

Make Salesforce and PayPal Work Together

Download the PDF of this article.

In this Article

- [Configure Your Salesforce Object](#)
- [Create Your Apex PayPal Notification Class](#)
- [Create Your Apex PayPal Notification Endpoint:](#)
- [Configure Your FormAssembly Form](#)
- [Test Your Setup](#)
- [Troubleshooting](#)

Related Articles

This page is an internal facing article.

How can I make my organization's PayPal account and Salesforce account interact more closely?

By default, only credit card transactions which take place immediately after the form is submitted will show in the FormAssembly response. A PayPal transaction can take anywhere from moments (with a credit card) to days (with an eCheck).

Once the payment clears on PayPal, the IPN should go out within minutes.

With a little bit of Salesforce VisualForce code, you can make your PayPal Instant Payment Notifications (IPNs) update your Salesforce records, closing the loop from your forms to PayPal to Salesforce.

**We do not support, adjust, or fix the code below.
The following code is provided to you as a starting point to
reference.
You will need to adjust this code based on your needs.**

Note: The following instructions are geared towards advanced Salesforce users who are comfortable with Apex code and site setup.

In the steps below, we'll use the example of a form set up to create a Salesforce Opportunity and receive payment for participation in that Opportunity by the respondent. The same procedure would work for any other Salesforce object type.

Configure Your Salesforce Object

In Salesforce, go to **Setup** → **App Setup** → **Customize** and select the object you wish to contain PayPal information. Here, we'll go with **Opportunity**. Add the following custom fields:

FormAssemblyID	<p>This is a unique key that FormAssembly adds to your object in order to reference it later.</p> <div>Properties: Text(255) (External ID) (Unique Case Insensitive) API Name: FormAssemblyID__c</div>
paid	<p>A checkbox for easy querying in Salesforce over your objects to see if the object's PayPal payment has completed.</p> <div>Properties: Checkbox API Name: paid__c</div>
PayPalInfo	<p>A large field to hold all data sent by PayPal about the transaction once complete.</p> <div>Properties: Long Text Area(32768) API Name: PayPalInfo__c</div>

Create Your Apex PayPal Notification Class

**We do not support, adjust, or fix the code below.
The following code is provided to you as a starting point to
reference.
You will need to adjust this code based on your needs.**

Note: The following instructions are geared towards advanced Salesforce users who are comfortable with Apex code and site setup.

In Salesforce, go to **Setup** → **App Setup** → **Develop** → **Apex Classes** and select **New**. Give the class the name `IPNHandlerController`, and add the following code into the interface.

Please note, you will need to modify this code to fit your specific site setup.

```

public class IPNHandlerController {

    public PageReference myIPNupdate() {
        try{
            PageReference pageRef = ApexPages.currentPage();
            //Get the value of the 'custom' parameter from current page
            String paramCustom = pageRef.getParameters().get('custom');
            opportunity = [select Id,paid__c from Opportunity where FormAssemblyID__c = :paramCustom ];

            String content = "";
            for(String key : pageRef.getParameters().keySet()){
                //Note that there is no guarantee of order in the parameter key map.
                content += key + ' : ' + pageRef.getParameters().get(key) + '\n';
            }
            opportunity.PayPalInfo__c = content;
            opportunity.paid__c = True;
            update opportunity;

            PageReference newPage = new ApexPages.StandardController(opportunity).view();
            newPage.setRedirect(true);

            return newPage;
        } catch (System.Exception e){
            //A failure occurred
            system.debug(e);
            return null;
        }
    }

    public Opportunity opportunity {get; set;}

    public IPNHandlerController() {
    }
}

```

Next, if testing / code coverage is a concern, create a new Apex Class called `IPNHandlerTestClass` by pasting this code into a new Apex Class window:

```
@istest
private class IPNHandlerTestClass {
    public static testMethod void testMyIPNupdateSuccess(){
        IPNHandlerController ipn = new IPNHandlerController();

        Opportunity c = new Opportunity(Name='Test01',CloseDate=date.parse('1/1/2010'),StageName='Qualification',
        FormAssemblyId__c='111111101111110z');
        insert c;
        ApexPages.currentPage().getParameters().put('custom', '111111101111110z');

        PageReference p = ipn.myIPNupdate();
        System.assertNotEquals(null,p);
    }

    public static testMethod void testMyIPNupdateFailure(){
        IPNHandlerController ipn = new IPNHandlerController();
        ApexPages.currentPage().getParameters().put('custom', '111111101111111z');
        PageReference p = ipn.myIPNupdate();
        System.assertEquals(null,p);
    }

    public static testMethod void testIPNHandlerController(){
        //You should customize this to fit your needs.
        System.assertEquals(true,true);
    }
}
```

Note that in the code above, we're using the Opportunity type object in Salesforce as our targeted object type to update when PayPal information comes in. You could just as easily use a Contact object or a Custom object, by substituting the object type name for Opportunity above.

Create Your Apex PayPal Notification Endpoint:

In Salesforce, go to **Setup** → **App Setup** → **Develop** → **Pages** and select **New**. Name the page `IPNHandler`, and paste the following code into the interface:

```
<apex:page controller="IPNHandlerController" action="{!myIPNupdate}" />
```

Now go to **Setup** → **App Setup** → **Develop** → **Sites** and place the page into your Salesforce Site. For example, with a new site, you could set the **Active Site Home Page** to `IPNHandler`. However, this is not recommended for existing sites.

Make a note of where IPNHandler page is located. It should be something that looks like:

<https://xxxxx.na3.force.com/yyyy/IPNHandler>

where xxxxx.na3.force.com is your Salesforce sites domain, [yyyy](#) is the path to where you are hosting the IPNHandler page, and [IPNHandler](#) is the name of the page we've created above.

Configure Your FormAssembly Form

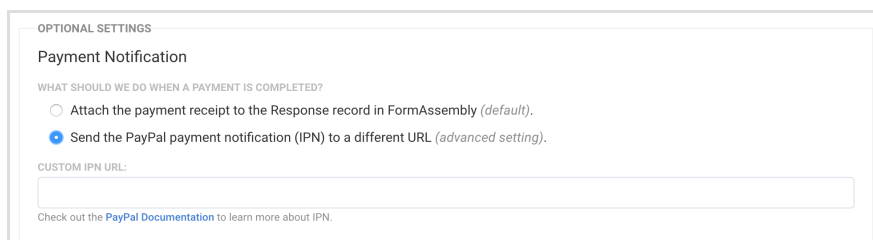
In FormAssembly, open your form's Salesforce Connector configuration page.

1. Map your usual Opportunity fields.
2. Map the Salesforce field [FormAssemblyID](#) to a formula.
3. Set that formula value to: https://www.tfaforms.com/responses/view/%%RESPONSE_ID%%

Now, open your form's PayPal Connector configuration page:

1. Map your PayPal connector as usual.
2. Change your IPN notification field to the location of your IPNHandler page. For example:

<https://xxxxxx.na3.force.com/IPNHandler>

A screenshot of the 'OPTIONAL SETTINGS' section in the PayPal Connector configuration page. Under 'Payment Notification', there are two radio button options. The first is 'Attach the payment receipt to the Response record in FormAssembly (default)'. The second is 'Send the PayPal payment notification (IPN) to a different URL (advanced setting)', which is selected. Below these options is a text input field labeled 'CUSTOM IPN URL:'. At the bottom, there is a link to 'PayPal Documentation'.

Test Your Setup

When properly configured, the process will work like so:

1. A respondent completes the form.
2. A new Opportunity record is created in Salesforce, with a value like <https://www.tfaforms.com/responses/view/1222222> in the **FormAssemblyID** Salesforce field.
3. Respondent completes PayPal section, starting the PayPal authorization and verification process.
4. When PayPal completes that transaction, PayPal will trigger an IPN to the location: <https://xxxxxx.na3.force.com/IPNHandler>

5. Once there, the Apex code we've created will lookup the Opportunity based on the custom parameter included in the PayPal IPN notification which will match the **FormAssemblyID** field, populate the field **paid** to checked, and populate the field **PayPalInfo** to a newline delimited list of the PayPal parameters passed from the IPN.
-

Troubleshooting

A couple of things to re-confirm, if you continue to have issues:

- Did you "activate" the [Force.com](https://force.com) site?
- Is the IPNHandler URL in FormAssembly correct?

There is an IPN log on the PayPal side which should be looked at first.

On the Salesforce side, you can check the Apex debug logs.

Note: PayPal's payment system is asynchronous, meaning that a person who pays by PayPal may have the transaction complete in seconds (if by credit card) or days (if by eCheck).

As a result, the data from the transaction is not available at the time the email notification is fired off and the Salesforce Connector is run. This is what sets up the IPNHandler in Salesforce to accomplish: provide a "receiver" that will receive PayPal's transaction info seconds or days later when the transaction is complete and the money is in your PayPal account.
