iTransact Split Form Documentation

Payroc, LLC

iTransact Split Form Documentation

Payroc, LLC

Publication date Version 5.0 - August 27, 2019

Table of Contents

List of Figures

Chapter 1. Split Form Integration Guide

1. Split Form

1.1. Introduction

If you do not have your own secure server, we suggest you utilize our Split Form. The Split Form allows you to meet all of the necessary CISP/PCI Security requirements for accepting credit cards through a gateway, without having to meet those requirements on your own server. The split form's main function is to collect payment information on your behalf and return the transaction results to your server. You can configure it to only collect card information, but you can also display your set of order items or collect the customer's billing and shipping information. You can use the optional style settings or form tag commands to include images and colors to make the secure page look and feel more like your own site.

The URL for the split form can be accessed at the following locations depending on your business needs.

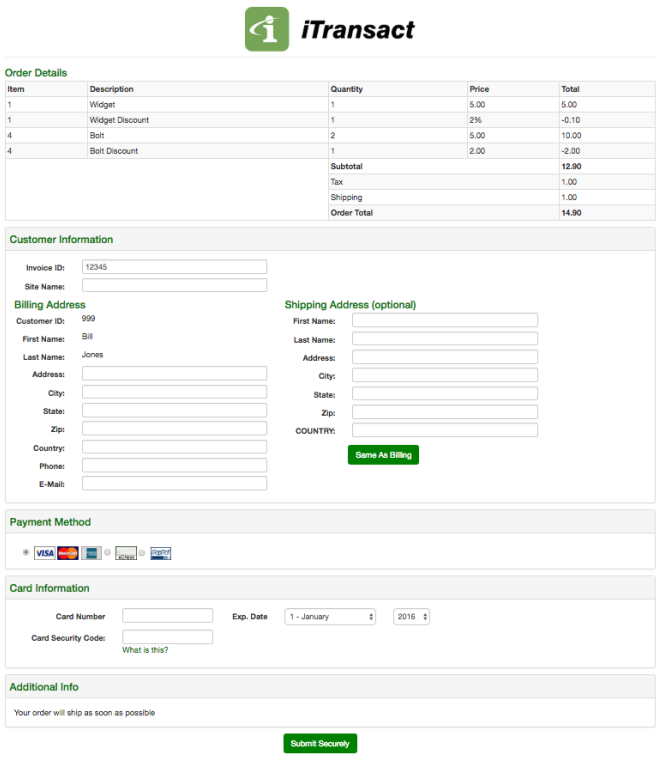
https://secure.itransact.com/customers/split

https://secure.paymentclearing.com/customers/split (For customers who would prefer to not use the iTransact brand)

https://gateway.itransact.com/customers/split (This domain is using an ESSL certificate)

Once you create a form integrated with Split, a submission of your form will generate the secure payment page on our server. Once the customer submits the form, the transaction will be processed and we will either submit the transaction results back to your system or display a default status page depending on how you configure the tool.

Figure 1.1. Split Form - Payment Page Example



1.2. Examples

The following example forms demonstrate the capabilities of this tool from simple to complex.

1.2.1. BuyNow style

This example is a simple Buy Now style button where no information is collected on your server. This form will generate a split form that collects the billing and shipping information as well as the payment info.

```
<form method="post" id="paymentForm" action="https://secure.itransact.com/customers/split">
    <input type="hidden" name="vendor_id" value="__YOUR_UID__" />
    <input type="hidden" name="ret_addr" value="http://test.com"/>
    <input type="hidden" name="ret_mode" value="post" />
    <input type='hidden' name='showaddr' value='1'>
    <input type='hidden' name='altaddr' value='1'>
    <input type='hidden' name='showcvv' value='1'>
    <input type="hidden" name="1-desc" value="An awesome widget">
    <input type="hidden" name="1-cost" value="1.99">
    <input type="hidden" name="1-qty" value="1">
    <input type="submit">Buy Now!</input>
  </form>
```

1.2.2. iFrame AVSOnly with no headers

This example form can be used in an iFrame to collect and validate payment information without actually holding funds on the customer's card. The resulting transaction XID can be used to process future payments.

```
<form method="post" id="paymentForm" action="https://secure.itransact.com/customers/split">
    <input type="hidden" name="vendor_id" value="__YOUR_UID__" />
    <input type="hidden" name="ret_addr" value="http://test.com"/>
    <input type="hidden" name="ret_mode" value="post" />
    <input type='hidden' name='showaddr' value='1'>
    <input type='hidden' name='altaddr' value='1'>
    <input type='hidden' name='show_header' value='0'>
    <input type='hidden' name='showcvv' value='1'>
    <input type="hidden" name="1-desc" value="An awesome widget">
    <input type="hidden" name="1-cost" value="0.00">
    <input type="hidden" name="1-qty" value="1">
    <input type="submit">Proceed to Secure Server</input>
  </form>
```

1.2.3. Customer data collected on merchant server, card information collected in split form with items and customer information displayed.

This example form could be used within a shopping cart that has already collected the customer's billing information. The split form will display the order items and the customer information but won't allow the customer to make further changes to their info. The number of hidden input fields does grow somewhat large in this scenario but this approach allows you to control every field individually.

```
<form method="post" id="paymentForm" action="https://secure.itransact.com/customers/split">
    <input type="hidden" name="vendor_id" value="__YOUR_UID__" />
    <input type="hidden" name="ret_addr" value="http://test.com"/>
    <input type="hidden" name="ret_mode" value="post" />
    <input type='hidden' name='showaddr' value='1'>
    <input type='hidden' name='first_name' value='John'>
    <input type='hidden' name='hide_input' value='first_name'>
    <input type='hidden' name='show_value' value='first_name'>
    <input type='hidden' name='last_name' value='Smith'>
    <input type='hidden' name='hide_input' value='last_name'>
    <input type='hidden' name='show_value' value='last_name'>
    <input type='hidden' name='address' value='123 Some St'>
    <input type='hidden' name='hide_input' value='address'>
    <input type='hidden' name='show_value' value='address'>
    <input type='hidden' name='city' value='Salt Lake City'>
    <input type='hidden' name='hide_input' value='city'>
    <input type='hidden' name='show_value' value='city'>
```

```
    <input type='hidden' name='state' value='Utah'>
    <input type='hidden' name='hide_input' value='state'>
    <input type='hidden' name='show_value' value='state'>
    <input type='hidden' name='zip' value='84103'>
    <input type='hidden' name='hide_input' value='zip'>
    <input type='hidden' name='show_value' value='zip'>
    <input type='hidden' name='country' value='USA'>
    <input type='hidden' name='hide_input' value='country'>
    <input type='hidden' name='show_value' value='country'>
    <input type='hidden' name='phone' value='801-555-1212'>
    <input type='hidden' name='hide_input' value='phone'>
    <input type='hidden' name='show_value' value='phone'>
    <input type='hidden' name='email' value='test@test.com'>
    <input type='hidden' name='hide_input' value='email'>
    <input type='hidden' name='show_value' value='email'>
    <input type='hidden' name='showcvv' value='1'>
    <input type="hidden" name="1-desc" value="An awesome widget">
    <input type="hidden" name="1-cost" value="1.00">
    <input type="hidden" name="1-qty" value="1">
    <input type="submit">Proceed to Secure Server</input>
  </form>
```

1.2.4. Customer data collected in form with additional merchant specified fields.

This form is an example of how to specify your own fields and have the fields saved and returned to your server during postback.

```
<form method="post" id="paymentForm" action="https://secure.itransact.com/customers/split">
    <input type="hidden" name='vendor_id' value='__YOUR_UID__' />
    <input type="hidden" name='ret_addr' value='http://test.com'/>
    <input type="hidden" name="ret_mode" value="post" />
    <input type='hidden' name='showaddr' value='1'>
    <input type='hidden' name='altaddr' value='1'>
    <input type="hidden" name='show_input' value='location'>
    <input type="hidden" name='passback' value='location'>
    <input type="hidden" name='save' value='location'>
    <input type="hidden" name='location' value='Downtown'>
    <input type="hidden" name='location_label' value='Enter Location'>
    <input type='hidden' name='showcvv' value='1'>
    <input type="hidden" name='1-desc' value='An awesome widget'>
    <input type="hidden" name='1-cost' value='0.00'>
    <input type="hidden" name='1-qty' value='1'>
    <input type="submit">Proceed to Secure Server</input>
  </form>
```

1.3. Required Fields

Please pass these fields for all Split Form transactions.

- **vendor_id** - This value is your Order Form UID which is found in your merchant settings in the control panel.
- **ret_addr** - This value is the URL of the page or dynamic script that a customer and/or data is passed to after an order is submitted. Please remember that any images or information on this page must have the absolute addresses. In most cases, this page will be emulated in our secure server environment.
- **Item Cost Methods** - There are two methods for setting the amount to be billed to the customer. One of these must be used with each split form transaction. If neither of these methods are included, the transaction will not be allowed.
  - **Passed Order Items** - Used when you have preset the order items for the transaction. See the order items section below.
  - **Customer Entered Amount** - Used when you allow your customers to enter the amount that they are paying or donating.

- **total_input** - This value should be "1" if you want to allow the customer to enter their own payment amount.
- **total_input_desc** - This value is the description of the order item. The localized version of 'Transaction Total' will be used if not passed.
- **total_input_cost** - This is an option field which will set the default value for the total input field. You could use it to set the current balance or suggested donation amount for example.

1.4. Customer Fields

The following fields can either be passed through with the request to split or they can be collected by the form by setting the showaddr and altaddr fields. You can also control whether these fields are rendered as input fields, display fields or hidden. This formatting control can be done at the field level and is described in the Field Display Control section of this document.

- **first_name** - This value is the customer's first name - 50 maximum characters.
- **last_name** - This value is the customer's last name - 50 maximum characters.
- **phone** - This value is the customer's phone number - 25 maximum characters.
- **email** - This value is the customer's email address - 255 maximum characters.
- **cust_id** - This value is a merchant defined customer id - 40 maximum characters.
- **address** - This value is the cardholder's billing street address - 100 maximum characters.
- **city** - This value is the cardholder's city of their billing address - 25 maximum characters.
- **state** - This value is the cardholder's state of their billing address - If the transaction is foreign and there is no corresponding state, please enter a comma into this field - 25 maximum characters.
- **zip** - This value is either the five digit or nine digit billing postal code - If the transaction is foreign and there is no corresponding postal code, please enter a comma into this field - 12 maximum characters.
- **country** - This value is the cardholder's country of their billing address - 45 maximum characters.
- **saddr** - This value is the cardholder's shipping street address - 100 maximum characters.
- **scity** - This value is the cardholder's city of their shipping address - 25 maximum characters.
- **sstate** - This value is the cardholder's state of their shipping address - If the transaction is foreign and there is no corresponding state, please enter a comma into this field - 25 maximum characters.
- **szip** - This value is either the five digit or nine digit shipping postal code - If the transaction is foreign and there is no corresponding postal code, please enter a comma into this field - 12 maximum characters.
- **sctry** - This value is the cardholder's country of their shipping address - 45 maximum characters.

1.5. Order Items

- **Reserved Item Descriptions**

  There are three items that will cause special behavior in the form. If you pass through a 'desc' value of 'tax', 'shipping' or 'surcharge' a Subtotal and Order Total field will be shown in the item table with properly calculated amounts. Each of these reserved fields will be displayed underneath the Subtotal field in the item table.

- **Required Item Attributes**

  Every passed line item must contain the following attributes

- ***_desc** - This value is the description of the order item. The * indicates the item number; `1_desc`, `2_desc`, `3_desc`, etc. A field separator of underscore [`1_desc`] or dash [`1-desc`] may be used. In addition, if your system doesn't allow field names beginning with a number, you may use `item_1_desc`, `item_2_desc`, etc. Only one naming convention per order form may be used.

- ***_cost** - This value is the numeric amount of the price of the item including dollars, decimal, and cents [XXX.xx]. The * indicates the item number; *1_cost*, *2_cost*, *3_cost*, etc. A field separator of underscore [`1_cost`] or dash [`1-cost`] may be used. In addition, if your system doesn't allow field names beginning with a number, you may use `item_1_cost`, `item_2_cost`, etc. Only one naming convention per order form may be used.

- ***_qty** - This value is the number of the item desired. * indicates the item number; `1_qty`, `2_qty`, `3_qty`, etc. A field separator of underscore [`1_qty`] or dash [`1-qty`] may be used. In addition, if your system doesn't allow field names beginning with a number, you may use `item_1_qty`, `item_2_qty`, etc. Only one naming convention per order form may be used.

- ***_VARIABLE** - This value is an attribute of the order item. The * indicates the item number; `1_VARIABLE`, `2_VARIABLE`, `3_VARIABLE`, etc. "`VARIABLE`" can be anything. A field separator of underscore [`1_VARIABLE`] or dash [`1-VARIABLE`] may be used. In addition, if your system doesn't allow field names beginning with a number, you may use `item_1_VARIABLE`, `item_2_VARIABLE`, etc. Only one naming convention per order form may be used.

```
<input name="1_size" value="Large">
<input name="1_color" value="White">
```

- **Discount Commands**

  - **Negative Value Item** - The format is the same as a normal item. A `*_qty`, `*_desc`, and `*_cost` must be passed for the negative item. The command is only to be used in conjunction with at least one non-negative value item with a positive total. This feature will subtract the amount of negative value item from the transaction total. The gateway will not allow a transaction to process with a negative total. A merchant should use the Post A Credit function if money needs to be credited to a customer's account. * indicates the item number; `1_desc`, `2_desc`, `3_desc`, etc. A field separator of underscore [`1_desc`] or dash [`1-desc`] may be used. In addition, if your system doesn't allow field names beginning with a number, you may use `item_1_desc`, `item_2_desc`, etc. Only one naming convention per order form may be used.

```
<input type = "hidden" name="1_cost" value="100.00">
<input type = "hidden" name="1_desc" value="Item description">
<input type = "hidden" name="1_qty" value="1">
<input type = "hidden" name="2_cost" value="-25.00">
<input type = "hidden" name="2_desc" value="Discount description">
<input type = "hidden" name="2_qty" value="1">
```

    The total to be charged to a customer's account in the above example is $75.00.

  - ***_discount** - The value of this field is the amount in dollars and cents which is amount of the discount given to the customer. This amount will be subtracted from the `*_cost` of the item it is being passed with. * indicates the item number; `1_discount`, `2_discount`, `3_discount`, etc. A field separator of underscore [`1_discount`] or dash [`1-discount`] may be used. In addition, if your system doesn't allow field names beginning with a number, you may use `item_1_discount`, `item_2_discount`, etc. Only one naming convention per order form may be used.

```
<input type = "hidden" name="1_cost" value="100.00">
<input type = "hidden" name="1_desc" value="Item Desc">
<input type = "hidden" name="1_qty" value="1">
<input type = "hidden" name="1_discount" value="10.00">
```

    The total to be charged to a customer's account in the above example is $90.00.

  - ***_discount_percent** - The value of this field is the percentage of the item amount being given to the customer. This percentage will be subtracted from the `*_cost` of the item it is being passed with. * indicates the item number; `1_discount_percent`, `2_discount_percent`, `3_discount_percent`, etc. A field sepa-

rator of underscore [1_discount_percent] or dash [1-discount_percent] may be used. In addition, if your system doesn't allow field names beginning with a number, you may use item_1_discount_percent, item_2_discount_percent, etc. Only one naming convention per order form may be used.

```
<input type = "hidden" name="1_cost" value="200.00">
<input type = "hidden" name="1_desc" value="Item Desc">
<input type = "hidden" name="1_qty" value="1">
<input type = "hidden" name="1_discount_percent" value="10.00">
```

The total to be charged to a customer's account in the above example is $180.00.

1.6. Optional Fields

These fields can be used to enhance your split forms, support additional data, and allow you to receive data back to your server after a transaction.

- **tpl_code** - The split form can be displayed in both French and Spanish. To change the displayed language use the tpl_code field and set the value to either 'fr' or 'es' for French or Spanish versions

  ```
  <input type="hidden" name="tpl_code" value="fr"/>
  ```

- **preauth** - Pass this command if you only want to pre-authorize a credit card to be charged at a later time.

  ```
  <input type="hidden" name="preauth"/>
  ```

- **ret_mode** - The value for this command is either post or redirect.

  ```
  <input type="hidden" name="ret_mode" value="redirect">
  ```

  or

  ```
  <input type="hidden" name="ret_mode" value="post">
  ```

  Please review all of the features of the Return Mode function [10].

- **lookup** - The value(s) of this command is/are the desired predetermined values of some of the required/non-required fields.

  ```
  <input type="hidden" name="lookup" value="first_name">
  ```

  Please review all of the details of the Lookup function [12].

- **passback** - The value(s) of this command is/are the desired values of most of the non-*Lookup* fields.

  ```
  <input type="hidden" name="ordernum" value="order#999">
  <input type="hidden" name="passback" value="ordernum">
  ```

  Please review all of the details of the Passback function [11].

- **post_back_on_error** - This command is used to request the decline and error responses be sent to the ret_addr on failed transactions. The value for this field should be "1".

  ```
  <input type="hidden" name="post_back_on_error" value="1">
  ```

  Please review all of the details of the error postback system [11].

- **save** - The value(s) for this command is/are the name of any of the fields (see the section detailing this) that you would like tracked through our meta-data system.

  ```
  <input type="hidden" name="save" value="ordernum">
  ```

- **email_text...email_text10** - This/these command(s) can be used to add up to ten additional messages (up to 255 characters each) on the confirmation email generated by the gateway. This information will appear on both the customer and merchant emails.

  ```
  <input type="hidden" name="email_text" value="Thanks for shopping with us today.">
  <input type="hidden" name="email_text2" value="Please shop with us again!">
  ```

- **cust_id** - The value for this custom field can be a unique alpha-numeric identifier up to 40 characters assigned on the merchant's end for tracking purposes. It is a searchable field.

```
<input type="hidden" name="cust_id" value="ABC123">
```

- **Formatting Commands** - These optional fields (including the form tags) will allow you to change the way the secure payment page will display. Many of these tags have corresponding settings that can be controlled in the Account Settings. Use of any of the formatting commands will circumvent the corresponding settings on your gateway.

  - **showaddr** - The value of this field must be “1” if you wish for the address information that was entered on the first half of Split Form to display.

```
<input type="hidden" name="showaddr" value="1">
```

  - **show_header** - If this field is passed through with the value 0 then the header of the form which would contain the merchant name or header image will be hidden.

```
<input type="hidden" name="show_header" value="0">
```

  - **mername** - This field allows you to change the company name that is displayed on the form by default.

```
<input type="hidden" name="mername" value="Some Alternate Name">
```

  - **altaddr** - The value for this is "1" if you would like the customer to enter a shipping address if different from the billing address.

```
<input type="hidden" name="altaddr" value="1">
```

  - **show_items** - This displays order items on secure page instead of just the transaction total if you pass a value of “1”.

```
<input type="hidden" name="show_items" value="1">
```

  - **showcvv** - The value of this field must be “1” if you would like to allow the customer to enter the CVV number printed on their credit card for verification purposes.

```
<input type="hidden" name="showcvv" value="1">
```

  - **nonum** - Use a value of “1” to remove the check number field from the payment page for a check transaction.

```
<input type="hidden" name="nonum" value="1">
```

  - **mertext** - This allows you to add a line of text at the bottom of the secure payment page.

```
<input type="hidden" name="mertext" value="Thank you. Please come again!">
```

  - **Card Image Commands** - Do not use for any card types you are not authorized to receive.

    - **visaimage** - If used, this value should be "1" if you are authorized to accept Visa if you want it to display, and "0" if you are not authorized for Visa acceptance or you do not want it to display. This command circumvents the setting in the gateway.

```
<input type="hidden" name="visaimage" value="0">
```

    - **mcimage** - This value should be "1" if you are authorized to accept MasterCard if you want it to display, and "0" if you are not authorized for MasterCard acceptance or you do not want it to display. This command circumvents the setting in the gateway.

```
<input type="hidden" name="mcimage" value="0">
```

    - **ameximage** - This value should be "1" if you are authorized to accept AMEX if you want it to display, and "0" if you are not authorized for AMEX acceptance or you do not want it to display. This command circumvents the setting in the gateway.

```
<input type="hidden" name="ameximage" value="0">
```

- **discimage** - This value should be "1" if you are authorized to accept Discover if you want it to display, and "0" if you are not authorized for Discover acceptance or you do not want it to display. This command circumvents the setting in the gateway.

```
<input type="hidden" name="discimage" value="0">
```

- **dinerimage** - This value should be "1" if you are authorized to accept Diners Card if you want it to display, and "0" if you are not authorized for Diners Card acceptance or you do not want it to display. This command circumvents the setting in the gateway.

```
<input type="hidden" name="dinerimage" value="0">
```

- **Field Display Control**

  - **show_input** - This field can be used to control whether a customer ID field is shown in the Billing Address inputs. You can also use this field to show custom fields which you would like the customer to populate. These custom fields need to be a passback or save field.

    Here is the field you need to have to show a Customer ID field in the Billing Address inputs.

```
<input type="hidden" name="show_input" value="cust_id">
```

    Here is an example of a field set that you would use to display a custom field. By passing the 'location' field through, the displayed input field will be pre-populated. If no location field is provided then the field will be empty. This example also includes a labeling input which is described below. If you do not provide a labeling input field then a label will be auto-generated using the field name: 'location' -> 'Location', 'your_location' -> 'Your Location', 'yourLocation' => 'Your Location'

```
<input type="hidden" name="show_input" value="location">
<input type="hidden" name="passback" value="location">
<input type="hidden" name="location" value="Downtown">
<Input type="hidden" name="location_label" value="Enter Location">
```

  - **hide_input** - This field can be used to hide any of the billing or shipping input fields. This would generally be done if you wanted to do something like pre-populate the country field.

```
<input type="hidden" name="hide_input" value="country">
<input type="hidden" name="hide_input" value="sctry">
<input type="hidden" name="country" value="USA">
<input type="hidden" name="sctry" value="USA">
```

  - **show_value** - This field can be used to display a field value in the case that the input field has been hidden. Expanding on the above example, the below example will show the country field value in the Billing Address area of the form.

```
<input type="hidden" name="show_value" value="country">
<input type="hidden" name="show_value" value="sctry">
```

  - **X_label** - This field can be used to re-label custom fields as well as any of the billing or shipping address fields.

```
<input type="hidden" name="label_sctry" value="S. Ctry.">
```

- **Payment Type Commands** - Do not use for any card types you are not authorized to receive. A payment type will only work correctly if you have been activated with the correct processor. If you would like to accept additional payment types, please contact our *support team.*

  - **acceptcards** - This value should be "1" if you want card acceptance fields to display.

```
<input type="hidden" name="acceptcards" value="0">
```

  - **acceptchecks** - This value should be "1" if you want check acceptance fields to display.

```
<input type="hidden" name="acceptchecks" value="0">
```

- **accepteft** - This value should be "1" if you want EFT acceptance fields to display.

```
<input type="hidden" name="accepteft" value="0">
```

- **disable_cards** - This tag allows you to remove the card acceptance fields from the final check out page if you pass the value of “1”. This command circumvents the setting in the gateway.

```
<input type="hidden" name="disable_cards" value="1">
```

- **disable_checks** - This tag allows you to remove the check acceptance fields from the final check out page if you pass the value of “1”.This command circumvents the setting in the gateway.

```
<input type="hidden" name="disable_checks" value="1">
```

- **Image Commands** - You may submit images to us to host on our server.

  - **background** - This command allows your submission to call an image that you have uploaded to our server to display as the background of the final checkout page. The background image should be named like `12345bg.gif` (or .jpg), where “`12345`” is your Gateway ID. This command circumvents the setting in the gateway.

```
<INPUT type="hidden" name="background" value=
 "https://secure.itransact.com/images/background/12345bg.gif">
```

  - **Header Graphic Commands** - These lines allow you to call an order form that displays your company's logo across the top of the final check out page. This same image will be used on Customer Billing Update page for recurring transactions if you use that feature. These commands circumvent the setting in the gateway.

    - **mastimage** - The value for this is “1”. This must always be paired with the mast tag.

    - **mast** - This command allows your script to call an image that you have uploaded to our server to display as the header graphic of the final checkout page. Maximum mast width is 800 pixels. This must always be paired with the mastimage tag. The header image should be named `12345mst.gif` (or .jpg), where “`12345`” is your Gateway ID. This command circumvents the setting in the gateway.

```
<INPUT type="hidden" name="mastimage" value="1">
<INPUT type="hidden" name="mast" value=
 "https://secure.itransact.com/images/mast/12345mst.gif">
```

- **Color Commands** - Change the coloring used in the template of your secure payment page. These commands circumvents the settings in the gateway.

  - **bgcolor** - This tag allows you to designate the background color of the final checkout page.

```
<INPUT type="hidden" name="bgcolor" value="green">
```

  - **fontcolor** - This tag allows you to designate the color of the text on the final checkout page.

```
<INPUT type="hidden" name="fontcolor" value="blue">
```

  - **alink** - This tag allows you to designate the color of an active link.

```
<INPUT type="hidden" name="alink" value="red">
```

  - **link** - This tag allows you to designate the color of a link.

```
<INPUT type="hidden" name="link" value="blue">
```

  - **vlink** - This tag allows you to designate the color of a visited link.

```
<INPUT type="hidden" name="link" value="blue">
```

- **Recurring Transaction Commands**

- **recur_recipe** - The value for this is the Recurring Recipe that you have built in your Control Panel. This must always be paired with the recur_reps tag.

- **recur_reps** - This numeric value is the number of times you would like a transaction to follow the recurring recipe which programs the gateway to bill a card on an ongoing basis at the merchant's request. This must always be paired with the recur_recipe tag.

- **recur_total** - This can be used if you are using the recurring billing feature to bill a card for a different amount than the initiating transaction. This feature can only be used in conjunction with recipes set for split amount recurring billing. If you pass this, you must pass recur_desc.

- **recur_desc** - This tag allows a merchant to change the billing description which appears on the confirmation emails for recurring transactions. This feature can only be used in conjunction with recipes set for split amount recurring billing. If you pass this, you must pass recur_total.

```
<input type="hidden" name="recur_recipe" value="monthly13">
<input type="hidden" name="recur_reps" value="6">
<input type="hidden" name="recur_total" value="50.00">
<input type="hidden" name="recur_desc" value="Enter a description here.">
```

2. Getting Data from Gateway

The gateway has methods for returning data to you after an HTML transaction. Those tools are the ret_mode [10], lookup [12], and passback [11] functions. You may use any or all of the methods for each transaction. The requested data will come back in a name/value pair string. The strings are always signed with a PGP Signature when either the lookup or passback function is used, so you can verify the response came from our server.

2.1. What is ret_mode?

The Return Mode function (ret_mode) enables you to specify how your customers are returned to your return address (ret_addr) after a successful transaction, as well as how transaction information is relayed to your script.

By default, customers are shown a “Thank You” page on the secure server after a successful transaction. When the customer clicks the `Continue` button, all name/value pairs are returned to your script using GET. However, the Return Mode function allows the customer to bypass the “Thank You” page and be directed to your ret_addr. This enables you to use the gateway services more seamlessly.

This function also has a feature enabling you to receive error messages and failures (declines) returned to a specified script on your server.

There are two values that can be used with the Return Mode Function. They are post and redirect. Each has a specific function. You can only use one method at a time.

2.1.1. Return Mode of Redirect

The redirect value of ret_mode value can be used if your ret_addr is a static HTML document. After a successful transaction, your customer is automatically redirected to your ret_addr, bypassing the “Thank You” page. Since this is a simple redirect, no passback or lookup values can be returned.

```
<input type="hidden" name="ret_mode" value="redirect">
```

2.1.2. Return Mode of POST

This method can only be used if the ret_addr page is a dynamic script/page. After a successful transaction, the information you have requested from the processing server will be posted to your ret_addr. All lookup and passback name/value pairs will be returned via the POST.

```
<input type="hidden" name="ret_mode" value="post">
```

2.1.3. POST of Error Messages

If you would like decline and error messages also sent to your ret_addr, you may use this feature. This feature is very powerful, giving you complete control of the entire transaction process, because you can take the error message and display it in your own error screen.

```
<input type="hidden" name="post_back_on_error" value="1">
```

Responses will appear as follows:

- If a transaction is declined, you will receive a name/value pair of “err=Text message of the error” along with the name/value pairs for each passback and lookup variable requested.
- If an internal error is encountered, you will receive a name/value pair of “die=1”.
- If the transaction is successful, neither of these will appear.
- The responses of *err* and *die* will never be sent on the same transaction. A transaction will either be successful, an error, or a die.

2.1.4. Considerations and Restrictions

- When using ret_mode with the redirect option, your ret_addr page should reside on a secure server. If not, your customers will receive a browser security warning.
- When using ret_mode with the post option, your ret_addr URL should be on a secure (HTTPS) server. The page will be displayed securely as an emulated page in the secure server environment.
- When using ret_mode, your ret_addr must be an absolute URL, not a relative URL. (i.e. *http://www.yoursite.com/cgi-bin/return.cgi* is absolute. *.../cgi-bin/return.cgi* is relative.) In addition, all links located on your ret_addr page must also be absolute URLs.
- The ret_mode function uses a standard HTTP 1.1 POST.
- If your ASP or Cold Fusion application cannot "see" the incoming name/value pairs, you will need to consult your documentation/system admin.
- For security reasons, you must use a standard port for post-back data. (port 80 for HTTP or port 443 for HTTPS) In other words, a ret_addr of *http://www.yoursite.com:9876* will not work correctly.
- The Return Mode function is much more dynamic than the standard GET method used to return customers to your site. Please be aware that if your ret_addr is not available, unreadable, incomplete, times out, etc., your customer's account will have already been billed and they will receive an error message. Only use the Return Mode function if you are very familiar with CGI scripting. Always test your applications before making them available to your users.

2.2. How do I use passback?

This feature gives a merchant the ability to request and receive information that was submitted as a part of the order form post submission. This feature can be used to access any of the data that can not be retrieved via the lookup function. Fields reserved for the lookup function can not be requested using the passback function. The passback and lookup functions can both be used on the same form. The passback function enables you to include input variables in your order form that are passed back to your return address after the transaction is completed. The value for the ret_addr must be a dynamic page or script that can accept, parse, and interpret name/value pairs. In this example, the ret_addr field in the order form was set to *http://www.yoursite.com/cgi-bin/return.cgi*. This example uses two variables. They are named "fieldname1" and "ordernum". This example shows variables returned as part of the query string.

```
<INPUT type="hidden" name="passback" value="fieldname1">
<INPUT type="hidden" name="fieldname1" value="Ord">
<INPUT type="hidden" name="passback" value="ordernum">
```

```
<INPUT type="hidden" name="ordernum" value="99">
```

After a successful order has been completed, the customer is returned to the URL of the ret_addr. Any requested data will be sent as part of the query_string or as name/value pairs from an HTML POST. Orders are always signed with a PGP Signature when either the lookup or passback function is used.

Example

```
http://www.site.com/cgi-bin/return.cgi?fieldname1=Ord&ordernum=99&signature=PGPSIGNATUREHERE
```

Considerations

- Field names used for the lookup function are reserved and may not be used as passback field names. However, the passback function may be used to request the contents of each description, cost, and quantity.
- Your ret_addr (return address) must be a script (such as Perl, PHP, ASP, CFM) that is capable of accepting and parsing variables posted from an HTML form.
- Each passback field must contain a value. If the field contains no value or an invalid value, you will receive a “nonexistent passback parameter” error.

2.3. How do I use lookup?

Don't be confused by the name of this function. This feature provides the same functionality of the passback function for 30 separate name/value pairs. The passback and lookup functions can both be used on the same form. The lookup function allows you to request specific customer data from the processing server at the time a transaction is processed. All data requested using the lookup function is sent directly to the return address (ret_addr) specified in the order form. The value for the ret_addr must be a dynamic page or script that can accept, parse, and interpret name/value pairs. A merchant can request as many of the name/value pairs as needed.

The following fields can be received in the POST via the lookup function:

- **first_name** - Returns the value entered as the first_name.
- **last_name** - Returns the value entered as the last_name.
- **address** - Returns the value entered as the address.
- **city** - Returns the value entered as the city.
- **state** - Returns the value entered as the state.
- **zip** - Returns the value entered as the postal code.
- **country** - Returns the value entered as the country.
- **phone** - Returns the value entered as the phone.
- **email** - Returns the value entered as the email.
- **sfname** - Returns the value entered as the shipping first name.
- **slname** - Returns the value entered as the shipping last name.
- **saddr** - Returns the value entered as the shipping address.
- **scity** - Returns the value entered as the shipping city.
- **sstate** - Returns the value entered as the shipping state.
- **szip** - Returns the value entered as the shipping postal code.
- **sctry** - Returns the value entered as the shipping country.

- **authcode** - Returns the credit card authorization code.
- **cc_last_four** - Returns the last four digits of the card number.
- **ck_last_four** - Returns the last four digits of the checking account number.
- **cc_name** - Returns the name of the card type used. "Visa", for example.
- **total** - Returns the transaction total.
- **test_mode** - Returns "1" if your account is in test mode or "0" if it is not in test mode.
- **when** - Time/date stamp in format of "20010509134443" - meaning 05/09/2001 at 13:44:43.
- **xid** - Returns the transaction ID.
- **batch_number** - Returns the batch number for all transactions except EFT, since EFT transactions aren't currently assigned batch numbers.
- **avs_response** - Returns the address verification response.
- **cvv2_response** - Returns the CVV response.
- **confemail** - Returns "1" if your account is set to send email confirmations to your customers or "0" if confirmations are not sent.
- **entry_method** - Returns either swipe or keyed.
- **email_text (email_text2 - email_text9)** - Returns the value(s) entered as the email_text field(s).
- **balance** - Returns the available credit line amount of the gift card/debit/prepaid if passed back from the processor.
- **authorized_amount** - Returns the partial amount that was approved for a retail swiped/keyed transaction.
- **billing_update_token** - If a transaction is initiating a recurring transaction tied to a recipe with Allow Customer Update enabled, this is the value of the token to be used in a link to the secure billing update page. Should be 60-80 characters.

To use the *lookup* function, simply add the the commands to HTML code to your order form with the appropriate field name. In this example, the ret_addr field in the order form is *http://www.yourdomain.com/cgi-bin/return.cgi*. The request would look like this:

```
<INPUT type="hidden" name="lookup" value="first_name">
<INPUT type="hidden" name="lookup" value="authcode">
<INPUT type="hidden" name="lookup" value="total">
<INPUT type="hidden" name="lookup" value="xid">
<INPUT type="hidden" name="lookup" value="phone">
```

Assuming the value for the first_name was "Bob", the response would look like this:

```
http://www.yourdomain.com/cgi-bin/return.cgi?first_name=Bob&authcode=852346
&total=29.95&xid=987654&phone=8015551212&signature=PGPSIGNATUREHERE
```

Considerations

- As with any other dynamic web page, your ret_addr (return address) must be a CGI script or some other application that is capable of parsing the name/value pairs that are passed.
- Requested lookup fields that do not have a value will not be included in the information passed back.
- While in test mode, an authcode lookup will return "000000" and an XID lookup will return "9999999999".