



# ***NebuCart Installation & Configuration Guide***

## Index

Index .....	2
Copyright.....	3
Terms of Usage .....	3
Support.....	3
Troubleshooting .....	3
Internet Explorer .....	3
Netscape Navigator .....	3
Cart Files.....	4
Cart Configuration.....	4
Cart Structure.....	4
Catalog Pages .....	5
Summaries: .....	6
Basic HTML Setup .....	6
Summary: .....	7
Product Quantity.....	7
Summary: .....	8
Purchase Limit.....	8
Summary: .....	8
Product Description .....	8
Summary: .....	9
Product Options.....	9
Simple Option List.....	9
Additional Options.....	9
Multiple Select Options .....	10
Cost Additive Options .....	10
Summary: .....	10
Cart View Page .....	10
Checkout Page .....	11
Unsecure Order Setup .....	13
Printable Order Form.....	13
Print Verification.....	14
Printable Order Page.....	15
Unsecure Form Order .....	15
Configuration Notes .....	15
Form Order Page.....	17
Security, AGAIN.....	17
Unsecure Credit Card Gateway Setup.....	17
Configuration Notes .....	18
Gateway Order Page.....	19
Back-Ends .....	20
Security, Yet AGAIN .....	20

## Copyright

NebuCart, the NebuCart logo, and all files comprising the NebuCart E-commerce System are copyright 1999-2001 Nebulus Designs, all rights reserved. Any reproduction or copying of said files are strictly forbidden without prior consent from the author.

## Terms of Usage

- NebuCart may be used as a single seat license (1 user/site only)
- These files may not be redistributed for profit
- These files may not be used by any party other than you
- No JavaScript module may be modified, with exception off the config file
- These files may not be offered for download on any other site except Nebulus Designs
- Any violation of this agreement makes you liable for damages or loss of income.

## Support

NebuCart Demo comes with limited [email](#) support. Please refer to the installation guide, demo files, or NebuCart [forum](#) for further help. When requesting help via email, please include your NC\_settings.js file and an exact listing of the errors and alerts that you are experiencing.

## Troubleshooting

During the course of installation or install, you may encounter various errors in the cart. Depending on the error you encounter, various parts of the cart may not display correctly, or possibly not display at all. Below are the methods you can use to debug your installation in either browser. For both browsers, you should back up the demo cart files just in case you need a fresh file to restore.

### Internet Explorer

If you see an alert symbol in the lower left corner of the browser window, double click it to view the error message. This error message should give you an overall description of the error that occurred, and a line number. You can open the files in question and check the lines referenced by the error message.

If you can download the Microsoft Script Debugger, it is by far the easiest method for debugging your cart installation.

### Netscape Navigator

Netscape is the easiest browser to use for debugging. I strongly suggest that you download a copy of Netscape just for the purpose of debugging JavaScript.

If you encounter an error in Netscape, you will see a window status notice that there was a JavaScript error. To view the error, type "JavaScript:" (without quotes) in the location bar and hit the Enter key. The Netscape JavaScript Debugger Window will then be launched with a complete listing of the error, description, line number, and the JavaScript modules in which the error occurred.

Good Luck!

## Cart Files

Below is a listing of what files are included with your NebuCart demo:

### JavaScript Files (\*.js extension)

NC\_buildcredit  
NC\_buildcustomer  
NC\_buildform  
NC\_detect  
NC\_finalbuttons  
NC\_formatting  
NC\_getshipping  
NC\_orderoptions  
NC\_settings  
NC\_shipoptions  
NC\_settings  
NC\_viewcart  
NebuCart

### HTML Files (\*.html extension)

cart  
cubes  
customer\_data  
formorder  
gateway  
index  
printorder  
printverify  
spheres

## Cart Configuration

The NC\_settings.js file is the only JavaScript file you will need to edit during your implementation of the NebuCart system. Make sure you make a backup copy of the file just in case!

The NC\_settings.js file is simply a text file with a .js extension, and can be edited by any text editor such as NotePad, or more preferably, UltraEdit. The settings are very straightforward and are commented extensively in the NC\_settings.js file. Reading the comments provided is conducive to a working cart!

In order to test the cart with a form handling CGI script, change the variables "unsecurePostAction" and "unsecureGatewayAction" in NC\_settings.js to point to the CGI script on your web account. You may also need to edit the variable "extraFormTags" to add the proper tags required by your script.

*Note: This is the install documentation for the NebuCart Demo, so it contains no information about secure ordering setup and execution. The install docs for the full version are included with the full version.*

## Cart Structure

The NebuCart E-commerce System is comprised of 13 JavaScript "modules" and supporting HTML files. With the inclusion of three of the modules in any page, that page will become enabled with the NebuCart system. These modules are NC\_settings.js, NebuCart.js, and NC\_formatting.js.

Since including these files is all that's required to enable a page with NebuCart, it can very easily be used to enable existing pages with little or no impact. With the inclusion of these files, you will also have access to all the cart variables and functions through either the cart scripts or custom scripts. This makes it very easy to customize your storefront to your needs!

Below is an examples of a very basic HTML page that is NebuCart enabled:

```
<HTML>
<HEAD>
<!-- include the NebuCart Settings -->
<script language=JavaScript src="path/to/js_files/NC_settings.js">

<!-- include the NebuCart Engine -->
<script language=JavaScript src="path/to/js_files/NebuCart.js">

<!-- include the NebuCart Formatting Functions-->
<script language=JavaScript src="path/to/js_files/NC_formatting.js">
</HEAD>

<BODY>
This page is now NebuCart Enabled!
</BODY>
</HTML>
```

This code setup is necessary for every page that will be NebuCart enabled. After adding the three base files to a page, you may include other NebuCart modules in the page body in almost any configuration or combination. See the other sections of the documentation for specific page configurations.

## Catalog Pages

Catalog pages can be created dynamically by an ASP, PHP, CGI, or some other server based script, or they may be static. There is no limit to the number of catalog pages or items that NebuCart can handle.

Products in NebuCart have six properties, although more can be added with customization: *ID*, *Price*, *Quantity*, *Description*, *Limit*, and *Options*. Shipping Weight may be added at a later date. The only required properties for adding, updating, and deleting an item from your cart is *ID*, *Price*, and *Quantity*.

If you have an HTML editor and/or a good working knowledge on setting up forms, creating product pages should be fairly easy. Of course, the registered version of NebuCart includes a Catalog Builder wizard... The following instructions will walk you through the creation of a product for a fictitious clothing store, with details on how to set up each one of these fields.

## Summaries:

For each Product property, there s a summary at the bottom of the page. This summary gives you a listing of what is required in implementing each property:

Product Property	A description of the property
Field Suffix	when creating a product, you will use the product's ID as a base for all the properties. The Field Suffix is the text used to denote the property type for that product. Example: Product Price would be denoted by Product_ID_price where "Product_ID" is the alphanumeric value of that product's ID.  NOTE: each field must use the product ID's base value in the name, and each individual item must use a unique name. Product ID's may not be numeric only, but include at least one alphanumeric character.
Field Type	What type of HTML form field may be used
Required	Denotes whether the field is optional or required for proper cart operation
Value(s)	Allowable values for the Field Type
Code	The HTML/JavaScript for the Field Type
Operation	Denotes any extra events or operations that may be used for this property (eg: Add items, update items, etc)
Object Type	The object
Link/Method/Code	The allowable code for performing the listed Operation by the listed Object Type

## Basic HTML Setup

The easiest way to learn how to create catalog items is to jump right in. Shall we? Let's assume that you're setting up a storefront for your clothing company. We'll walk through the process of creating the tags needed for selling a shirt with a product ID of "NC\_shirt".

This is the basic code required for creating a page that is enabled as a catalog with a single example item with a product ID of "NC\_Shirt":

```
<HTML>
<HEAD>
<!-- include the NebuCart Settings -->
<script language=JavaScript src="path/to/js_files/NC_settings.js">

<!-- include the NebuCart Engine -->
<script language=JavaScript src="path/to/js_files/NebuCart.js">

<!-- include the NebuCart Formatting Functions-->
<script language=JavaScript src="path/to/js_files/NC_formatting.js">
</HEAD>

<BODY>
<!-- Always include the NebuCart form tag in catalog pages! -->
<FORM NAME="NC_form">

<!-- here is the basic item -->
Quantity: <input type="text" name="NC_Shirt">
<a href="javascript:AddItem('NC_Shirt')">Add to Cart</a>
<input type="hidden" name="NC_Shirt_price" value="19.95">

</FORM>
<!-- always close your form tags -->
</BODY>
</HTML>
```

The resulting page would display a text field for quantity and a link to add the item to your cart. As you can see by the field "NC\_Shirt\_price", the price of the shirt is \$19.95. Based on your configuration, the currency notation may be different. Upgrades for configurable decimal delimiter are in progress.

### Summary:

Product Property	Field Suffix	Field Type	Required	Value(s)
Price	_price	hidden	yes	number > 0
Operation	Object Type	Link/Method/Code		
Add Item to Cart	Image/Txt Link	<a href=" javascript:AddItem('Product_ID')">object</a>		
Add Item to Cart	Form Button	<input type="button" value="Add" onClick="AddItem('Product_ID')">		

## Product Quantity

There may be instances where you want to customize the Quantity field. You can create Quantity fields with any of these form inputs:

- Empty text field waiting for input
- Text field with a default value
- Drop down menu with values from 1 to N (N being the last number in the select menu)

The only requirement for Quantity fields is that you use numerical values greater than zero. If you use an empty text box, then the customer will be required to enter a numerical value. If they don't enter a value or it is not a number, they will get a pop up alert asking for a correct value.

Below are how each Quantity configuration would look:

Empty text field (notice field size control)

```
Quantity: <input type="text" name="NC_Shirt" size="3">
<a href="javascript:AddItem('NC_Shirt')">Add to Cart</a>
<input type="hidden" name="NC_Shirt_price" value="19.95">
```

Text field with a default value (notice field size control)

```
Quantity: <input type="text" name="NC_Shirt" size="3" value="1" >
<a href="javascript:AddItem('NC_Shirt')">Add to Cart</a>
<input type="hidden" name="NC_Shirt_price" value="19.95">
```

Hidden field with a default value

```
<a href="javascript:AddItem('NC_Shirt')">Add to Cart</a>
<input type="hidden" name="NC_Shirt" value="1">
<input type="hidden" name="NC_Shirt_price" value="19.95">
```

Drop down menu (offers the choice between 1 to 3 shirts)

```
Quantity:
<select name="NC_Shirt">
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
</select>
<a href="javascript:AddItem('NC_Shirt')">Add to Cart</a>
<input type="hidden" name="NC_Shirt_price" value="19.95">
```

**Summary:**

Product Property	Field Suffix	Field Type	Required	Value(s)	Code
Quantity	none	text	yes	none or number > 0	<input type="text" name="Product_ID"> or <input type="text" name="Product_ID" value="N">
Quantity	none	hidden	yes	number > 0	<input type="hidden" name="Product_ID" value="N">
Quantity	none	select	yes	number > 0	<select name="Product_ID"> <option value="1">1</option> ... <option value="N">N</option> </select>
Operation	Object Type	Link/Method/Code			
N/A	N/A	N/A			

**Purchase Limit**

If we wanted to limit the number of shirts the customer could buy to say, 5, we'd add an additional field like this:

```
<input type="hidden" name="NC_Shirt_limit" value="5">
```

The only requirement on *Product Limit* is that it be a numerical value greater than zero. If a customer tries to purchase more than the allowed limit, they will get a prompt notifying them of the limit, and the maximum allowable number of that item will be added to their cart.

**Summary:**

Product Property	Field Suffix	Field Type	Required	Value(s)	Code
Limit	_limit	hidden	no	number > 0	<input type="hidden" name="Product_ID_limit" value="N">
Operation	Object Type	Link/Method/Code			
N/A	N/A	N/A			

**Product Description**

When a customer adds an item to their cart, the cart view page will display their selections. It's always helpful to have a description for your product, so let's add that field:

```
<input type="hidden" name="NC_Shirt_desc" value="MyStore Super Jersey">
```

Now when they add the item to the cart, the description "MyStore Super Jersey" will be displayed along with the product ID.

**Summary:**

Product Property	Field Suffix	Field Type	Required	Value(s)	Code
Description	_desc	hidden	no	any alpha-numeric value	<input type="hidden" name="Product_ID_desc" value="description">
Operation	Object Type	Link/Method/Code			
N/A	N/A	N/A			

## Product Options

Most products have options, whether it be size, color, add-ons, etc. The NebuCart system can support up to 20 option lists per product, but it is unlikely that you will use that many for a single product.

### Simple Option List

To make an option list as space efficient as possible, NebuCart supports only select, or drop down menus for options. To create a simple list of size options for our example shirt (product ID "NC\_Shirt"), the code would look like this:

```
<select name="NC_Shirt_opt">
<option value="Small">Small</option>
<option value="Med">Medium</option>
<option value="Lrg">Large</option>
<option value="XLrg">Xtra Large</option>
</select>
```

Before the customer adds the shirt to their cart, they have the option of picking a size. By default, the first item in a select list is selected should the customer not pick an option. You can change which item is selected by default by adding the text "selected" in the option tag of choice.

Let's say the customer picked a medium shirt and added it to their cart. When the cart displays, the option "Med" would be displayed in the options column for that item.

Now if the customer returns to that catalog page and selects a different option, say a Small shirt, and adds it to their cart, the cart will display both a small and medium shirt in the cart.

### Additional Options

Now, let's assume that you're not only selling different sized shirts, but different colors as well. To add an option list for this, the code would be:

```
<select name="NC_Shirt_opt1">
<option value="White">White</option>
<option value="Tan">Tan</option>
<option value="Green">Red</option>
<option value="Black">Black</option>
</select>
```

Easy right? You'll notice that we named this added option list "NC\_Shirt\_opt1". This is to give it a unique name so that NebuCart can use it. For each additional option list for a given product, you'll need to increment the Field Suffix by 1. Otherwise, errors will occur.

So if the customer selected a Small, Tan shirt and added it to their cart, the options column would reflect those option choices.

### Multiple Select Options

You may have a case where you can give the customer the option of picking more than one option from a single list. This is easily done by adding the *multiple* property to the select menu. This is a property of HTML drop downs that allow a user to select more than one item from a list. When you create a multiple drop down, it's a good idea to add a display size (`size="N"`) to the select as well so that they can see more than one choice at a time. Here's how a multiple select definition would look:

```
<select name="NC_Shirt_opt" size="3" multiple>
```

### Cost Additive Options

Some times, options can cost you, the merchant, extra. So why not pass on the value? Let's take a look at our size option for the product *NC\_Shirt*. It may be that you want to charge the customer an extra \$2.95 for picking the extra large shirt. In that case, all you need to do is add that cost to the value property of that option like so:

```
<option value="XLrg|2.95" >Xtra Large (Add $2.95)</option>
```

Now when the customer picks the extra large shirt, the cost of the shirt (originally \$19.95) would be \$22.90! It's that easy. One note: The customer should always be aware of any additional costs to the product, so make sure that you put the added cost either in the text of the option or that it's readily viewable somewhere on the catalog page!

### Summary:

Product Property	Field Suffix	Field Type	Required	Value(s)	Text	Code
Option	_opt, _opt1 ... _optN (N = number from 1 to 20)	select single	no	Any text for option description. Options that add cost must be formatted like this: description cost	Any alphanumeric	<select name="Product_ID_opt"> <option value="Option Text">Option Text</option> <option value="Option Text Added_Cost">Option Text (Add \$Added_Cost)</option> </select>
Option	_opt, _opt1 ... _optN (N = number from 1 to 20)	select multiple	no	Any text for option description. Options that add cost must be formatted like this: description cost	Any alphanumeric	<select name="Product_ID_opt" size="N" multiple> <option value="Option Text">Option Text</option> <option value="Option Text Added_Cost">Option Text (Add \$Added_Cost)</option> </select> (N = the display size of the multiple select)
Operation	Object Type	Link/Method/Code				
N/A	N/A	N/A				

## Cart View Page

The Cart View page, as with any e-commerce package, is critical to the NebuCart system. This page allows the customer to view a detailed listing of purchased items, quantities, and a running total of the current session.

Below is the code for a basic cart view page:

```
<HTML>
<HEAD>
<!-- include the NebuCart Settings -->
<script language=JavaScript src="path/to/js_files/NC_settings.js">

<!-- include the NebuCart Engine -->
<script language=JavaScript src="path/to/js_files/NebuCart.js">

<!-- include the NebuCart Formatting Functions-->
<script language=JavaScript src="path/to/js_files/NC_formatting.js">
</HEAD>

<BODY>
<!-- include the Cart View script -->
<script language=JavaScript src="path/to/js_files/NC_viewcart.js">
</BODY>
</HTML>
```

The resulting page would display a formatted table depicting the customer's current cart contents. This page is displayed any time a customer adds an item to their cart from any catalog page, or if a link to the page is clicked. If the *supressCart* variable is set to "true" in the NC\_settings file (registered version only), the cart will not display when an item is added, but a pop up alerting the customer to the addition of the item will be shown.

If the cart view script is on the cart page designated in the NC\_settings.js file, the customer will have the option to update and delete items, and will also be presented with buttons for continued shopping, checkout, or full cart delete.

If the cart script is not on the designated cart page, it will only display a static listing of the cart contents with shipping, tax (if applicable), subtotal, and grand total.

If the cart is empty in either case, the customer will see "You Cart is Empty" with a "back" button.

The only requirement to displaying the cart view is that it be outside any form tags.

## Checkout Page

The checkout process in NebuCart is broken up into two phases. The first phase is the customer information and shipping option collection. From there, the customer may select his or her checkout option (printable form, secure/unsecure form, secure/unsecure credit gateway) taking them to the second phase of checkout. Typically, the checkout page displays the fields for collecting the customer's name, phone number, email, and other pertinent shipping info, as well as alternate shipping information, and their choice of shipping method.

Below is the basic HTML code for creating the first phase checkout page:

```
<HTML>
<HEAD>
<!-- include the NebuCart Settings -->
<script language=JavaScript src="path/to/js_files/NC_settings.js">

<!-- include the NebuCart Engine -->
<script language=JavaScript src="path/to/js_files/NebuCart.js">

<!-- include the NebuCart Formatting Functions-->
<script language=JavaScript src="path/to/js_files/NC_formatting.js">
</HEAD>

<BODY>
<!-- Always include the NebuCart form tag in catalog pages! -->
<FORM NAME="NC_form">

<!-- include the Cart Detection script-->
<script language=JavaScript src="path/to/js_files/NC_detect.js.">

<!-- include the Customer Info script-->
<script language=JavaScript src="path/to/js_files/NC_getshipping.js">

<!-- include the Shipping Options script-->
<script language=JavaScript src="path/to/js_files/NC_shipoptions.js">

<!-- include the Order Oprions script-->
<script language=JavaScript src="path/to/js_files/NC_orderoptions.js">

</FORM>
<!-- always close your form tags -->
</BODY>
</HTML>
```

The script `NC_detect.js` performs a simple check to see if the customer's cart is empty. If it is, then they will see the notification on the page that the cart is empty.

When the customer enters their billing/contact information, it gets stored in a cookie for the duration set by the variable `customerTime` in the settings file. Note that credit card info is not available or stored on this unsecured page.

If you want to suppress the alternate shipping information, set the `getAltShipping` variable to "false" and it will not be displayed on the checkout page.

The shipping options script uses the `shipOptions` array from the settings file to generate a radio button list of the available shipping options. This list only gets rendered if the `useShipOptions` variable is set to "true". Otherwise, it writes nothing.

The order options script will provide the customer with their available checkout options. These are configured in the settings file.

The only requirement to displaying the checkout page is that all scripts be inside the NebuCart form tags.

## Unsecure Order Setup

On the first checkout page, the customer should be presented with up to five ordering options:

- Printable Form
- Unsecure Order Form
- Unsecure Credit Card Gateway
- Secure Order Form
- Secure Credit Card Gateway

The demo version of NebuCart does not support the secure ordering options, so they will not be discussed in this document. Instructions for secure ordering options are available in the registered version.

### Security

You should be aware that unsecure order options, aside from the printable form, are not widely accepted by online shoppers. If you do not offer secure ordering options, you may be losing business. Should you decide to offer unsecure ordering, you become liable for that customer's information!

### Enabling/Disabling Unsecure Order

By default, the NebuCart demo is configured to offer unsecure ordering options. This is done with the variable *useUnsecure* in the settings file. Setting it to "false" disables unsecure ordering options, and they will not be displayed on the checkout page.

### Enabling/Disabling Secure Order

By default, the NebuCart demo is configured to offer secure ordering options, though the demo cart engine does not support secure ordering. This is done with the variable *useSecure* in the settings file. Setting it to "false" disables secure ordering options, and they will not be displayed on the checkout page.

Should both ordering options be set to "false", the customer will see a configuration error with a link to your email address.

### *Printable Order Form*

When the customer chooses to use the printable order form option, they are routed to an order verification page so that they can verify their billing/shipping information, as well as their order before printing. If the information is correct, they can then select the "Print Order" button and be forwarded to a print ready page.

## Print Verification

Below is the basic code for a print verification page:

```

<HTML>
<HEAD>
<!-- include the NebuCart Settings -->
<script language=JavaScript src="path/to/js_files/NC_settings.js">

<!-- include the NebuCart Engine -->
<script language=JavaScript src="path/to/js_files/NebuCart.js">

<!-- include the NebuCart Formatting Functions-->
<script language=JavaScript src="path/to/js_files/NC_formatting.js">
</HEAD>

<BODY>
<!-- Always include the NebuCart form tag in catalog pages! -->
<FORM NAME="NC_form">

Thanks! Please verify that the information below is correct before
proceeding to
final checkout. If you need to make some changes to your cart or alter
your
shipping/billing information, please do so now. If not, let's
continue...

<!-- include the Customer Information script-->
<script language=JavaScript src="path/to/js_files/NC_buildcustomer.js">

<!-- include the Cart View script-->
<script language=JavaScript src="path/to/js_files/NC_viewcart.js">

If you continue, the resulting page will be a printable order form.
Just select File > Print from your browser's toolbar to print out your
form.
Once you've printed the order form, please include the full amount of
the
order via check or money order and send it to:

<!-- include the Merchant Information script-->
<script language=JavaScript src="path/to/js_files/NC_buildlogo.js">

Thanks!

<!-- include the Order Buttons script-->
<script language=JavaScript src="path/to/js_files/NC_finalbuttons.js">

</FORM>
<!-- always close your form tags -->
</BODY>
</HTML>

```

## Printable Order Page

Below is the basic code for a print verification page:

```
<HTML>
<HEAD>
<!-- include the NebuCart Settings -->
<script language=JavaScript src="path/to/js_files/NC_settings.js">

<!-- include the NebuCart Engine -->
<script language=JavaScript src="path/to/js_files/NebuCart.js">

<!-- include the NebuCart Formatting Functions-->
<script language=JavaScript src="path/to/js_files/NC_formatting.js">
</HEAD>

<BODY>

<!-- include the Merchant Information script-->
<script language=JavaScript src="path/to/js_files/NC_buildlogo.js">

<!-- include the Customer Information script-->
<script language=JavaScript src="path/to/js_files/NC_buildcustomer.js">

<!-- include the Cart View script-->
<script language=JavaScript src="path/to/js_files/NC_viewcart.js">

</BODY>
</HTML>
```

For printable forms, you may want to keep the page as simple and graphic free as possible. A clean order form makes for a happy customer! If you would like to make the printable order page automatically attempt to print, add this <body> tag to the document:

```
<body onLoad="javascript:window.print()">
```

## Unsecure Form Order

When the customer chooses to use the unsecure form option, they are routed to the form order page (defined by the *COform* variable in the settings file) so they can enter their credit card information and finalize/submit their order.

It is interesting to note that the unsecure form order page can not only post data to an email handling form, but to any script that can handle a form post. The possibilities for back-end scripting is virtually endless! To get an idea of the many kinds of scripts that can be utilized, check out the CGI Resource Index. We suggest that you experiment with different scripts as a back-end to the NebuCart system. If you find a nice script that can be used as another back-end, please let us know!

### Configuration Notes

In the *NC\_settings* file, there are several variables geared specifically towards form posting. The following info should help you get your form order working quickly.

#### *unsecurePostAction*

Set the *unsecurePostAction* variable to a valid URL of a script that can handle a form post. NebuCart comes configured to post data to the Matt's Script Archive script, formmail.cgi since it is

so widely used (and probably already installed on your account!). If you point to your installation of the script, you should be able to submit orders via email. A good example is:

```
var unsecurePostAction = 'http://www.yourdomain.com/cgi-bin/formmail.cgi';
```

#### *extraFormTags*

This variable is an array that can hold any fields that may be required by your script to operate. For instance, formmail.cgi requires the fields *recipient*, *subject*, *redirect*, and *return\_link\_title* in order to function properly. Since the *NC\_buildform.js* script is built to create generalized forms, it's better to leave specialized, non cart related form fields out of the script and add them when the time comes.

When the buildform.js file is executed, it creates the form tag with the action equal to the value of *unsecurePostAction*, then immediately writes all the tags in the extraFormTags array. After that, it creates all the form tags relevant to the NebuCart system. An example of a formmail.cgi specific *extraFormTags* array is below:

```
var extraFormTags = new Array(
  '<input type="hidden" name="recipient" value="sales@yourdomain.com">',
  '<input type="hidden" name="subject" value="YourStore Sample Order">',
  '<input type="hidden" name="redirect"
value="http://www.yourdomain.com/thanks.html">',
  '<input type="hidden" name="return_link_title" value="Back to
YourStore.com">
  ');
```

If your script doesn't require any special tags, then you can leave this array empty and no extra tags will get rendered.

#### *cgiMailField*

This is the last of the form order specific fields. the variable *cgiMailField* holds the name of the field that form handling script uses as the recipient of the emailed order, you.

Form handling scripts can email a form to more than one person if the recipient field contains more than one email address. *cgiMailField* is used by the *NC\_buildform* script to insert the customer's email address into that field so that they can receive a copy of their order.

The formmail.cgi script uses a field called *recipient* from which to harvest the target email address. So the proper use of *cgiMailField* in this case would be:

```
var cgiMailField = 'recipient';
```

Of course, if you're using a different back-end, won't be copying the customer on the email, or are using gateway processing, you can set *cgiMailField = ''*

## Form Order Page

Once you've configured your settings for form posting to your particular script, the form order page will handle passing the order information to your script. Below is the basic code for a form order page:

```
<HTML>
<HEAD>
<!-- include the NebuCart Settings -->
<script language=JavaScript src="path/to/js_files/NC_settings.js">

<!-- include the NebuCart Engine -->
<script language=JavaScript src="path/to/Js_files/NebuCart.js">

<!-- include the NebuCart Formatting Functions-->
<script language=JavaScript src="path/to/Js_files/NC_formatting.js">
</HEAD>

<BODY>

<!-- include the Form Builder script-->
<script language=JavaScript src="path/to/Js_files/NC_buildform.js">

<!-- include the Customer Information script-->
<script language=JavaScript src="path/to/Js_files/NC_buildcustomer.js">

<!-- include the Cart View script-->
<script language=JavaScript src="path/to/Js_files/NC_viewcart.js">

<!-- include the Credit Card Entry script-->
<script language=JavaScript src="path/to/Js_files/NC_buildcredit.js">
</BODY>
</HTML>
```

The *NC\_buildform* script, as mentioned before, handles the creation of all the hidden form fields that will be used to send all the order information - ordered items, options, quantities, etc, customer shipping/billing info, and of course the credit card data, to your form handling script.

The scripts *NC\_buildcustomer* and *NC\_viewcart* are simply used to display all the final information and cart totals to the customer for verification.

*NC\_buildcredit* actually handles creating the Card Type radio button set as defined by *cardOption* variable, as well as the input fields for Name on Card, Card Number, and Expiration Date. It also renders the button for submitting the order to your script should the card info be entered correctly.

## Security, AGAIN

We can't stress enough the sensitivity that people have over giving their credit information over the web, even when the connection is secure. You really should think about that full version...

## Unsecure Credit Card Gateway Setup

When the customer chooses to use the unsecure credit option, they are routed to the credit card order page (defined by the *COgateway* variable in the settings file) so they can enter their credit card information and finalize/submit their order. For credit gateway orders, it is assumed that you have a merchant account that either has a web portal for card validation and transaction capture,

or a script based component you can run on your account for card validation. If you do not have a merchant account, then you will not need to use this form of order submission.

### Configuration Notes

In the *NC\_settings* file, there are several variables geared specifically towards gateway posting. The following info should help you get your form order working quickly.

#### *unsecureGatewayAction*

Set the *unsecureGatewayAction* variable to a valid URL of your card processing component or portal. For example, merchants that use Verisign's PayFlow Link portal would set their *unsecureGatewayAction* variable like so:

```
var unsecureGatewayAction =
'https://payflowlink.verisign.com/payflowlink.cfm';
```

#### *extraFormTags*

This variable is an array that can hold any fields that may be required by your credit processing portal to operate. For instance, Verisign's PayFlow Link requires the fields *LOGIN*, *PARTNER*, *TYPE*, and other fields in order to function properly. Since the *NC\_buildform.js* script is built to create generalized forms, it's better to leave specialized, non cart related form fields out of the script and add them when the time comes.

When the *buildform.js* file is executed, it creates the form tag with the action equal to the value of *unsecurePostAction*, then immediately writes all the tags in the *extraFormTags* array. After that, it creates all the form tags relevant to the NebuCart system. An example of a PayFlow Link specific *extraFormTags* array is below:

```
var extraFormTags = new Array(
'<input type="hidden" name="LOGIN" value="your_login">',
'<input type="hidden" name="PARTNER" value="VeriSign">',
'<input type="hidden" name="TYPE" value="S">',
'<input type="hidden" name="METHOD" value="CC">',
'<input type="hidden" name="EMAILCUSTOMER" value="False">',
'<input type="hidden" name="EMAILMERCHANT" value="False">'
);
```

If your gateway doesn't require any special tags, then you can leave this array empty and no extra tags will get rendered.

### **NOTE (if you read only one thing on this page...)**

You should really, *really*, *really* read your card processor's documentation so that you understand how much data, if any, is captured and retained when processing credit card orders. Most processing systems take a set number of fields with proprietary field names, with little or no support for extra fields. If your card processor doesn't capture all the fields passed to it, then you will most likely lose critical order information.

It's a good idea to perform some pre-processing data capture by posting the data to a script that can write the order to a file or database on your account, then repost the data to your gateway. This method is being used with most of the sites that are currently using NebuCart, so believe us when we say it works.

We guarantee that when properly configured, NebuCart will post all the form information to your gateway processor. Once it leaves your site, however, it is up to you or your card processor to maintain that data. We DO NOT provide technical support for your gateway (unless you've \$ contracted \$ us to do so), so if something is not performing as it should, check your documentation first!

## Gateway Order Page

Once you've configured your settings for posting to your particular gateway, the gateway order page will handle passing the order information to your card processor. Below is the basic code for a gateway order page:

```
<HTML>
<HEAD>
<!-- include the NebuCart Settings -->
<script language=JavaScript src="path/to/js_files/NC_settings.js">

<!-- include the NebuCart Engine -->
<script language=JavaScript src="path/to/Js_files/NebuCart.js">

<!-- include the NebuCart Formatting Functions-->
<script language=JavaScript src="path/to/Js_files/NC_formatting.js">
</HEAD>

<BODY>

<!-- include the Form Builder script-->
<script language=JavaScript src="path/to/Js_files/NC_buildform.js">

<!-- include the Customer Information script-->
<script language=JavaScript src="path/to/Js_files/NC_buildcustomer.js">

<!-- include the Cart View script-->
<script language=JavaScript src="path/to/Js_files/NC_viewcart.js">

<!-- include the Credit Card Entry script-->
<script language=JavaScript src="path/to/Js_files/NC_buildcredit.js">
</BODY>
</HTML>
```

The *NC\_buildform* script, as mentioned before, handles the creation of all the hidden form fields that will be used to send all the order information - ordered items, options, quantities, etc, customer shipping/billing info, and of course the credit card data, to your credit card processor, or pre-validation data capture script.

The scripts *NC\_buildcustomer* and *NC\_viewcart* are simply used to display all the final information and cart totals to the customer for verification.

*NC\_buildcredit* actually handles creating the Card Type radio button set as defined by *cardOption* variable, as well as the input fields for Name on Card, Card Number, and Expiration Date. It also renders the button for submitting the order to your gateway should the card info be entered correctly.

### **Back-Ends**

Due to the widely varied field naming conventions used by all the gateway processors, the default *NC\_buildform* and *NC\_buildcredit* scripts may not send the correct fields to your particular gateway. These two are the most highly customized scripts in the NebuCart system. We already have customized scripts for Verisign, Authorize.net, and WorldPay Jr., so check our Back-Ends page to see what is available. If we don't have it, we can write it!

### **Security, Yet AGAIN**

We can't stress enough the sensitivity that people have over giving their credit information over the web, even when the connection is secure. You really should think about that full version...