

PaymentConnect™

for Appexchange



Level 2 Integration: Event Registration Example

Copyright Linvio, Inc. 2010

Table of Contents

Event Registration Sample Code.....	3
Screen Captures.....	5
Sample Code	9
EventReg	9
EventRegController.....	10
EventConfirm	14
Other Level 2 Integration Pages.....	15
Additional Help.....	16

Event Registration Sample Code

The Event Registration sample demonstrates Level 2 integration with PaymentConnect and shows how you can build payment processing into custom Visualforce workflow in Salesforce without having to create your own payment terminal or implement API callouts to your payment processor.

Level 1 Integration	<p>Uses click-to-configure features in PaymentConnect to:</p> <ol style="list-style-type: none"> 1. Map fields from standard and custom Salesforce objects to PaymentConnect terminals 2. Automate handling of payment notifications from outside of Salesforce (e.g. shopping carts and buy now links) 3. Create post-payment workflow, emails and reports <p>Requires advanced Salesforce administrator skill.</p>
Level 2 Integration	<p>Makes use of pre-built Salesforce and Sites payment terminals to add payment processing to custom javascript and visualforce workflows. Requires intermediate Salesforce developer skills.</p>
Level 3 Integration	<p>API-level integration with PaymentConnect. Requires advanced Apex developer skills as well as in-depth knowledge about your payment processor API features.</p>

In this example, we have created an event registration page (EventReg) that optionally takes a Salesforce Contact record Id as an input parameter, displays event registration options and redirects the Salesforce user to the PaymentConnect Payment Terminal to complete the transaction.

By passing parameter values to the terminal the sample code is able to set up the payment record and store any custom objects & fields prior to transaction processing, and specify what page the user will be directed to after the payment has been processed.

The following diagram shows the flow of control from the event registration page, to payment terminal and finally to the event registration confirmation page:

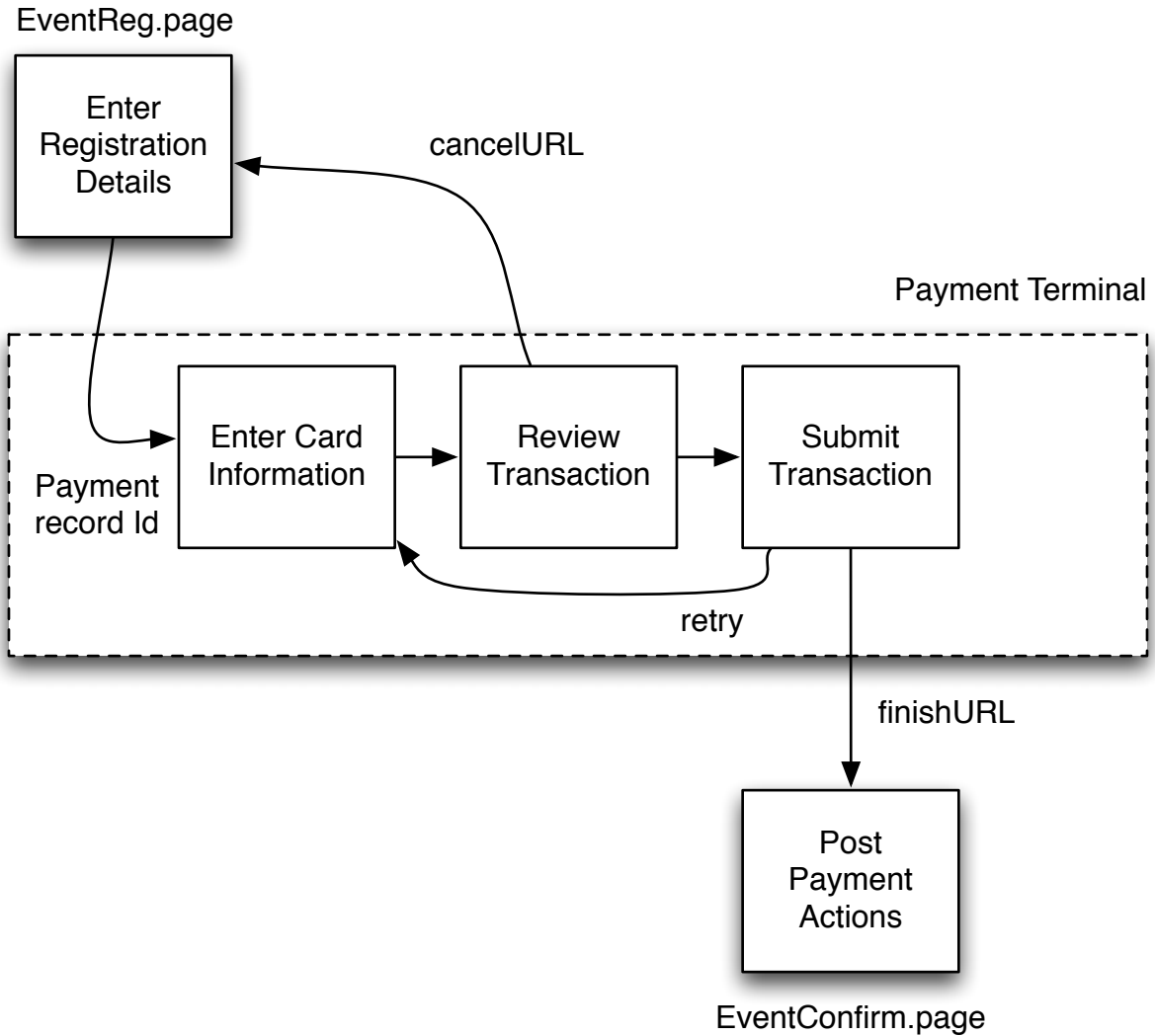


Figure 1. Using the existing PaymentConnect Payment Terminal, the bulk of the payment processing work has already been done for you. Just set up the Payment record, pass control to the terminal, and continue your custom workflow post-payment.

Screen Captures

The following screen captures show the process flow for the sample event registration process:

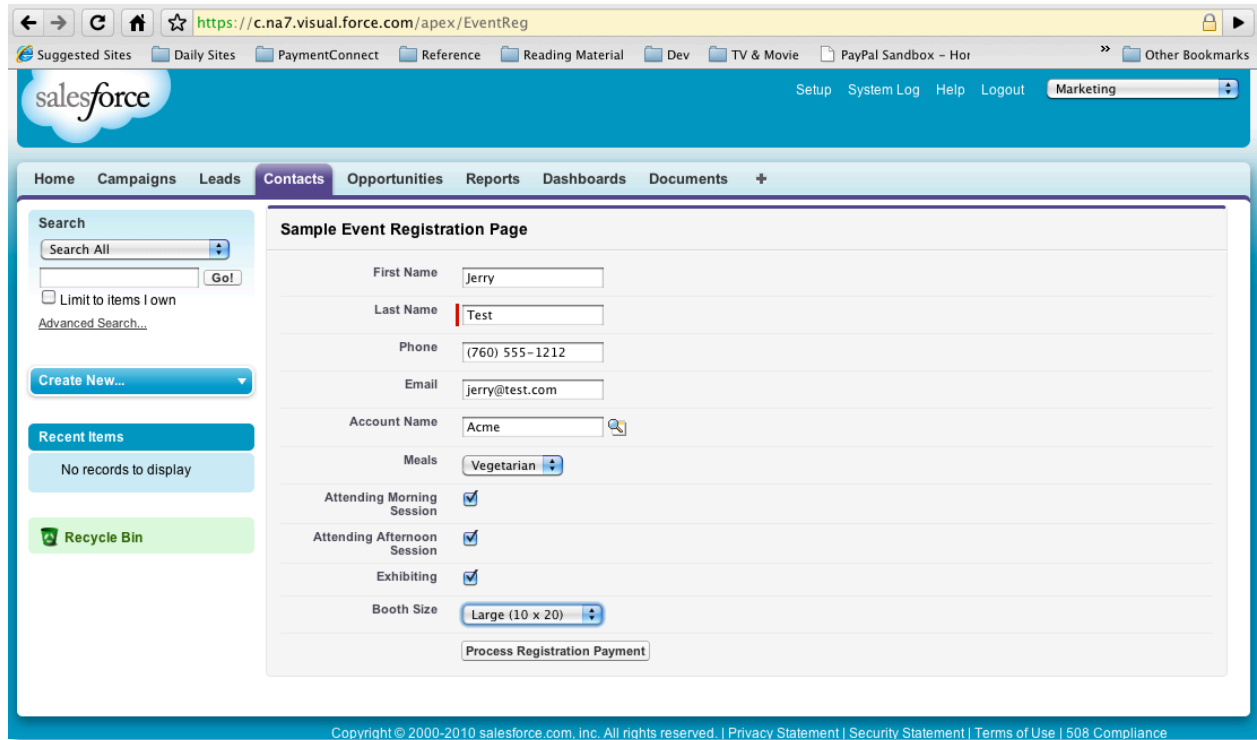


Figure 2. EventReg - Custom visualforce registration page.

On the EventReg page the user enters customer information and event registration options. When the user clicks "Process Registration Payment" the page calculates the total registration amount & tax, adds the contact, creates a payment and shopping cart items, and redirects the user to the PaymentConnect payment terminal.

At this point in the process, the developer can optionally set custom field values on the Payment record, connect the payment to other objects (i.e. a custom event registration object) and set the On-Payment-Completed tag field on the Payment or individual Shopping Cart Items to trigger automated post-payment workflows.

The screenshot displays the PaymentConnect Payment Terminal interface. At the top, there is a navigation menu with options: Home, Campaigns, Leads, Contacts, Opportunities, Reports, Dashboards, and Documents. Below the navigation, a search bar is visible with a 'Search All' dropdown and a 'Go!' button. A 'Limit to items I own' checkbox and an 'Advanced Search...' link are also present. On the left side, there are sections for 'Create New...' and 'Recent Items', which lists 'Event Registration', 'Event Registration (Full Day)', 'Exhibitor Booth (10x20)', and 'Jerry Test'. A 'Recycle Bin' icon is located at the bottom left. The main content area is titled 'Payment Terminal' and includes a 'Processor Connection' dropdown set to 'Authnet'. Below this, there are 'Cancel' and 'Review Transaction' buttons. The 'Transaction Details' section contains the following information: Payment Name (Event Registration), Amount (3000.0), Shipping & Handling (0.00), Tax (150.00), Total Amount (3,150.00), and Tax Percent (5.0%). There is also a checkbox for 'Apply Tax to Shipping'. The 'Related Standard Objects' section shows 'Opportunity' and 'Contact' (Jerry Test) fields, along with an 'Account' field set to 'Acme'. The 'Account Information' section includes 'Payment Type' (Credit Card Charge selected), 'Card Type' (Visa), 'Credit Card Number' (4111111111111111), 'CVV Number' (123), and 'Card Expiration' (Jan 2014). The 'Address Information' section is partially visible at the bottom.

Figure 3. PaymentConnect Payment Terminal - Entering card information.

When the PaymentConnect Payment Terminal loads, it retrieves all the transaction data from the Payment record created in the last step, and displays it in the terminal so the user can enter credit card information and submit the charge request.

Payment Terminal Help for this Page ?

Transaction Review

Processor Connection: Authnet

Please review the transaction details below and click 'Submit Transaction' to process this payment.

Opportunity:
Account: Acme
Contact: Jerry Test

Payment Name: Event Registration
Amount: 3000.0
Tax: \$150.00
Shipping: 0.00
Total Amount: \$3,150.00 USD

Payment Type: Credit Card
Credit Card Number: 4111111111111111
Credit Card Type: Visa
CVV Number: 123
Expiration Month/Year: 01 2014

Billing Address: 555 Data Blvd
Los Angeles, CA 94335
US

Shipping Address:

Figure 4. Payment Terminal transaction review.

The Payment Terminal displays a summary of the transaction before submitting the charge request to the payment processor. At this point, the user can click “Cancel” and the transaction will be aborted, taking the user back the custom event registration page.

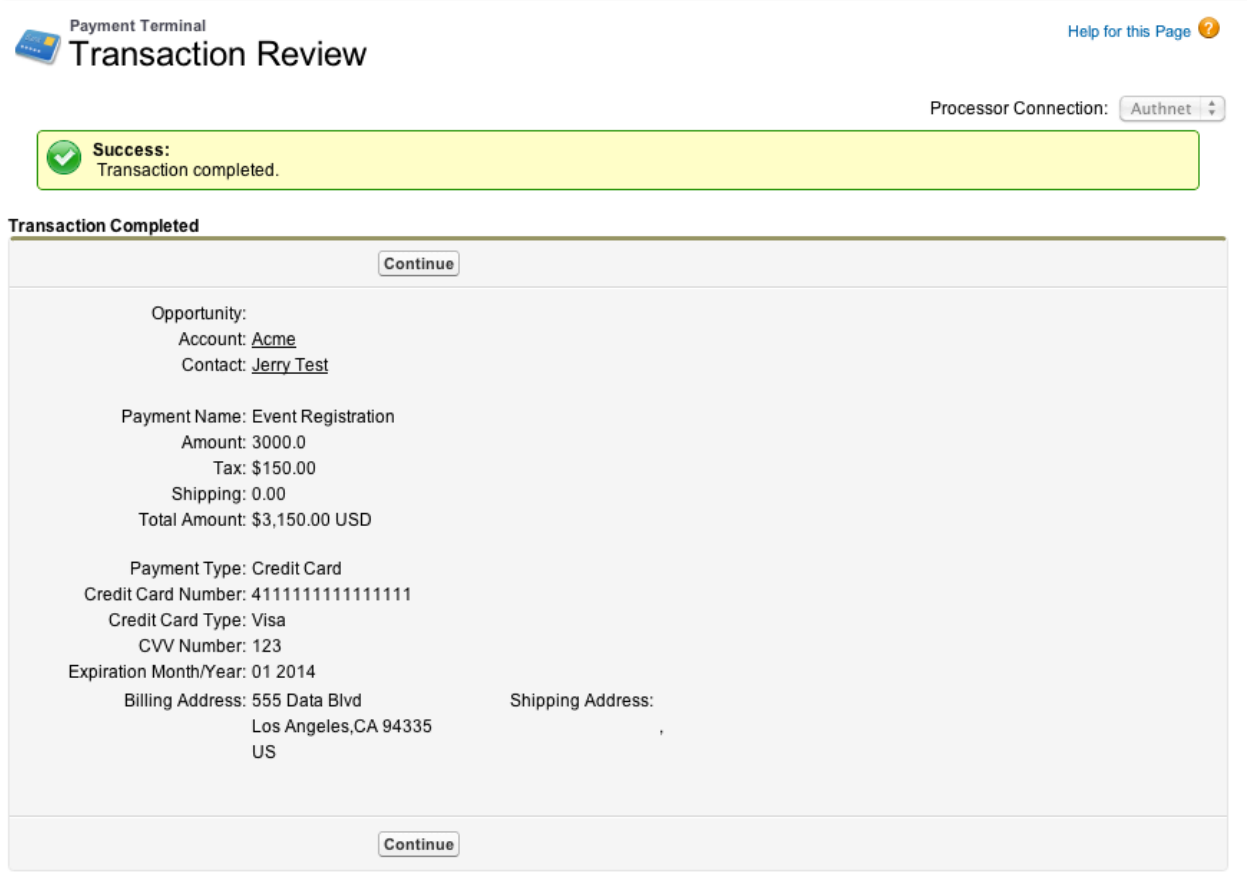


Figure 5. Payment Terminal - transaction result.

The last step in the Payment Terminal is to display the transaction result. If the charge is successful, a success message will be displayed. If not, the user can hit the “Back” button and retry with corrected card or contact information.

When the user clicks “Continue” he/she will be redirected to the final page in this scenario – the custom registration confirmation page.

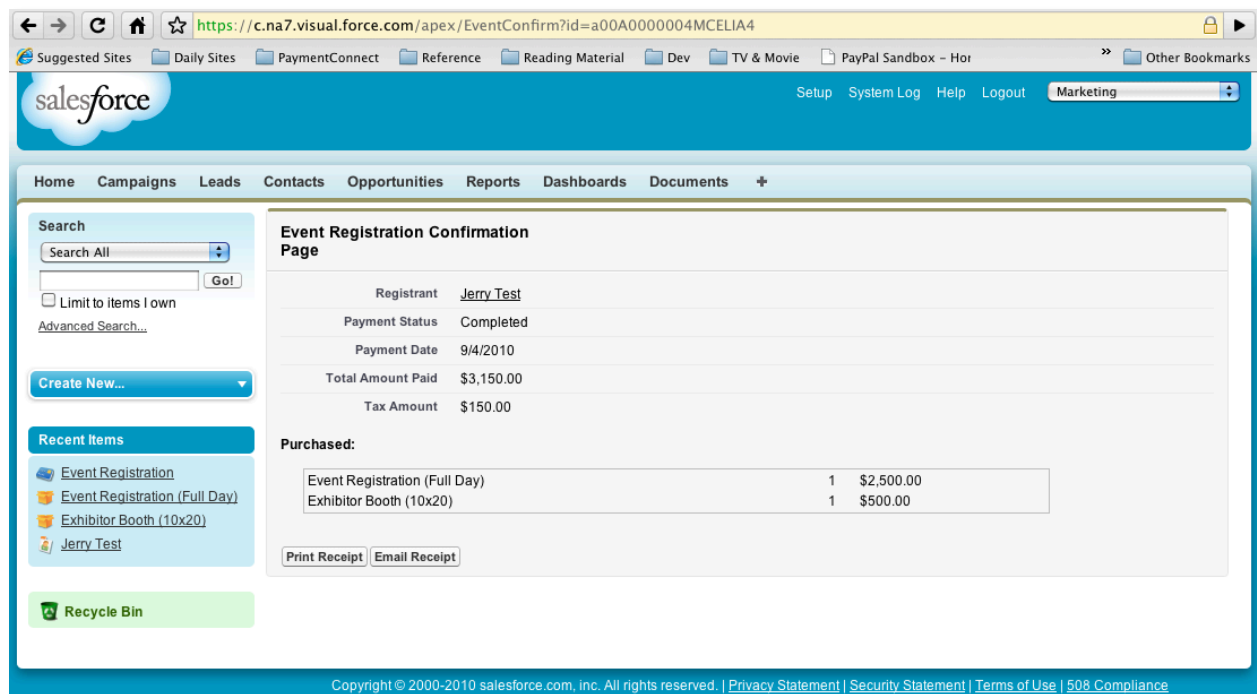


Figure 6. EventConfirm - confirmation of payment completed and options to print receipt, etc.

The final page in the event registration process is EventConfirm, which reloads the Payment record (now marked “Completed” after a successful charge) and displays a transaction summary and other post-payment process options.

Sample Code

EventReg

```
<apex:page standardController="Contact" extensions="EventRegController">
<apex:pageBlock title="Sample Event Registration Page" >
<apex:form >
<apex:pageBlockSection columns="1">

<apex:outputField value="{!Contact.Name}" rendered="{!Contact.Id <> null}"/>
<apex:inputField value="{!Contact.FirstName}" rendered="{!Contact.Id ==
null}"/>
<apex:inputField value="{!Contact.LastName}" rendered="{!Contact.Id ==
null}"/>
<apex:inputField value="{!Contact.Phone}" rendered="{!Contact.Id == null}"/>
<apex:inputField value="{!Contact.Email}" rendered="{!Contact.Id == null}"/>
<apex:inputField value="{!Contact.AccountId}" rendered="{!Contact.Id ==
null}"/>

<apex:pageBlockSectionItem >
  <apex:outputLabel value="Meals"/>
  <apex:selectList size="1">
```

```

        <apex:selectOption itemLabel="Vegetarian" itemValue="Veggie"/>
        <apex:selectOption itemLabel="Standard" itemValue="Standard"/>
    </apex:selectList>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
    <apex:outputLabel value="Attending Morning Session" />
    <apex:inputCheckbox value="{!morning}"/>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
    <apex:outputLabel value="Attending Afternoon Session" />
    <apex:inputCheckbox value="{!afternoon}"/>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
    <apex:outputLabel value="Exhibiting" />
    <apex:inputCheckbox value="{!exhibitor}" />
</apex:pageBlockSectionItem>
<apex:pageBlockSectionItem >
    <apex:outputLabel value="Booth Size"/>
    <apex:selectList size="1" value="{!booth}">
        <apex:selectOption itemLabel="Deluxe (20 x 20)" itemValue="20x20"/>
        <apex:selectOption itemLabel="Large (10 x 20)" itemValue="10x20"/>
        <apex:selectOption itemLabel="Standard (5 x 10)" itemValue="5x10"/>
    </apex:selectList>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
<apex:outputText value=""/>
<apex:commandButton value="Process Registration Payment"
action="{!processPayment}"/>
</apex:pageBlockSectionItem>

</apex:pageBlockSection>
</apex:form>
</apex:pageBlock>
</apex:page>

```

EventRegController

```

public with sharing class EventRegController {

    private pymt__PaymentX__c payment {get;set;}
    public Contact contact {get;set;}
    public Boolean morning {get;set;}
    public Boolean afternoon {get;set;}
    public Boolean exhibitor {get;set;}
    public String booth {get;set;}
}

```

```

    private Map<String, Double> boothRates = new
Map<String,Double>{'20x20'=>1000.00, '10x20'=>500.00, '5x10'=>250.00};

    public String returnUrl {
        get {
            if (returnURL == null || returnURL.length()==0) returnURL =
ApexPages.currentPage().getHeaders().get('Referer');
            if (returnURL <> null && returnURL.contains('?')) {
                returnURL = returnURL.split('\\?')[0];
            }
            return returnURL;
        }
        set;
    }

    public eventRegController(ApexPages.StandardController controller) {
        this.contact = (Contact)controller.getRecord();

        // Setup "In Process" payment record
        this.payment = new pymt__PaymentX__c(
            Name = 'Event Registration',
            pymt__Status__c = 'In Process',
            pymt__Date__c = Date.today(),
            pymt__Contact__c = this.contact.Id,
            pymt__Amount__c = 10.00
        );

    }

    public PageReference processPayment() {
        PageReference nextPage;
        PageReference currentPage = Page.eventReg;
        currentPage.getParameters().put('id',this.contact.Id);

        // Create contact if new
        if (this.contact.Id == null) insert this.contact;

        this.payment.pymt__Contact__c = this.contact.Id; // make sure
contact Id is stored on payment and upsert so we have a payment id to assign
to cart items
        upsert this.payment;

        // Add Shopping Cart Items
        pymt__Shopping_Cart_Item__c[] cartItems = new
pymt__Shopping_Cart_Item__c[]{};
        if (this.booth <> null && this.exhibitor &&
this.boothRates.keySet().contains(this.booth)) {
            cartItems.add(

```

```

        new pymt__Shopping_Cart_Item__c(
            Name='Exhibitor Booth ('+this.booth+)',
            pymt__Quantity__c = 1,
            pymt__Unit_Price__c =
this.boothRates.get(this.booth),
            pymt__Contact__c = this.contact.Id,
            pymt__Payment__c = this.payment.Id
            //,pymt__On_Payment_Completed__c =
'SomeCustomOPCTag',
            //pymt__Product__c = '01txxxxxxxxxxxxx',
            //pymt__Product_Code__c = 'productx'
        ));
    }

    if (this.morning && this.afternoon) {
        cartItems.add(
            new pymt__Shopping_Cart_Item__c(
                Name='Event Registration (Full Day)',
                pymt__Quantity__c = 1,
                pymt__Unit_Price__c = 2500.00,
                pymt__Contact__c = this.contact.Id,
                pymt__Payment__c = this.payment.Id
                //,pymt__On_Payment_Completed__c =
'SomeCustomOPCTag',
                //pymt__Product__c = '01txxxxxxxxxxxxx',
                //pymt__Product_Code__c = 'productx'
            ));
    } else if (this.morning || this.afternoon) {
        cartItems.add(
            new pymt__Shopping_Cart_Item__c(
                Name='Event Registration
('+(this.morning?'Morning':'Afternoon')+ ' Only)',
                pymt__Quantity__c = 1,
                pymt__Unit_Price__c = 1250.00,
                pymt__Contact__c = this.contact.Id,
                pymt__Payment__c = this.payment.Id
                //,pymt__On_Payment_Completed__c =
'SomeCustomOPCTag',
                //pymt__Product__c = '01txxxxxxxxxxxxx',
                //pymt__Product_Code__c = 'productx'
            ));
    }

    // subtotal and create cart items in Salesforce
    Double subtotal = 0;
    if (cartItems.size()>0) {
        for (pymt__Shopping_Cart_Item__c item :cartItems) {
            subtotal += item.pymt__Quantity__c *
item.pymt__Unit_Price__c;

```

```

    }
    insert cartItems;
}

// optionally calculate tax
Double tax = subtotal * .05;

// save updated total and tax amounts in payment record
this.payment.pymt__Amount__c = subtotal + tax;
this.payment.pymt__Tax__c = tax;
update this.payment;

nextPage = Page.pymt__PaymentTerminal;
nextPage.getParameters().put('cancelURL',
this.returnURL+'?id='+this.contact.Id);

nextPage.getParameters().put('finishURL', domain(this.returnURL)+'/apex/EventC
onfirm?id='+this.payment.Id);
nextPage.getParameters().put('id', this.payment.Id);
nextPage.setRedirect(true);
return nextPage;
}

// return the domain portion of a full url (assumes the domain ends in
.com)
public String domain(String url) {
    if (url <> null && url.contains('.com')) {
        return (url.split('.com')[0]+' .com');
    }
    return '';
}

private static testmethod void testEventRegController() {

    Test.setCurrentPage(Page.EventReg);
    Contact contact = new Contact();
    Account account = new Account(name = 'ApexTest');
    insert account;

    ApexPages.StandardController stdController = new
ApexPages.StandardController(contact);
    EventRegController controller = new
EventRegController(stdController);

    // Test utility methods
    controller.domain('https://www.salesforce.com?parm=x');
    String returnUrl = controller.returnURL;

    // Enter new contact details

```

```

    controller.contact.firstname = 'Bob';
    controller.contact.lastname = 'Test';
    controller.contact.email = 'bob@apextest.com';
    controller.contact.AccountId = account.id;

    // Select registration options
    controller.morning = true;
    controller.afternoon = true;
    controller.exhibitor = true;
    controller.booth = '5x10';

    controller.processPayment();
    System.assert(controller.payment.Id <> null);
}
}

```

EventConfirm

```

<apex:page standardController="pymt__PaymentX__c">
<apex:form >
<apex:pageBlock title="Event Registration Confirmation Page" >

<apex:pageBlockSection columns="1">

<apex:pageBlockSectionItem >
<apex:outputLabel value="Registrant"/>
<apex:outputField value="{!pymt__PaymentX__c.pymt__Contact__c}"/>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
<apex:outputLabel value="Payment Status"/>
<apex:outputField value="{!pymt__PaymentX__c.pymt__Status__c}"/>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
<apex:outputLabel value="Payment Date"/>
<apex:outputField value="{!pymt__PaymentX__c.pymt__Date__c}"/>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
<apex:outputLabel value="Total Amount Paid"/>
<apex:outputField value="{!pymt__PaymentX__c.pymt__Amount__c}"/>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
<apex:outputLabel value="Tax Amount"/>
<apex:outputField value="{!pymt__PaymentX__c.pymt__Tax__c}"/>
</apex:pageBlockSectionItem>

```

```

</apex:pageBlockSection>
<br/>
<h1>Purchased:</h1>
<br/><br/>
<table style="border:solid 1px #c0c0c0;width:80%;margin-left:20px;">
<apex:repeat value="{!pymt__PaymentX__c.pymt__R00N40000001tGNtEAM__r}"
var="item">
<tr>
<td>
<apex:outputField value="{!item.Name}" />
</td><td>
<apex:outputField value="{!item.pymt__Quantity__c}" />
</td><td>
<apex:outputField value="{!item.pymt__Total__c}" />
</td>
</tr>
</apex:repeat>
</table>
<br/><br/>
<apex:commandButton value="Print Receipt" onclick="alert('Not yet
implemented');"/>
<apex:commandButton value="Email Receipt" onclick="alert('Not yet
implemented');"/>
</apex:pageBlock>
</apex:form>
</apex:page>

```

Other Level 2 Integration Pages

PaymentConnect also includes the following packaged pages that can be incorporated into custom visualforce in a similar manner as shown above:

Page	Application
pymt__PayPalARBTerminal	For creating recurring billing profiles through PayPal.
pymt__AuthNetARBTerminal	For creating automated recurring billing profiles through Authorize.net.
pymt__CashEntry	For entering cash, check or money order transactions into Salesforce.
pymt__SiteCheckout	For Sites (public facing) one time charges. Includes support for Google Checkout, RBS Worldpay and PayPal "Buy Now" processing, in addition to PayPal and AuthNet credit card entry.
pymt__SiteSubscribeARB	For Sites (public facing) Authorize.net subscriptions.

pymt__SiteSubscribePRB	For Sites PayPal subscriptions.
ptm__PaymentTerminal	Included in the Secure Terminal add-on for PaymentConnect. Interchangeable with the pymt__PaymentTerminal.
ptm__PayPalARBTerminal	Included in the Secure Terminal add-on for PaymentConnect. Interchangeable with pymt__PayPalARBTerminal.
ptm__AuthNetARBTerminal	Included in the Secure Terminal add-on for PaymentConnect. Interchangeable with pymt__AuthNetARBTerminal.

See the PaymentConnect Developer Guide for specifics on URL parameters supported by each of these pages.

Additional Help

For more information visit the Linvio website at www.linvio.com or our forum www.linvio.com/forum .

Consulting and integration services are also available from Linvio.

Linvio, Inc.
sales@linvio.com
(860) 546-8466