Yes Software

CodeCharge™

Online Credit Card Verification Using VeriSign PayFlow Link

www.codecharge.com

Table of Contents:

1. Introduction:

This tutorial demonstrates how to use the VeriSign PayFlow Link system to verify credit cards for online e-commerce transactions. To best illustrate the techniques involved, the tutorial adopts a 'hands-on' approach which begins with the 'Bookstore' sample application and progressively adds functionality to it to make it capable of verifying credit card transactions. Note that the end result should be considered a prototype aimed at proving the concept and as such is not suitable for deployment in a real world situation.

2. Prerequisites:

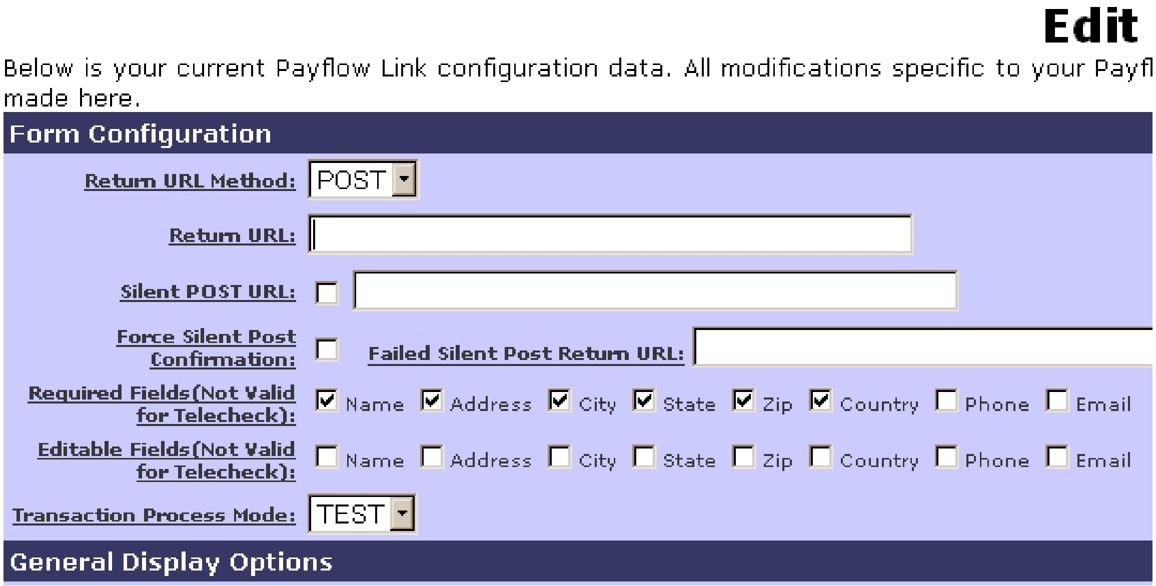
The following are the resources required to work along with this tutorial:

- CodeCharge
- A CodeCharge supported database
- A web server.
- A VeriSign PayFlow Link account (Free if used in test mode).
- The Bookstore example application which comes bundled with CodeCharge

3. Setting Up a VeriSign PayFlow Link Account

To successfully follow this tutorial, you need a VeriSign PayFlow Link account which can be acquired at https://register-payflow.verisign.com/testreg.cfm. Once you have signed up and activated the account, login to the VeriSign Manager (hereafter referred to as Manager) where you can download a comprehensive users guide on how to configure the various options. To download the users guide, click on the "Downloads" option in the Manager then at the bottom of the ensuing page click on "VeriSign Manager User Guide – Acrobat PDF" to download the file.

Please take a while to acquaint yourself with the system and once you are comfortable with the various features and options, move on to the configuration section of the Manager.



Edit

Below is your current Payflow Link configuration data. All modifications specific to your Payfl
made here.

Form Configuration

Return URL Method:	POST
Return URL:	
Silent POST URL:	□
Force Silent Post Confirmation:	□ Failed Silent Post Return URL:
Required Fields(Not Valid for Telecheck):	☑ Name ☑ Address ☑ City ☑ State ☑ Zip ☑ Country □ Phone □ Email
Editable Fields(Not Valid for Telecheck):	□ Name □ Address □ City □ State □ Zip □ Country □ Phone □ Email
Transaction Process Mode:	TEST

General Display Options

Edit the your account configuration according to the instructions in the users guide. For the purpose of this tutorial please note the following nuances:

Property	Value
Method	Post
Return URL	The URL of the Web site that your customers will be directed to when their transactions are complete. As you shall see later on in this tutorial, a new page called 'VerisignReturn.[Ext]' shall be created to serve this purpose and hence your URL should be something like: http://www.yourdomain.com/bookstore/verisignreturn.asp**.** You can also enter a static IP address in place of the domain name
Silent Post URL	Leave this option blank
Force Silent Post Confirmation	Leave this option blank
Failed Silent Post Return URL	Leave this option blank
Required Fields	Select Name, Address, City, State, Zip, and Country
Editable Fields	Leave all the fields blank
Transaction Process Mode	**TEST**
Merchant Disp Name	Enter a name of your choice or alternatively configure an image
Justification	Centered
Color and Image Configuration	Optional
Receipt Header Text	Optional
Receipt Footer Text	Optional
Receipt Button Text	Enter a label of your choice
E-mail Receipt to Customer	Yes
E-mail from Merchant Address	Enter your email address
E-mail to Merchant Address	Enter your email address
E-mail to Merchant Address (copy)	Optionally enter another email address
E-mail Header Text	Enter text of your choice
E-mail Footer Text	Enter text of your choice
AVS	**NO**
Accepted URLs	Enter your domain name or IP address

Save your changes!!

4. Modifying the Bookstore database.

To accomplish the added functionality needed for credit card verification, the bookstore database has to be modified as follows:

1. Add the following fields to the **members** table:

Name	Data Type
city	text
state	text
zipcode	text
country	text
expdate	text

The first four fields will store address details while the last one will contain the expiry date of the credit card (MM/YYYY). While you are at it, fill in the new fields with some data. For the credit card number, VeriSign provides the number 5105105105105100 for testing purposes. You can read more about it in the VeriSign Manager User Guide.

2. Add a new table called **invoices** with the following columns:

Name	Data Type
invoice_id	text
member_id	number

Once a transaction has been completed successfully, the reference number returned from the VeriSign website is used to create a unique invoice record for the transaction. Additionally, the ordered merchandise is removed from the **orders** table and placed into a new **purchases** table defined as follows:

Name	Data Type
order_id	number
invoice_id	text [50]
member_id	number
item_id	number
quantity	number

Notice that apart from the *invoice_id* field, the **purchases** table is similar to the **orders** table. In essence, the **purchases** table stores previous orders pertaining to a specific invoice_id (reference number).

Optionally, you could add more records to the **card_type** table e.g. Discover, MasterCard, JCB, Enroute, Diners Club.

5. CodeCharge Design and Coding

You are now ready to begin designing and coding in CodeCharge. Before you open the Bookstore example application, you might want to make a copy of the entire folder in case you happen to need the original version in the future.

Open the Bookstore sample application and if you are not already familiar with it, you might want to generate the pages and view the store to see where you will be picking up from. At the least, generate and view the pages to ensure that everything is working fine. During the course of this tutorial, generate the modified pages ever so often to ensure that there are no errors.

In the course of this tutorial the following changes will be made:

- Add a link from the **ShoppingCart** page to the **PrepareCheckOut** page.
- Add a page called **PrepareCheckOut** which receives order details from the **ShoppingCart** page and displays the information to be sent to the VeriSign website.
- Add a Page called **VerisignReturn** which receives pertinent details from the VeriSign website once a transaction has been completed.
- Add city, state, zip, country and expdate fields to the **MyInfo** page so that the additional fields added to the **members** table can be edited.
- Adjust the page layouts to make them more readable.

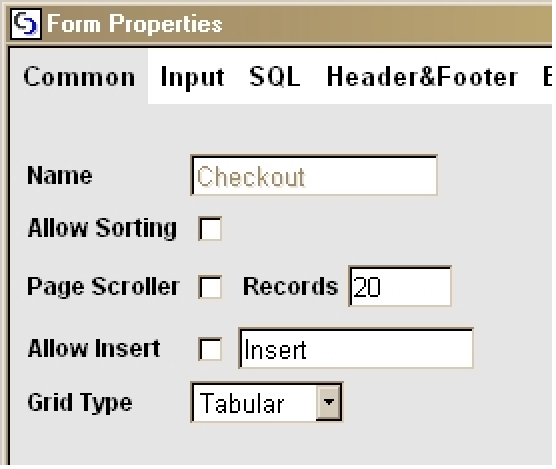
5.1 Adding A Check Out Link To The Shopping Cart Page.

The first thing to do is add a link from the **ShoppingCart** page to the **PrepareCheckOut** page. As of now, the **PrepareCheckOut** page does not exist but it shall be created in the next stage. To add the link, follow the steps below:

1. Insert a new form with the following properties.

Property Name	Value
Name	Checkout
Form Title	Check Out
Type	Grid
Form Action	Leave Blank
Table	Leave Blank

2. Open the form properties dialog box and under the **Common** tab, adjust the properties as shown below:



Name: Checkout
Allow Sorting □
Page Scroller □ Records 20
Allow Insert □ Insert
Grid Type Tabular

3. Click on the **SQL** tab and after checking the *Use SQL string* checkbox enter the following SQL script:

```
select member_id, sum(quantity*price) as sub_total from items, orders
where orders.item_id=items.item_id
group by member_id
```

The above script calculates the purchase total for the current orders by computing the sum of the product of order price and quantity (sum(quantity*price)). To do so, the script needs to know the member_id of the current user so as to extract the relevant records from the **orders** table. The member_id is obtained from a session variable called UserID which is always set when a user logs into the store.

4. Click on the **Input** tab and add a field with the following values:

Property Name	Value
Field	member_id
Variable Name	UserID
Type	Session
Data Type	Number
Operation	=
Required	Checked
Transfer	Checked

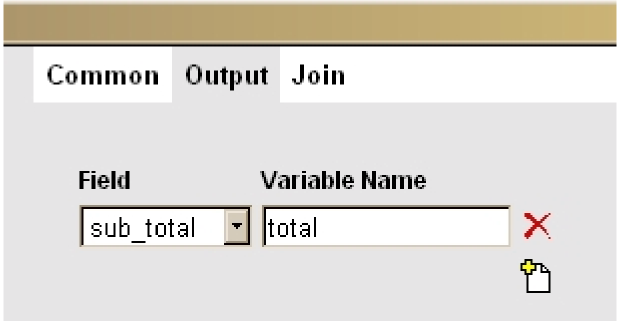
As noted above, this input field will provide the current member_id to the SQL script previously entered.

Click **Ok** to close the form properties window.

5. Under the Fields portion, add a blank field of type URL. Open its property dialog box and under the **Common** tab**,** enter the following values:

Property Name	Value
Name	Link
Default Value	Proceed to checkout...
Data Type	Text
Page Link	Leave blank for now
Use URL Field	Unchecked
URL Field	Blank
Html	Unchecked
Add Transit Params	Checked

6. Click on the **Output** tab then click on the 🗋 to add a new parameter. Add the sub_total field then assign it variable name 'total' as shown below. This serves to send the purchase total via the URL to the **PrepareCheckOut** page where it shall appear in a text box.



Field	Variable Name
sub_total	total

And that does it for the **ShoppingCart** page.

5.2 The PrepareCheckOut Page

The **PrepareCheckOut** page retrieves member information and assembles it in a html form together with some other information needed for credit card verification. Recall from the PayFlow Link users manual that when submitting information for verification, there are four required fields: LOGIN, PARTNER, AMOUNT and TYPE. As stipulated in the guide, the names of all the fields submitted must be in UPPERCASE. In addition to the four required fields, the form created here submits NAME, CARDNUM, EXPDATE, EMAIL, ADDRESS, CITY, STATE, ZIP, COUNTRY and DESCRIPTION fields as well.

Ideally, most of the values submitted through the form should be hidden but for the purpose of demonstration and easy manipulation, the field values are submitted here via textboxes.

However, before creating the submission form, two other forms shall be created to display member and order item information.

1. Add a new page and assign it the following properties:

Property Name	Value
Name	PrepareCheckOut
Title	Bookstore Check Out
Security Level	1- Low
Columnar Layout	Unchecked

2. To display member information, add a new form with the following properties:

Property Name	Value
Name	Member
Form Title	Member Information
Type	Record
Form Action	PrepareCheckOut
Table	members

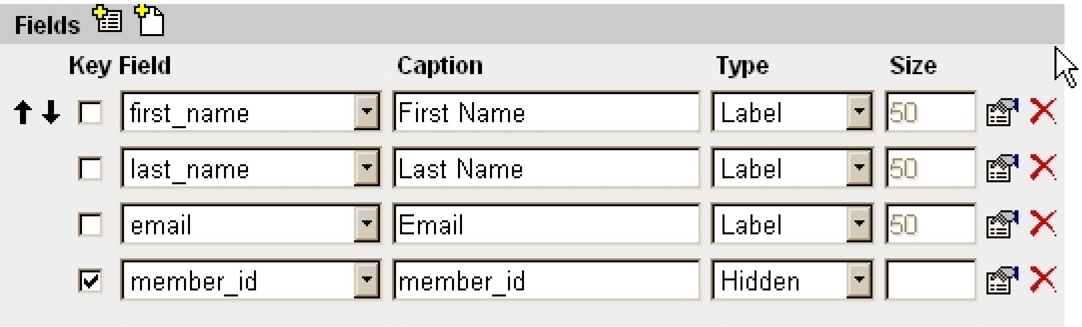
3. Open the form properties dialog box and under the **Common** tab uncheck all the checkboxes.

4. Click on the **Input** tab and add a field with the following values:

Property Name	Value
Field	member_id
Variable Name	UserID
Type	Session
Data Type	Number
Operation	=
Required	Checked
Transfer	Checked

Click **Ok** to close the form properties window.

5. Under the Fields Portion, add four fields as shown below:



Fields

Key	Field	Caption	Type	Size
□	first_name	First Name	Label	50
□	last_name	Last Name	Label	50
□	email	Email	Label	50
☑	member_id	member_id	Hidden	

6. To display item information, add a new form with the following properties:

Property Name	Value
Name	Items
Form Title	Ordered Items
Type	Grid
Form Action	Leave Blank
Table	Leave Blank

7. Open the form properties dialog box and under the **Common** tab uncheck all the checkboxes and leave the rest of the fields as they were.

8. Click on the **SQL** tab and after checking the *Use SQL string* checkbox enter the following SQL script:

```
select order_id, name, price, quantity, member_id, quantity*price as
sub_total
from items, orders where orders.item_id=items.item_id
order by order_id
```

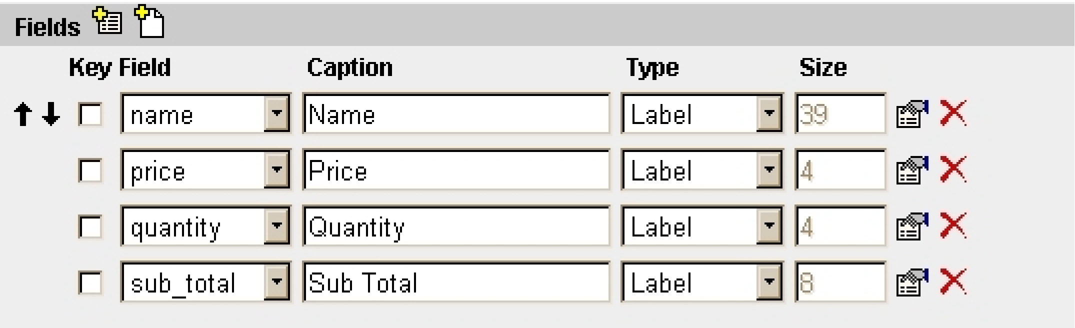
The above script retrieves the current members order items and calculates subtotals based on quantity ordered.

9. Click on the **Input** tab and add a field with the following values:

Property Name	Value
Field	member_id
Variable Name	UserID
Type	Session
Data Type	Number
Operation	=
Required	Checked
Transfer	Checked

Click **Ok** to close the form properties window.

10. Under the Fields Portion, add four fields as shown below:



Fields

	Key Field	Caption	Type	Size
□	name	Name	Label	39
□	price	Price	Label	4
□	quantity	Quantity	Label	4
□	sub_total	Sub Total	Label	8

The next form to be added contains the bulk of the information to be submitted for verification. You might recall from the **members** database table that the users name is stored in two separate fields entitled *first_name* and *last_name*. Since VeriSign requires the name to be sent in one variable, the two fields mentioned have to be concatenated using a “Before Show” event as we will see below.

11. Add a new form with the following properties:

Property Name	Value
Name	CCInfo
Form Title	Credit Card Information
Type	Grid
Form Action	Leave Blank
Table	members

12. Open the form properties dialog box and under the **Common** tab uncheck all the checkboxes and change the Grid Type to Columnar.

13. Click on the **Input** tab and add a field with the following values:

Property Name	Value
Field	member_id
Variable Name	UserID
Type	Session
Data Type	Number
Operation	=
Required	Checked
Transfer	Checked

14. Click on the Events tab and in the “Before Show” event, enter the following code:

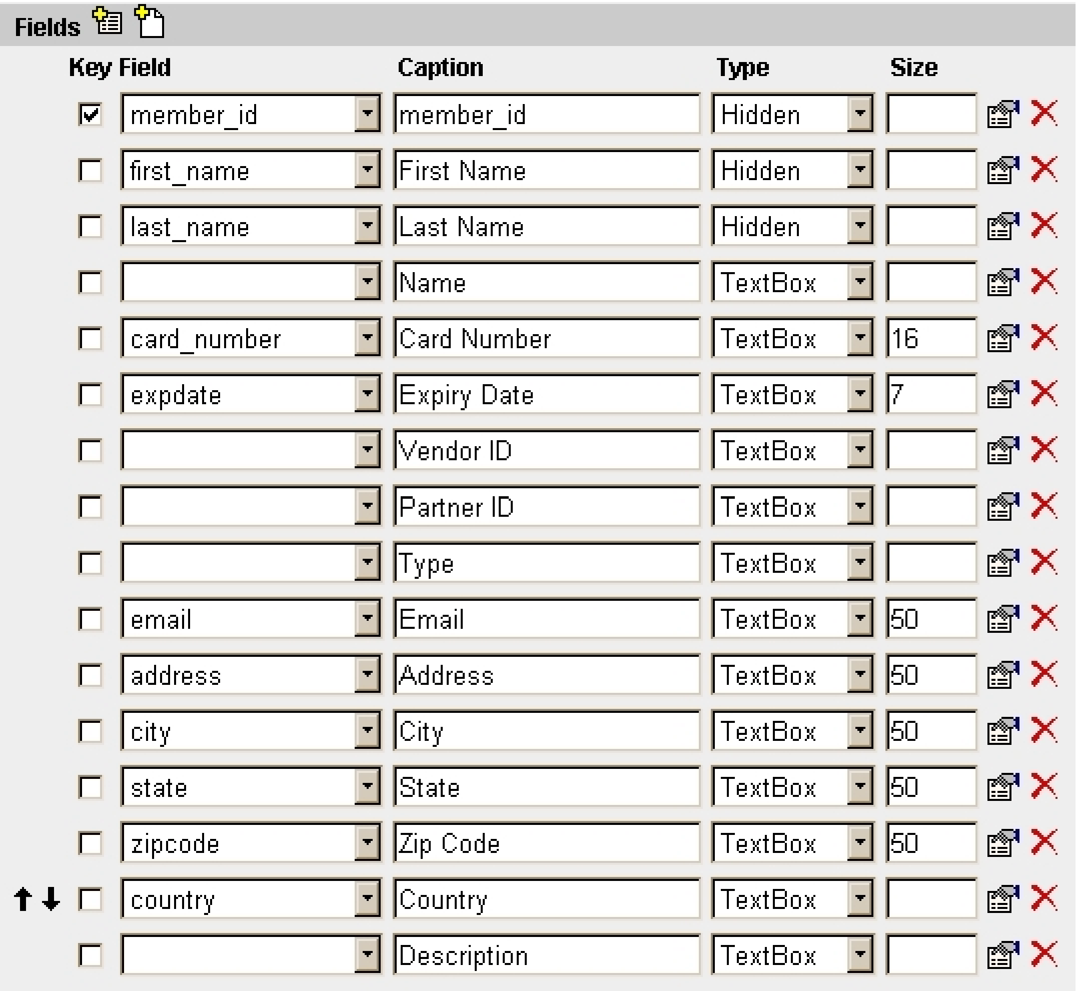
ASP	fldname = fldfirst_name & " " & fldlast_name
PHP	$fldNAME = $fldfirst_name." ".$fldlast_name;
Cold	<cfset fldname = fldfirst_name & ' ' & fldlast_name>

Fusion	

As you might have guessed, the above code concatenates the *first_name* and the *last_name* fields and stores the resultant value in the *name* field.

Click **Ok** to close the form properties window.

15. Under the Fields Portion, add the fields shown below:



Fields

Key	Field	Caption	Type	Size
☑	member_id	member_id	Hidden	
☐	first_name	First Name	Hidden	
☐	last_name	Last Name	Hidden	
☐		Name	TextBox	
☐	card_number	Card Number	TextBox	16
☐	expdate	Expiry Date	TextBox	7
☐		Vendor ID	TextBox	
☐		Partner ID	TextBox	
☐		Type	TextBox	
☐	email	Email	TextBox	50
☐	address	Address	TextBox	50
☐	city	City	TextBox	50
☐	state	State	TextBox	50
☐	zipcode	Zip Code	TextBox	50
↑↓ ☐	country	Country	TextBox	
☐		Description	TextBox	

16. Set the properties for the fields as follows:

Name	Default Value	Required
member_id	Blank	N/A
first_name	Blank	N/A
last_name	Blank	N/A
NAME	Blank	Yes
CARDNUM	Blank	Yes
EXPDATE	Blank	Yes
LOGIN	[your VeriSign login]	Yes
PARTNER	[your Partner id e.g. tsn]	Yes

TYPE	S	Yes
EMAIL	Blank	Yes
ADDRESS	Blank	Yes
CITY	Blank	Yes
STATE	Blank	Yes
ZIP	Blank	Yes
COUNTRY	Blank	Yes
DESCRIPTION	Purchase of textbooks	Yes

Note that apart from the member_id, first_name and last_name fields, all other field names have to be in uppercase.

17. Since the Grid form does not have a submit button, we have to manually wrap the above fields in a html form. Open the CCInfo form properties dialog box and under the **Header&Footer** tab, enter the following in the header section.

```
<form method="POST" action="https://payflowlink.verisign.com/payflowlink.cfm">
```

Click **Ok** to close the dialog box. The observant reader is probably wondering why there is no code for a submit button nor closing </form> tag. The reason is because we are still missing one of the required fields; the AMOUNT field. Getting the value for this field requires custom SQL which is located in the form to be created next. The submit button code and the closing </form> tag shall be located in the footer section of the next form.

18. Add a new form with the following properties:

Property Name	Value
Name	Total
Form Title	Leave Blank
Type	Grid
Form Action	Leave Blank
Table	Leave Blank

19. Open the form properties dialog box and under the **Common** tab uncheck all the checkboxes and change the Grid Type to Columnar.

20. Click on the **SQL** tab and after checking the *Use SQL string* checkbox enter the following SQL script:

```
select member_id, sum(quantity*price) as sub_total from items, orders
where orders.item_id=items.item_id
group by member_id
```

The above script calculates the purchase total for the current orders by computing the sum of the product of order price and quantity.

21. Click on the **Input** tab and add a field with the following values:

Property Name	Value
Field	member_id
Variable Name	UserID
Type	Session
Data Type	Number
Operation	=
Required	Checked
Transfer	Checked

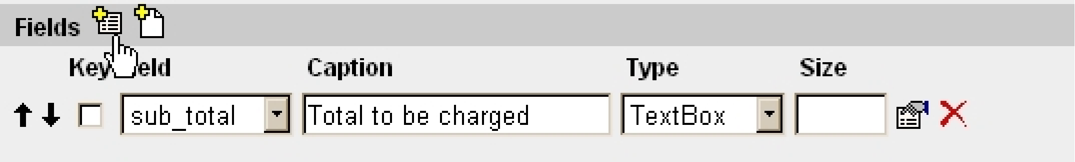
22. Click on the **Header&Footer** tab and enter the following in the footer section.

```
<input type="submit" value="Click here to Purchase"></p> </form>
```

The above code inserts a submit button and terminates the html form created in the CCInfo form header section.

Click **Ok** to close the dialog box.

23. Under the Fields portion, add a field as show below:

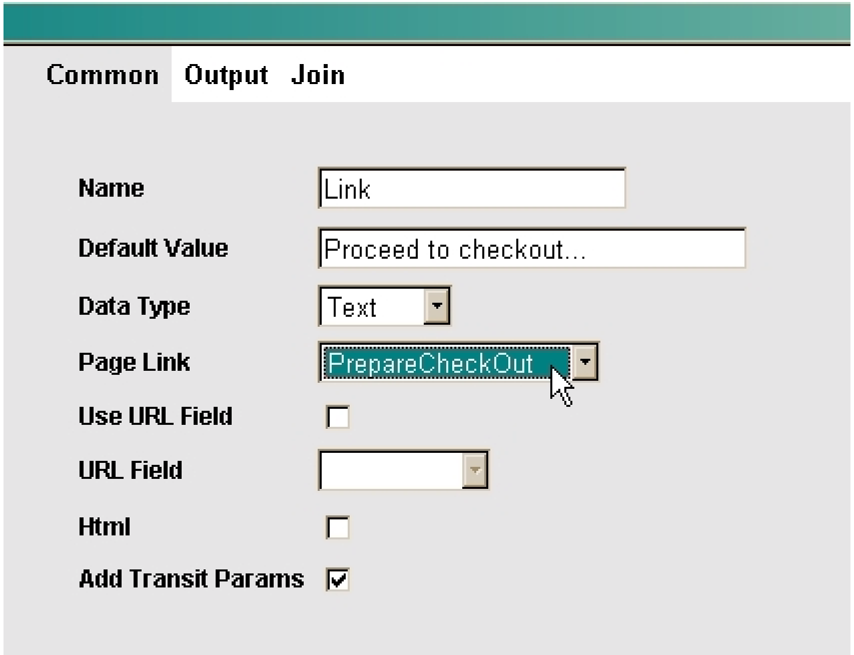


Fields

	Key	Field	Caption	Type	Size
↑↓	☐	sub_total	Total to be charged	TextBox	

24. Open the properties dialog box for the field above and check the “Required” checkbox. Set the name of the field to AMOUNT (uppercase). Click **Ok** to close the dialog box.

We are now done with the **PrepareCheckOut** page. Before generating the pages for testing, go back to the **ShoppingCart** page and do some patching up. You might recall that when we specified the URL link to the **PrepareCheckOut** page, we did not fill in the “Page Link” value. To rectify this, click on the **ShoppingCart** page and then highlight the **Checkout** form. In the fields portion of the page, open the properties dialog box of the URL field then specify **PrepareCheckOut** as the Page Link as shown below.



Name	Link
Default Value	Proceed to checkout...
Data Type	Text
Page Link	PrepareCheckOut
Use URL Field	□
URL Field	
Html	□
Add Transit Params	☑

At this juncture, you can generate the pages and you should be able to add some books to your shopping cart and submit them to VeriSign for verification. The next page to be created receives confirmation parameters from VeriSign and uses them to populate the **invoices** and **purchases** tables.

5.3 Creating the VerisignReturn Page

Upon receiving confirmation parameters from VeriSign, the **VerisignReturn** page displays the received response message (RESPMSG) and reference number (PNREF). If the transaction was successful (RESULT = 0), a new invoice is created using the reference number and at the same time, the ordered item records are copied to the **purchases** table and deleted from the **orders** table.

1. To begin with, add a new page and assign it the following properties:

Property Name	Value
Name	VerisignReturn
Title	Finalize Transcation Details
Security Level	**No authentication**
Columnar Layout	Unchecked

2. Add a new form with the following properties:

Property Name	Value
Name	Items
Form Title	Shopping Cart Items
Type	Grid
Form Action	Leave Blank
Table	Leave Blank

3. Open the form properties dialog box and under the **Common** tab uncheck all the checkboxes.

4. Click on the **SQL** tab and after checking the *Use SQL string* checkbox enter the following SQL script:

```
select order_id, name, price, quantity, member_id, quantity*price as
sub_total
from items, orders where orders.item_id=items.item_id
order by order_id
```

The above script retrieves the current members order items and calculates subtotals based on the quantity ordered.

5. Click on the **Input** tab and add a field with the following values:

Property Name	Value
Field	member_id
Variable Name	UserID
Type	Session
Data Type	Number
Operation	=
Required	Checked
Transfer	Checked

6. Click on the **Events** tab and under the *Open* event, enter the following code depending on your implementation language.

ASP	dim pnref dim result dim respmsg pnref= getParam("PNREF") result = getParam("RESULT") respmsg = getParam("RESPMSG") response.write "<font color = red>" & respmsg & "</font> " if result = 0 then cn.execute("insert into invoices (member_id, invoice_id) values (" & Session("UserID") & ", '" & pnref & "')") cn.execute("insert into purchases (invoice_id, order_id, member_id, item_id, quantity) select '" & pnref & "', order_id, member_id, item_id, quantity from orders where member_id = " & Session("UserID")) response.write "Thank you for shopping with us. For your records, the reference number for this transcation is:<font color = red>" & pnref & "</font> " else response.write "This transaction was not approved. Please adjust your payment details and try again." end if
Cold Fusion	<cfset pnref = #Form.PNREF#> <cfset result = #Form.RESULT#> <cfset respmsg = #Form.RESPMSG#>

	<CFOUTPUT>#respmsg# </CFOUTPUT> <CFIF result EQ 0> <CFQUERY DATASOURCE="Bookstore" NAME="InsertInvoice"> insert into invoices (member_id, invoice_id) values (<CFOUTPUT>#UserID#</CFOUTPUT>, <CFOUTPUT>#pnref#</CFOUTPUT>) </CFQUERY> <CFQUERY DATASOURCE="Bookstore" NAME="InsertPurchases"> insert into purchases (invoice_id, order_id, member_id, item_id, quantity) select <CFOUTPUT>'#pnref#'</CFOUTPUT>, order_id, member_id, item_id, quantity from orders where member_id = <CFOUTPUT>#UserID#</CFOUTPUT> </CFQUERY> <CFOUTPUT>Thank you for shopping with us. For your records, the reference number for this transcation is:<font color = red> #pnref# </font> </CFOUTPUT> <CFELSE> <CFOUTPUT>This transaction was not approved. Please adjust your payment details and try again.</CFOUTPUT> </CFIF>

PHP	$pnref= get_param("PNREF");
	$result = get_param("RESULT");
	$respmsg = get_param("RESPMSG");
	echo ("<font color = red>". $respmsg."</font>
	");
	if ($result == 0)
	{
	$db->query("insert into invoices (member_id, invoice_id) values (". get_session("UserID") . ", '". $pnref ."')");
	$db->query("insert into purchases (invoice_id, order_id, member_id, item_id, quantity) select '" . $pnref . "', order_id, member_id, item_id, quantity from orders where member_id = " . get_session("UserID"));
	echo ("Thank you for shopping with us.
	 For your records, the reference number for this transcation is:<font color = red>". $pnref ."</font>
	");
	}
	else
	{
	echo ("This transaction was not approved.
	 Please adjust your payment details and try again.");
	}

Click **Ok** to close the form properties dialog box.

Basically, the code above receives three parameters sent from VeriSign and if the transaction was successful (i.e. RESULT has a value of 0), a new invoice is created using the reference number (PNREF). At the same time, the ordered item records are copied to the **purchases** table from the **orders** table. The response message (RESPMSG) is also displayed, along with a message containing the reference number (PNREF).

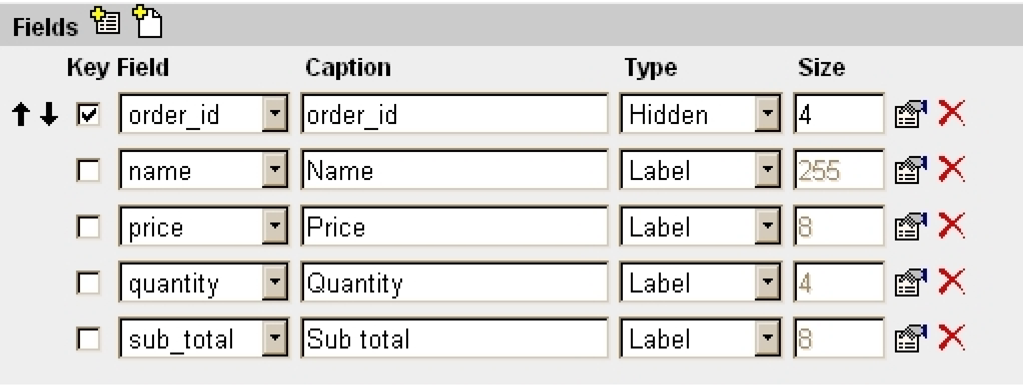
7. Once the order records have been copied to the **purchases** table, they need to be deleted from the **orders** table. Since the details of the ordered items need to be displayed in the current form, the script that does the deletion is placed in the *Close* event of the page. Open the **Page** Properties dialog box and enter the following code in the *Close* event.

ASP	: result = 0 then n.execute("delete from orders where member_id = " & Session("UserID")) nd if
Cold Fusion	<CFIF result EQ 0> <CFQUERY DATASOURCE="Bookstore" NAME="DeleteOrders"> delete from orders where member_id = <CFOUTPUT>#UserID#</CFOUTPUT> </CFQUERY> </CFIF>

PHP	if ($result == 0)
	{
	$db->query("delete from orders where member_id = " . get_session("UserID"));
	}

Click **Ok** to close the form properties dialog box.

8. Under the Fields portion, add the following fields:



Fields

	Key	Field	Caption	Type	Size
↑↓	☑	order_id	order_id	Hidden	4
	□	name	Name	Label	255
	□	price	Price	Label	8
	□	quantity	Quantity	Label	4
	□	sub_total	Sub total	Label	8

9. Follow the procedure used in section 5.2 steps 2-5 to create a Member form noting the following difference.

- Form Action should be **VerisignReturn** not **PrepareCheckOut**

10. Add a new form with the following properties:

Property Name	Value
Name	Total
Form Title	Leave Blank
Type	Grid
Form Action	Leave Blank
Table	Leave Blank

11. Open the form properties dialog box and under the **Common** tab uncheck all the checkboxes.

12. Click on the **SQL** tab and after checking the *Use SQL string* checkbox enter the following SQL script:

```
select member_id, sum(quantity*price) as sub_total from items, orders
where orders.item_id=items.item_id
group by member_id
```

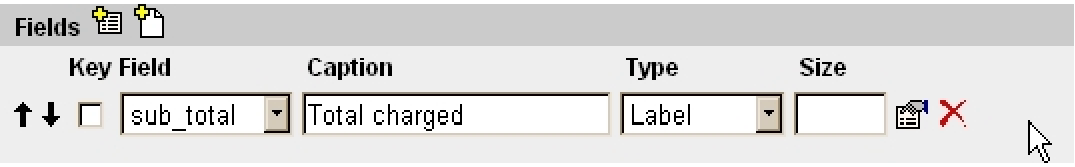
The above script calculates the purchase total for the purchased items by computing the sum of the product of order price and quantity.

13. Click on the **Input** tab and add a field with the following values:

Property Name	Value
Field	member_id
Variable Name	UserID
Type	Session
Data Type	Number
Operation	=
Required	Checked
Transfer	Checked

Click **Ok** to close the dialog box.

14. Under the Fields portion, add a field as show below:



Fields

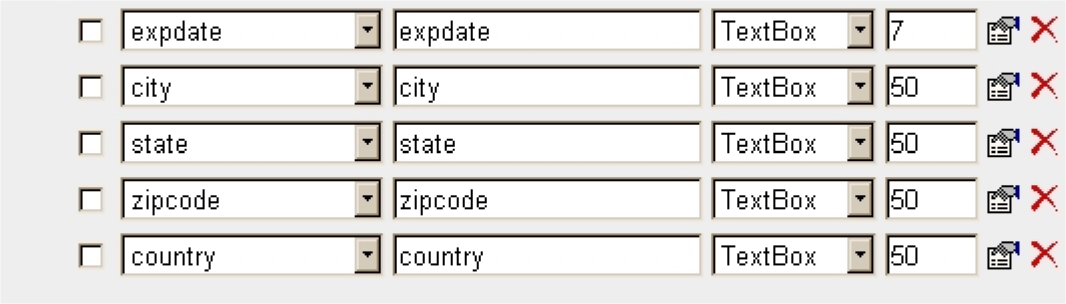
Key Field		Caption	Type	Size
☐	sub_total	Total charged	Label	

And that does it for the **VerisignReturn** page.

5.4 Adjusting the MyInfo Page

At the beginning of this tutorial, recall that we added some fields to the **members** table in the database. To enable online administration of these additional fields, the **MyInfo** page needs to be adjusted to allow editing of these fields.

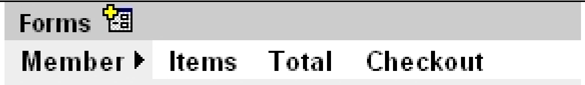
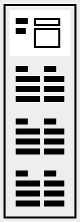
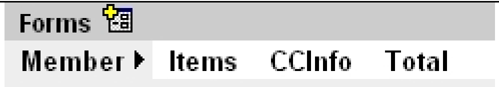
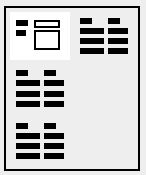
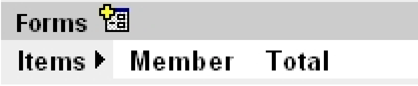
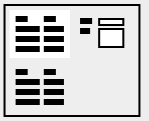
1. Click on the MyInfo page from among the list of pages.
2. Add the following fields to the ones that already exist.



□	expdate	expdate	TextBox	7
□	city	city	TextBox	50
□	state	state	TextBox	50
□	zipcode	zipcode	TextBox	50
□	country	country	TextBox	50

6.0 Layout

Considering that we added quite a number of forms to the two pages we added, it would be wise to adjust the layout of these forms to make them more presentable and in line with the rest of the pages. You are at liberty to rearrange the forms as you wish but the following scheme is provided as a sample scheme.

ShoppingCart	 Forms Member ▸ Items Total Checkout	
PrepareCheckOut	 Forms Member ▸ Items CCInfo Total	
VeriSignReturn	 Forms Items ▸ Member Total	

7.0 Conclusion

That brings us to the end of this tutorial. You should now be able to submit order information to VeriSign and receive feedback. If you run into problems, go through the configuration of the fields and make sure they are exactly as presented here. If you are using PHP, be aware of the capitalization of the fields. For instance, if you enter the UserID variable as userID, your code will not work correctly.

As mentioned at the beginning, this tutorial is meant to get you on your way towards implementing e-commerce solutions. However, the subject of e-commerce encompasses many more details than are covered in this tutorial. Due to the volatility of the subject, you are encouraged to seek further information before implementing a full-scale system.