPal accounts) you get what is referred as [Website Payments Pro](#) on PayPal Developer Center.

To be able to call methods that are part of Direct Payment ([DoDirectPayment](#) and [CreateRecurringPayments](#)) you first need to satisfy some conditions:

1. Have Business account that is based in US, UK or Canada
2. Oblige that you'll implement both Express Checkout and Direct Payment on your website
3. Submit application for Website Payments Pro through your PayPal account and have it approved
4. Pay monthly fee (currently \$30 per month)

After you have Website Payments Pro account in place, calling Direct Payment methods is pretty straightforward - if in doubt either visit [API Reference](#) page or look at code attached to this article. Just know that if you try to call any Direct Payment method on account that doesn't have Pro enabled, you'll get error with code 10501 (this is [one of the most common problems](#) reported in Sandbox forum).

Lastly, once you start dealing with credit cards you'll need to take care about PCI Compliance; here is nice topic that provides [more information on that](#).

Conclusion

My hope is that this article gave you good overview of PayPal integration options. If it did that, I'll be at peace - as once you have understanding of concepts laid out in this article, you'll easily fetch needed details from provided links. Sure, there are some topics we haven't touched, like [Encrypted Website Payments](#), [PayPal API Certificates](#) or [Payflow Gateway](#) but I think you can tackle even that on your own once you fully understand all things written here.

If you get stuck on anything, I suggest that you first visit [PayPal Developer Community](#) and ask your question in appropriate forum. Number of great, knowledgeable developers like [Ciaran](#), [sliddicoat](#), [skier](#), [angelleye](#), etc. monitor those forums and it's highly probable that you'll receive answer to almost any PayPal issue within hour. I also have an account on that site ([lepiple](#)) and try to answer questions whenever I have time; so feel free to send me private message if you drop by or run into trouble.

History

- October 1st, 2009 - Initial version of the article

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPO\)](#)

About the Author

Predrag Tomasevic There isn't much to tell... 😊


Occupation: Architect

Company: Freelancer

Member

Location:  Serbia

Discussions and Feedback

 **1 message** have been posted for this article. Visit http://www.codeproject.com/KB/aspnet/aspnet_c_aspnet.aspx to post and view comments on this article, or click [here](#) to get a print view with messages.

[PermaLink](#) | [Privacy](#) | [Terms of Use](#)

Last Updated: 6 Oct 2009

Editor: [Predrag Tomasevic](#)

Copyright 2009 by Predrag Tomasevic
Everything else Copyright © [CodeProject](#), 1999-2009
Web15 | [Advertise on the Code Project](#)