6 Checkout Implementation

There are many reasons why institutions decide to implement a Checkout site. Among the most common are:

- Some institutions prefer to use a “front-end” store that they’ve already built but has no way of accepting payments.
- Some institutions want full creative control over their site but prefer not to deal with the checkout process due to PCI or other banking requirements.
- Many ERPs are designed to easily integrate with third party software, such as Transact Payments, that provide eCommerce solutions for future students and alumni. Banner, Jenzabar, PowerCampus, and PeopleSoft all have developed applications that Transact Payments Checkouts can integrate with.
- Other third party vendors that offer parking, housing, continuing education, or other student services already have applications that can direct students through a Transact Payments Checkout site.

In the first and second bullet points, Checkouts can be customized to your specifications and include the items, references, and other transactional data necessary to report against or extract to your student or financial system, provided no additional interfaces need to be designed. If additional interfaces are required, they can almost always be built, but the specifics would have to be discussed with your Client Manager or Implementation Team.

In the third and fourth bullet points, Transact Payments Implementation Teams are often already aware of the requirements and can set these up with only limited guidance. Several ERPs have “payment adapters,” which are specialized checkout packages that include interfaces and other configurations. An Implementation Team member can assist you with these types of implementation.

If you’re unsure whether a checkout would meet your requirements, Transact’s Client Managers and other staff can assist you.

6.1 Determining Your Needs

If you plan to implement a Checkout site, you will want to think of the needs you are attempting to meet. Answer questions such as:

- What items are being purchased? Do they all have to be separate Item Codes in Transact Payments or can they be combined into a single item code?

  Tip: For simplicity, many Checkouts only have one item. While this is acceptable, keep in mind that a receipt will usually be displayed or emailed to a student and the description of an item on that receipt should be unambiguous.

- Will G/L information need to be passed into the system or can this be set on the item by default?

  Tip: Typically the G/L information is set on the Item Code. However, it can be transmitted to the Transact Payments system from the checkout site by using the **gl** field in the checkout request.

- Will you need to import identifying information on the user for the purpose of verification, refunds, or other purposes?

  Tip: References don’t need to be for admin use only. A reference can be set to display on the receipt so that user knows more details about what they’re paying for. Examples include descriptive references such be a continuing education class name, the term code or description, special instructions provided by the user, meal choices, or even a T-shirt size.

- Will anyone else need to find this information in the Transact Payments system?

  Tip: Additional Store Owners, eMarket Administrators, or Inquiry users may need to be configured to allow staff to access the payment information. Alternatively, automatically emailed reports may be sufficient.

Once you have a general idea of what sort of items you plan on selling through the Checkout site, you can prepare the Transact Payments database.

6.2 Preparing Your Transact Payments Database

When setting up your Checkout site, it's best to think of it in exactly the same terms as a Storefront. All the work described in the other sections of this manual needs to be finished before you can begin testing. The main challenge with a Checkout is that you cannot see the catalog page. The user will skip all of the initial pages and will be taken directly to the page where they will select their payment method. Therefore, it may be difficult to visualize the store's progress, and you may find it useful to keep a spreadsheet of each item and the appropriate references for them.

Tip: You don't need to pass every reference to Transact Payments from your front-end site. Not doing so will make setup much simpler. For example, if you are setting up a Checkout for transcript requests, Transact Payments rarely needs to have the full address of the user as long as you are storing that information within a database on your ERP or third party system. It's usually sufficient just to pass to Transact Payments fields such as the user's name, email address, or other identifying information, making it easy to tie the transaction back to your system.

6.3 Checkout Requests

Checkout requests are the information passed from your front-end store to the Transact Payments Checkout site. Detailed information on how these requests work is documented in the *eMarket Payment Integration Guide*, available through your Client Manager or implementation team.

Here is a cursory overview of how a checkout request works:

An HTML form must be created to send the information to the Transact Payments database. These forms can be very simple or extremely complex, but all requests will have the same basic parameters. An example form looks like this:

```
<HTML>
 <HEAD>
  <title>Our Storefront</title>
 </HEAD>
 <body>
  <form method="get" action="http://commerce.cashnet.com/MyUnivCheckout">
<input type="hidden" name="itemcode1" value="PSYCH-CONF"><br>
<input type="hidden" name="amount1" value="124.00"><br>
```

```
<input type="hidden" name="ref1type1" value="EMAIL"><br>
<input type="hidden" name="ref1val1" value="joesmith@myschool.edu"><br>
<input type="hidden" name="custcode" value="987654321"><br>
<input type="hidden" name="fname" value="Joe"><br>
<input type="hidden" name="lname" value="Smith"><br>
<input type="submit" name="submit" value="Make Payment">
  </form>
 </body>
</HTML>
```

The first parameter to take note of is the **form** parameter.

```
<form method="get" action="http://commerce.cashnet.com/MyUnivCheckout">
```

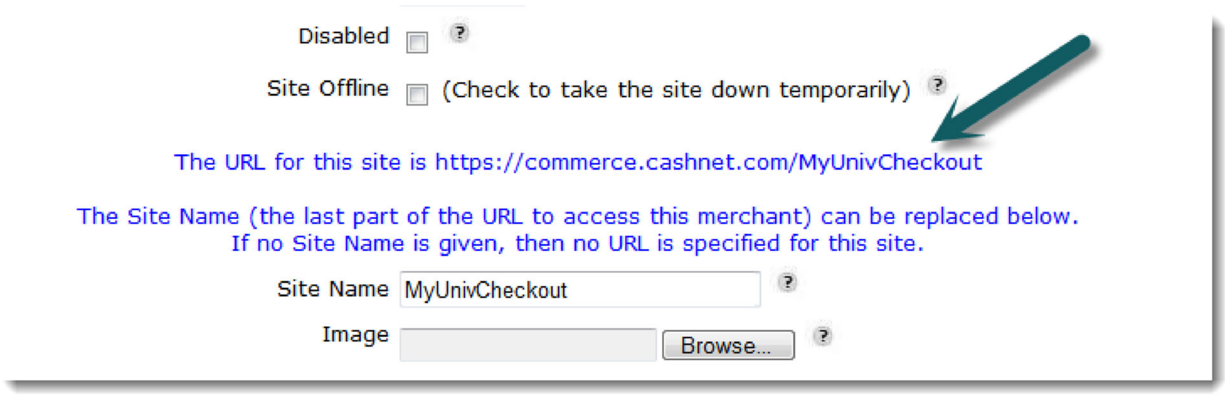
In the above example, there are two important attributes: the method and action. The method describes the way in which the rest of the form will be sent to the Transact Payments database. The system supports the GET and POST methods.

Note: Although the fine details of how these requests work isn't relevant to this manual, it's important to note that the GET method is limited to about 3000 characters. If a particularly large form must be sent to Transact Payments, it's better to send it as a POST.

In addition, GET requests send all parameters as name/value pairs that are potentially visible in the checkout string. Because of this, some IT staff members prefer to send any form data as a POST.

The other attribute in the checkout request is the URL for your checkout site. This URL can be found in Merchant setup about halfway down the page. Remember, this URL is not directly accessible. It must always be sent as part of a checkout request string.

Figure 58. Checkout String



Disabled ☐

Site Offline ☐ (Check to take the site down temporarily)

The URL for this site is https://commerce.cashnet.com/MyUnivCheckout

The Site Name (the last part of the URL to access this merchant) can be replaced below.
If no Site Name is given, then no URL is specified for this site.

Site Name: MyUnivCheckout

Image: Browse...

6.3.1 Item Parameters

The next set of parameters relevant to the form is the input parameters. These define the fields that will be sent to the Transact Payments database and represent the various fields that will populate the checkout transaction. A full list of these parameters can be found in the *eMarket Payment Integration* document, but these are the most common ones.

```
<input type="hidden" name="itemcode1" value="PSYCH-CONF"><br>
<input type="hidden" name="amount1" value="124.00"><br>
```

There are several attributes in this form. The first is the “type.” This determines what kind of field is present in the form, which can be anything from hidden fields, such as the ones above, to text boxes, radio buttons, or checkboxes. For simplicity, these are all hidden. The “name” attribute is the field that will be sent to the Transact Payments database, and the “value” is the contents of that field. For example, sending the name attribute as ‘itemcode1’ and the value attribute as ‘PSYCH-CONF’ means that Transact Payments will interpret it as ‘itemcode1=PSYCH-CONF’. The database will then create a transaction with the first item code matching this value, provided the PSYCH-CONF item code exists in the Transact Payments database.

You’ll notice that many of the other fields have a ‘1’ at the end. This indicates that the Transact Payments database should associate these fields with the first item. In the example above, the database will interpret the PSYCH-CONF item’s price as $124.00 and it will have an EMAIL reference of ‘joesmith@myschool.edu’.

6.3.2 Non-Item Parameters

There are several fields without this numbering, which means those fields apply to the transaction as a whole. These include customer code, first name, and last name. These fields correspond to demographic information stored on the transaction record itself rather than being associated with a particular item.

```
<input type="hidden" name="custcode" value="987654321"><br>
<input type="hidden" name="fname" value="Joe"><br>
<input type="hidden" name="lname" value="Smith"><br>
```

In the example above there are only two non-item parameters, but many others can be defined, such as address, city, state, and zip code. Note, however, that none of these fields is used if a default customer is set on the Merchant.

Note: A valid student ID can be passed in the ‘custcode’ field and will be visible on a student’s ePayment site. New customer records can also be passed to Transact Payments with the ‘custcode’, ‘fname’, ‘lname’, and other fields.

However, if a default customer is being used, these fields will already be “locked” in the Transact Payments database, which will then ignore these non-item parameters. If you want to include this information with a default customer, we suggest that you create a Reference for that purpose (for example, passing a FULLNAME parameter). Just remember to send these fields in the reference format described below.

6.3.3 References

References must be passed to the Transact Payments database in a special manner. Because an item can have many references, there are actually two numbers for each reference. The first is a sequential number for each reference associated with an item. The second is the item code number the reference is associated with. For example, ref1type1 is the *first* reference for the *first* item, whereas ref3type2 would be the *third* reference on the *second* item. This is very important when passing multiple references or items into the system.

References must also be passed as separate fields. The reference type (ref1type1) is the name of the reference as it is set up in the database. In the example below, a reference called EMAIL must be created before it can be sent to the Transact Payments database. However, the database needs to also know what you want to populate the email reference with. This is the purpose of the reference value field (ref1val1).

```
<input type="hidden" name="ref1type1" value="EMAIL"><br>
<input type="hidden" name="ref1val1" value="joesmith@myschool.edu"><br>
```

In the example below, the database interprets this to mean that the EMAIL reference is being used, and the email address is 'joesmith@myschool.edu'. If we had two references, it might appear something like this:

```
<input type="hidden" name="ref1type1" value="EMAIL"><br>
<input type="hidden" name="ref1val1" value="joesmith@myschool.edu"><br>
<input type="hidden" name="ref2type1" value="DINNER"><br>
<input type="hidden" name="ref2val1" value="Chicken"><br>
```

Note that there are two references now associated with the first item code: EMAIL and DINNER. Joe's email address is 'joesmith@myschool.edu' and he would like chicken for dinner that evening.

Tip: References are often difficult to understand with checkout sites at first. If it helps, build the item in your TRAIN database within another eMarket or your ePayment site first so you can see how it appears in a normal store setting. Then, transfer the References into a form.

In addition, free form reference fields can cause malformed checkout requests if not properly filtered. For example, if a user enters an ampersand within a field and this is sent to the Transact Payments database, it will be interpreted as a separate field and cut off a portion of the request. Therefore, special characters should always be URL encoded.

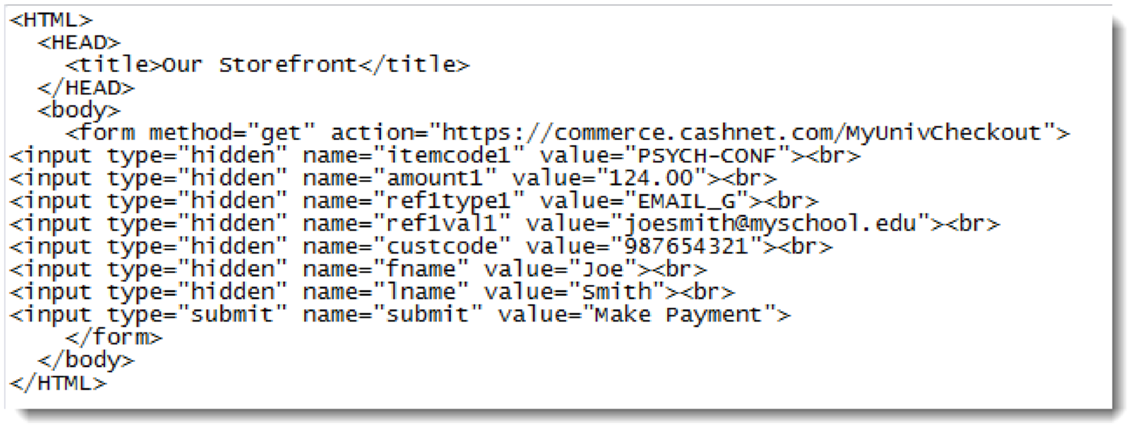
6.3.4 Full Request Flow

For a better understanding of how a Checkout works, let's walk through the entire process. First, we've determined we have a need for an event eMarket, so we perform the following steps:

1. We create the eMarket itself in the Transact Payments database following the Initial Setup and Store Setup sections of the guide and have implemented our store.
2. We create one item code called 'PSYCH-CON' and one reference called EMAIL.

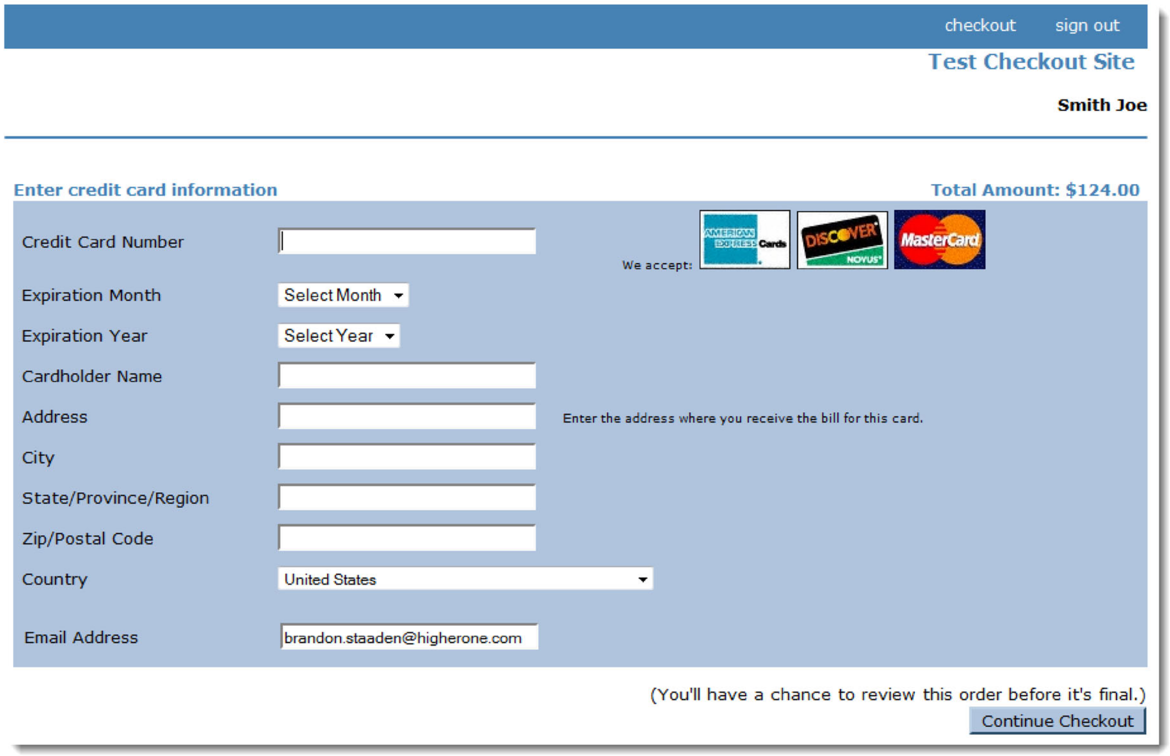
3. In order to test our eMarket checkout site, we create a form and submit it to the system. Assuming that the URL was correct and the Item Code and References were valid, the form will take us to the Transact Payments database.

Figure 59. Example HTML Form



```
<HTML>
  <HEAD>
    <title>Our Storefront</title>
  </HEAD>
  <body>
    <form method="get" action="https://commerce.cashnet.com/MyUnivCheckout">
<input type="hidden" name="itemcode1" value="PSYCH-CONF"><br>
<input type="hidden" name="amount1" value="124.00"><br>
<input type="hidden" name="ref1type1" value="EMAIL_G"><br>
<input type="hidden" name="ref1val1" value="joesmith@myschool.edu"><br>
<input type="hidden" name="custcode" value="987654321"><br>
<input type="hidden" name="fname" value="Joe"><br>
<input type="hidden" name="lname" value="Smith"><br>
<input type="submit" name="submit" value="Make Payment">
    </form>
  </body>
</HTML>
```

Figure 60. Test Checkout Site



Test Checkout Site

Smith Joe

Enter credit card information

Total Amount: $124.00

Credit Card Number	
Expiration Month	Select Month
Expiration Year	Select Year
Cardholder Name	
Address	
City	
State/Province/Region	
Zip/Postal Code	
Country	United States
Email Address	brandon.staaden@higherone.com

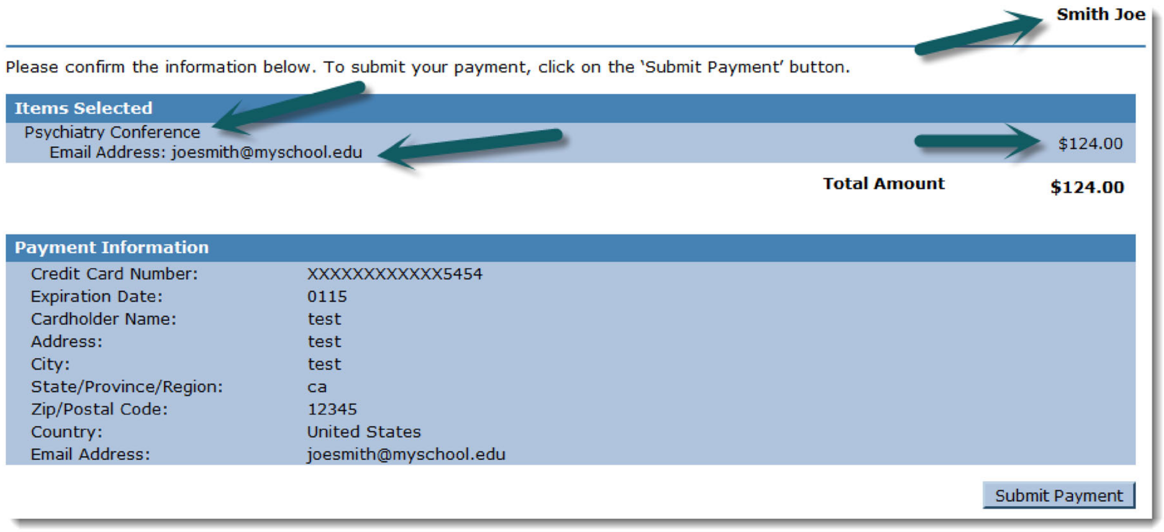
We accept:

Enter the address where you receive the bill for this card.

(You'll have a chance to review this order before it's final.)

4. After filling in the credit card information and clicking Continue Checkout, we should see a confirmation page. Here, we will want to verify that any data sent to the Transact Payments database is present (remember that references are only visible here if “Show on Receipt” was checked during reference setup).

Figure 61. Confirmation Page



Smith Joe

Please confirm the information below. To submit your payment, click on the 'Submit Payment' button.

Items Selected

Psychiatry Conference Email Address: joesmith@myschool.edu	$124.00
Total Amount	**$124.00**

Payment Information

Credit Card Number:	XXXXXXXXXXXX5454
Expiration Date:	0115
Cardholder Name:	test
Address:	test
City:	test
State/Province/Region:	ca
Zip/Postal Code:	12345
Country:	United States
Email Address:	joesmith@myschool.edu

5. After submitting the payment, you should be presented with a receipt.

Note: Creating a form manually is not the only way to access a checkout site. An HTTP GET request can be constructed by simply attaching fields to a URL. For example, the form referenced throughout this section can be represented as the following URL string:

```
https://commerce.cashnet.com/MyUnivCheckout?itemcode1=PSYCH-
CONF&amount1=124.00&ref1type1=EMAIL_G&ref1val1=joesmith@myschool.edu&custcode
=987654321&fname=Joe&lname=Smith
```

Be careful when creating test URL strings of this nature so that you do not inadvertently send checkout requests to your production database.

6.4 Notes on Integration

Checkout sites must often be integrated with other systems. If you already have an extract, it's possible that the items purchased through a Checkout can be sent back to your ERP through one of the extracts. You may require assistance in configuring the items to extract properly.

Some institutions prefer replying on reports instead of automatically extracting transactions. This is an easier method from a technical perspective and requires no additional configuration beyond setting up the report required.

Notifications can also be used in lieu of an extract. It's important to remember, however, that there is no verification that notifications have been received. Therefore, we suggest you back up your notifications with a report so that receipt of the transaction can be verified. If a new extract is required, these cannot be configured on your own. You will need to contact your Client Manager or your Implementation Team for assistance.