Merchant Integration Guide

.NET API with Visa, MasterCard

Level 2/3 Addendum Data Support

V1.1.0

Revision Number	**Date**	**Description**
V1.0	December 02, 2013	-Initial Draft
V1.0.1	November 19th, 2014	-Added Visa and Master Level 2/3 addendum data

Table of Contents

1. About this Documentation

This documentation contains the basic information for using the .NET API for sending credit card transactions. In particular it describes the format for sending transactions and the corresponding responses you will receive. This document is to be used by merchants that require the ability to pass Level 2 / 3 data for Visa, MasterCard and American Express.

2. System and Skill Requirements

In order to use .NET your system will need to have the following:
1. A web server with an SSL certificate
2. .NET Framework
3. Port 443 open for bi-directional communication

As well, you will need to have the following knowledge and/or skill set:
1. Install a dll into the global assembly cache
2. Knowledge of the .NET Framework

Note:

It is important to note that all Merchants and Service Providers that store, process, or transmit cardholder data must comply with PCI DSS and the Card Association Compliance Programs. However, certification requirements vary by business and are contingent upon your "Merchant Level" or "Service Provider Level". Failure to comply with PCI DSS and the Card Association Compliance Programs may result in a Merchant being subject to fines, fees or assessments and/or termination of processing services. Non-compliant solutions may prevent merchants boarding with Moneris Solutions.

As a Moneris Solutions client or partner using this method of integration, your solution must demonstrate compliance to the Payment Card Industry Data Security Standard (PCI DSS) and/or the Payment Application Data Security Standard (PA DSS) 2.0. These standards are designed to help the cardholders and merchants in such ways as they ensure credit card numbers are encrypted when transmitted/stored in a database and that merchants have strong access control measures, logging, secure software updates, secure remote access and support.

For further information on PCI DSS and PA DSS requirements, please visit http://www.pcisecuritystandards.org.

For more information on how to get your application PCI-DSS compliant, please contact our Integration Specialists and visit https://developer.moneris.com to download the PCI-DSS Implementation Guide.

3. What is the Process I will need to follow?

You will need to follow these steps.
1. Do the required development as outlined in this document
2. Test your solution in the test environment
3. Activate your store
4. Make the necessary changes to move your solution from the development environment into production as outlined in this document

4. Transaction Types and Flow for Non-Level 2 / 3 supported card types

eSelectplus supports a wide variety of Level1 transactions through the API. Below is a list of transaction supported by the API; other terms used for the transaction type are indicated in brackets.

Purchase – (sale) The purchase transaction verifies funds on the customer's card, removes the funds and readies them for deposit into the merchant account.

Preauth – (authorisation / preauthorisation) The preauth verifies and locks funds on the customer's credit card. The funds are locked for a specified amount of time, based on the card issuer. To retrieve the funds from a preauth so that they may be settled in the merchant account a capture must be performed.

Capture – (Completion/Preauth Completion) Once a Preauth is obtained the funds that are locked need to be retrieved from the customer's credit card. The capture retrieves the locked funds and readies them for settlement in to the merchant account.

Void – (Correction, Purchase Correction) Purchase and Captures can be voided the same day* that they occur. A void must be for the full amount of the transaction and will remove any record of it from the cardholder statement.

Refund – (Credit) A refund can be performed against a purchase or a capture to refund any part, or all of the transaction.

Force Post – (Offline Sale) The Force Post is used when a merchant obtains the authorization number directly from the issuer using a phone or any third party authorization method. The Force Post retrieves the locked funds and readies them for settlement into the merchant's account.

Independent Refund – (Credit) An Independent Refund can be performed to credit money to a Credit Card. This transaction does not require a prior Purchase or Capture. Please note, the Independent Refund transaction may or may not be supported on your account. If you receive a transaction not allowed error when attempting an independent refund, it may mean the transaction is not supported on your account. If you wish to have the Independent Refund transaction type temporarily enabled (or re-enabled), please contact the Service Centre at 1-800-471-9511.

Batch Close – (End of Day/Settlement) When a batch close is performed it takes the monies from all purchase, capture, void and refund transactions so they will be deposited the following business day. For funds to be deposited the following business day the batch must close before 11pm EST.

* A void can be performed against a transaction as long as the batch that contains the original transaction remains open. When using the automated closing feature batch close occurs daily between 10 – 11 pm EST.

5. Transaction Examples for Non-Level 2 / 3 supported card types

Included below is the sample code that can be found in the “Examples” folder of the API download.

Purchase (basic)

In the Purchase (USPurchase) example we require several variables. store_id, api_token, order_id, amount, pan (credit card number), expiry date and crypt type). There are also a number of optional fields, such as cust_id, dynamic_descriptor, commcard_invoice and commcard_tax_amount) available for Corporate Purchasing Cards. Please refer to Appendix A. Definition of Request Fields for variable definitions.

```
namespace USMoneris
{
    using System;
        public class TestPurchase
        {
          public static void Main(string[] args)
          {
                    /******************* REQUEST VARIABLES*******************************/

                string host = "esplusqa.moneris.com";
                string store_id = "monusqa002";
                string api_token = "qatoken";
            //string status = "true";

                    /****************** TRANSACTION VARIABLES *****************************/

                string order_id;                //will prompt user for input
                string amount = "5.00";
                string card = "4242424242424242";
                string exp = "1212";
                string crypt = "7";
                string commcard_invoice = "INVC090";
                string commcard_tax_amount = "1.00";

                    Console.Write ("Please enter an order ID: ");
                    order_id = Console.ReadLine();

               HttpsPostRequest mpgReq =
                    new HttpsPostRequest(host, store_id, api_token,
                               new USPurchase(order_id,
                                            amount,
                                            card,
                                            exp,
                                            crypt,
                                            commcard_invoice,
                                            commcard_tax_amount));

          /*Status Check Example
           HttpsPostRequest mpgReq =
                    new HttpsPostRequest(host, store_id, api_token, status,
                               new USPurchase(order_id,
                                            amount,
                                            card,
                                            exp,
                                            crypt,
                                            commcard_invoice,
                                            commcard_tax_amount));
           */




                try
                {
                    Receipt receipt = mpgReq.GetReceipt();

                        Console.WriteLine("CardType = " + receipt.GetCardType());
                        Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                        Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                        Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                        Console.WriteLine("TransType = " + receipt.GetTransType());
                        Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                        Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                        Console.WriteLine("Message = " + receipt.GetMessage());
                        Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                        Console.WriteLine("Complete = " + receipt.GetComplete());
                        Console.WriteLine("TransDate = " + receipt.GetTransDate());
                        Console.WriteLine("TransTime = " + receipt.GetTransTime());
                        Console.WriteLine("Ticket = " + receipt.GetTicket());
                        Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                Console.WriteLine("CardLevelResult = " + receipt.GetCardLevelResult());
                //Console.WriteLine("StatusCode = " + receipt.GetStatusCode());
                //Console.WriteLine("StatusMessage = " + receipt.GetStatusMessage());

                }
                catch (Exception e)
                {
                    Console.WriteLine(e);
                }
          }

        }
}
```

Preauth (basic)

The PreAuth (USPreAuth) transaction is virtually identical to the purchase with the exception of the transaction type. It is ‘USPreAuth’ instead of ‘USPurchase’. Like the Purchase example, PreAuths require several variables. store_id, api_token, order_id, amount, pan (credit card number), expiry date, and crypt type. There are also a number of optional fields, such as cust_id, dynamic_descriptor, commcard_invoice and commcard_tax_amount) available for Corporate Purchasing Cards. Please refer to Appendix A. Definition of Request Fields for variable definitions.

```
namespace USMoneris
{
    using System;

        public class TestPreAuth
        {
          public static void Main(string[] args)
          {

                string host = "esplusqa.moneris.com";
                string store_id = "monus00001";
                string api_token = "montoken";
        //string status = "true";

        string order_id;                //will prompt for user input
                string amount = "9.00";
                string card = "5550080000122007";
                string exp_date = "1411";
                string crypt = "7";

                Console.Write ("Please enter an order ID: ");
                order_id = Console.ReadLine();

                HttpsPostRequest mpgReq =
                    new HttpsPostRequest(host, store_id, api_token,
                               new USPreAuth(order_id, amount, card, exp_date, crypt));

            /*Status Check Example
                   HttpsPostRequest mpgReq =
                  new HttpsPostRequest(host, store_id, api_token, status,
                             new USPreAuth(order_id, amount, card, exp_date, crypt));
            */


          /**********************   REQUEST  ************************/

            try
                {
                    Receipt receipt = mpgReq.GetReceipt();

                        Console.WriteLine("CardType = " + receipt.GetCardType());
                        Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                        Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                        Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                        Console.WriteLine("TransType = " + receipt.GetTransType());
                        Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                        Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                        Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                        Console.WriteLine("Message = " + receipt.GetMessage());
                        Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                        Console.WriteLine("Complete = " + receipt.GetComplete());
                        Console.WriteLine("TransDate = " + receipt.GetTransDate());
                        Console.WriteLine("TransTime = " + receipt.GetTransTime());
                        Console.WriteLine("Ticket = " + receipt.GetTicket());
                        Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                Console.WriteLine("CardLevelResult = " + receipt.GetCardLevelResult());
                //Console.WriteLine("StatusCode = " + receipt.GetStatusCode());
                //Console.WriteLine("StatusMessage = " + receipt.GetStatusMessage());

                }
                catch (Exception e)
                {
                    Console.WriteLine(e);
                }
          }

        }
}
```

Capture

The capture (USCompletion) transaction is used to secure the funds locked by an USPreAuth transaction. When sending an 'USCompletion' request you will need two pieces of information from the original USPreAuth – the order_id and the txn_number from the returned response.

To reverse the full amount of the PreAuth, please use the Capture transaction with a dollar amount of "0.00".

```
namespace USMoneris
{
    using System;
        public class TestCompletion
        {
          public static void Main(string[] args)
          {
                  /****************** REQUEST VARIABLES*******************************/

                  string host = "esplusqa.moneris.com";
                  string store_id = "monusqa002";
                  string api_token = "qatoken";
          //string status = "true";

                  /***************** TRANSACTION VARIABLES ****************************/

                  string order_id;              //will prompt user for input
                  string txn_number;
                  string amount = "5.00";
                  string crypt = "7";
                  string commcard_invoice = "INVC090";
                  string commcard_tax_amount = "1.00";
          string dynamic_descriptor = "123456";

                  Console.Write ("Please enter an order ID: ");
                  order_id = Console.ReadLine();

                  Console.Write ("Please enter a txn number: ");
                  txn_number= Console.ReadLine();

          USCompletion c = new USCompletion(order_id, amount, txn_number, crypt,        commcard_invoice,
commcard_tax_amount);

          c.SetDynamicDescriptor(dynamic_descriptor);

              HttpsPostRequest mpgReq = new HttpsPostRequest(host, store_id, api_token, c);


          /* Status Check Example
           HttpsPostRequest mpgReq =
                     new HttpsPostRequest(host, store_id, api_token, status,
                             new USCompletion(order_id,
                                                                                    amount,
                                                                                    txn_number,
                                                                                          crypt,
                                                                                          commcard_invoice,
                                                                                          commcard_tax_amount
                                                                                          )
                                                                                  );
            */

                 try
                 {
                     Receipt receipt = mpgReq.GetReceipt();

                         Console.WriteLine("CardType = " + receipt.GetCardType());
                         Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                         Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                         Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                         Console.WriteLine("TransType = " + receipt.GetTransType());
                         Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                         Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                         Console.WriteLine("ISO = " + receipt.GetISO());
                         Console.WriteLine("Message = " + receipt.GetMessage());
                         Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                         Console.WriteLine("Complete = " + receipt.GetComplete());
                         Console.WriteLine("TransDate = " + receipt.GetTransDate());
                         Console.WriteLine("TransTime = " + receipt.GetTransTime());
                         Console.WriteLine("Ticket = " + receipt.GetTicket());
                         Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                 Console.WriteLine("CardLevelResult = " + receipt.GetCardLevelResult());
                 //Console.WriteLine("StatusCode = " + receipt.GetStatusCode());
                 //Console.WriteLine("StatusMessage = " + receipt.GetStatusMessage());

                 }
                 catch (Exception e)
                 {
                     Console.WriteLine(e);
                 }
           }

         }
}
```

Void

The void (USPurchaseCorrection) transaction is used to cancel a transaction that was performed in the current batch. No amount is required because a void is always for 100% of the original transaction. The only transactions that can be voided are captures and purchases. To send a void the order_id and txn_number from the capture or purchase are required.

```
namespace USMoneris
{
    using System;
        public class TestPurchaseCorrection
        {
          public static void Main(string[] args)
          {
                  /******************* REQUEST VARIABLES*******************************/

                  string host = "esplusqa.moneris.com";
                  string store_id = "monusqa002";
                  string api_token = "qatoken";
          //string status = "true";

                  /***************** TRANSACTION VARIABLES ****************************/

                  string order_id;              //will prompt user for input
                  string txn_number;
                  string crypt = "7";
          string dynamic_descriptor = "123456";

                  Console.Write ("Please enter an order ID: ");
                  order_id = Console.ReadLine();

                  Console.Write ("Please enter a txn number: ");
                  txn_number= Console.ReadLine();

          USPurchaseCorrection pc = new USPurchaseCorrection(order_id, txn_number, crypt);
          pc.SetDynamicDescriptor(dynamic_descriptor);

                  HttpsPostRequest mpgReq = new HttpsPostRequest(host, store_id, api_token, pc);

          /*Status Check Example
          HttpsPostRequest mpgReq =
            new HttpsPostRequest(host, store_id, api_token, status,
                       new USPurchaseCorrection(order_id, txn_number, crypt));
           */
                try
                {
                    Receipt receipt = mpgReq.GetReceipt();

                        Console.WriteLine("CardType = " + receipt.GetCardType());
                        Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                        Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                        Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                        Console.WriteLine("TransType = " + receipt.GetTransType());
                        Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                        Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                        Console.WriteLine("ISO = " + receipt.GetISO());
                        Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                        Console.WriteLine("Message = " + receipt.GetMessage());
                        Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                        Console.WriteLine("Complete = " + receipt.GetComplete());
                        Console.WriteLine("TransDate = " + receipt.GetTransDate());
                        Console.WriteLine("TransTime = " + receipt.GetTransTime());
                        Console.WriteLine("Ticket = " + receipt.GetTicket());
                        Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                //Console.WriteLine("StatusCode = " + receipt.GetStatusCode());
                //Console.WriteLine("StatusMessage = " + receipt.GetStatusMessage());

                }
                catch (Exception e)
                {
                    Console.WriteLine(e);
                }
          }

        }
}
```

Refund

The Refund (USRefund) transaction will credit a specified amount to the cardholder's credit card. A Refund can be sent up to the full value of the original Capture or Purchase. To send an 'USRefund' you will require the order_id and txn_number from the original 'USCompletion' or 'USPurchase'.

```
namespace USMoneris
{
    using System;
        public class TestRefund
        {
          public static void Main(string[] args)
          {
                  /****************** REQUEST VARIABLES*******************************/

                  string host = "esplusqa.moneris.com";
                  string store_id = "monusqa002";
                  string api_token = "qatoken";
          //string status = "true";

                  /***************** TRANSACTION VARIABLES *****************************/

                  string order_id;             //will prompt user for input
                  string amount = "1.00";
                  string txn_number;
                  string crypt = "7";
          string dynamic_descriptor = "123456";

                  Console.Write ("Please enter an order ID: ");
                  order_id = Console.ReadLine();

                  Console.Write ("Please enter a txn number: ");
                  txn_number= Console.ReadLine();

              USRefund r = new USRefund(order_id, amount, txn_number, crypt);

          r.SetDynamicDescriptor(dynamic_descriptor);

          HttpsPostRequest mpgReq = new HttpsPostRequest(host, store_id, api_token, r);

          /*Status Check Example
           HttpsPostRequest mpgReq =
                    new HttpsPostRequest(host, store_id, api_token, status,
                               new USRefund(order_id, amount, txn_number, crypt));
          */
                try
                {
                    Receipt receipt = mpgReq.GetReceipt();

                        Console.WriteLine("CardType = " + receipt.GetCardType());
                        Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                        Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                        Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                        Console.WriteLine("TransType = " + receipt.GetTransType());
                        Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                        Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                        Console.WriteLine("ISO = " + receipt.GetISO());
                        Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                        Console.WriteLine("Message = " + receipt.GetMessage());
                        Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                        Console.WriteLine("Complete = " + receipt.GetComplete());
                        Console.WriteLine("TransDate = " + receipt.GetTransDate());
                        Console.WriteLine("TransTime = " + receipt.GetTransTime());
                        Console.WriteLine("Ticket = " + receipt.GetTicket());
                        Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                //Console.WriteLine("StatusCode = " + receipt.GetStatusCode());
                //Console.WriteLine("StatusMessage = " + receipt.GetStatusMessage());

                }
                catch (Exception e)
                {
                    Console.WriteLine(e);
                }
          }

        }
  }
```

Independent Refund

The Independent Refund (USIndependentRefund) will credit a specified amount to the cardholder's credit card. The Independent Refund does not require an existing order ID to be logged in the eSelect plus gateway; however, the credit card number and expiry date will need to be passed. The Independent Refund transaction requires several variables. store_id, api_token, order_id, amount, pan, expiry date and crypt type). There are also optional fields, such as cust_id and dynamic_descriptor. The transaction format is almost identical to a Purchase or a PreAuth.

NOTE The Independent Refund transaction may or may not be supported on your account. If you receive a transaction not allowed error when attempting an **independent refund**, it may mean the transaction is not supported on your account. If you wish to have the Independent Refund transaction type temporarily enabled (or re-enabled), please contact the Service Centre at 1-800-471-9511.

```
namespace USMoneris
{
    using System;
        public class TestIndependentRefund
        {
          public static void Main(string[] args)
          {

                  /******************* REQUEST VARIABLES*******************************/
          string host = "esplusqa.moneris.com";
          string store_id = "monusqa002";
          string api_token = "qatoken";
          //string status = "true";

                  /****************** TRANSACTION VARIABLES *****************************/
                  string order_id;              //will prompt user for input
                  string cust_id = "Ced_Benson32";
                  string amount = "5.00";
                  string card = "4242424242424242";
                  string exp = "1212";
                  string crypt= "7";
                  Console.Write ("Please enter an order ID: ");
                  order_id = Console.ReadLine();
                HttpsPostRequest mpgReq =
                    new HttpsPostRequest(host, store_id, api_token,
                               new USIndependentRefund(order_id, cust_id, amount, card, exp, crypt));

          /*Status Check Example
            HttpsPostRequest mpgReq =
                    new HttpsPostRequest(host, store_id, api_token, status,
                               new USIndependentRefund(order_id, cust_id, amount, card, exp, crypt));
           */
                try
                {
                    Receipt receipt = mpgReq.GetReceipt();

                        Console.WriteLine("CardType = " + receipt.GetCardType());
                        Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                        Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                        Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                        Console.WriteLine("TransType = " + receipt.GetTransType());
                        Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                        Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                        Console.WriteLine("ISO = " + receipt.GetISO());
                        Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                        Console.WriteLine("Message = " + receipt.GetMessage());
                        Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                        Console.WriteLine("Complete = " + receipt.GetComplete());
                        Console.WriteLine("TransDate = " + receipt.GetTransDate());
                        Console.WriteLine("TransTime = " + receipt.GetTransTime());
                        Console.WriteLine("Ticket = " + receipt.GetTicket());
                        Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                Console.WriteLine("CardLevelResult = " + receipt.GetCardLevelResult());
                //Console.WriteLine("StatusCode = " + receipt.GetStatusCode());
                //Console.WriteLine("StatusMessage = " + receipt.GetStatusMessage());

                }
                catch (Exception e)
                {
                    Console.WriteLine(e);
                }
          }
```

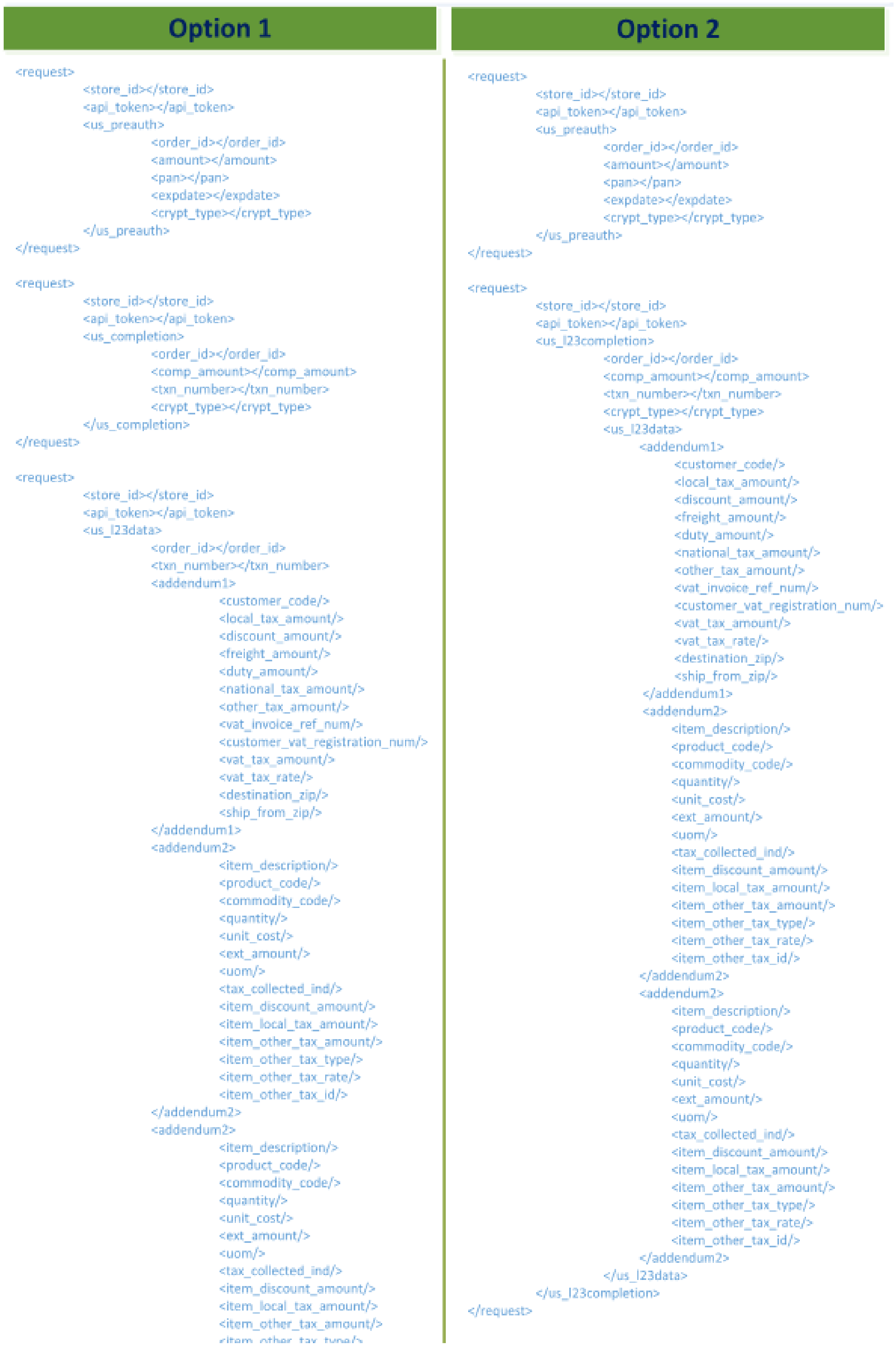
Batch Close

At the end of every day (11pm EST) the batch needs to be closed in order to have the funds settled the next business day. ***By default eSelectplus will close your batch automatically for you daily whenever there are funds in the open batch.*** Some merchants prefer to control batch close, and disable the automatic functionality. For these merchants we have provided the ability to close your batch through the API. When a batch is closed the response will include the transaction count and amount for each type of transaction for each type of card. To disable automatic close please access the Merchant Resource Centre (https://esplus.moneris.com/usmpg), go to the Admin menu and then choose Store Settings: the Batch Close options are located on this page.

```
namespace USMoneris
{
    using System;
        public class TestBatchClose
        {
          public static void Main(string[] args)
          {
          string host = "esplusqa.moneris.com";
          string store_id = "monusqa002";
          string api_token = "qatoken";

            string ecr_no = "64000001";//ecr within store  "64000001"

                HttpsPostRequest mpgReq =
                    new HttpsPostRequest(host,store_id, api_token, new USBatchClose (ecr_no));
                try
                {
                    Receipt receipt = mpgReq.GetReceipt();
                    if ( (receipt.GetReceiptId()).Equals("Global Error Receipt") )


                        {


                            Console.WriteLine("CardType = " + receipt.GetCardType());
                            Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                            Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                            Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                            Console.WriteLine("TransType = " + receipt.GetTransType());
                            Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                            Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                            Console.WriteLine("ISO = " + receipt.GetISO());
                            Console.WriteLine("BankTotals = null");
                            Console.WriteLine("Message = " + receipt.GetMessage());
                            Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                            Console.WriteLine("Complete = " + receipt.GetComplete());
                            Console.WriteLine("TransDate = " + receipt.GetTransDate());
                            Console.WriteLine("TransTime = " + receipt.GetTransTime());
                            Console.WriteLine("Ticket = " + receipt.GetTicket());
                            Console.WriteLine("TimedOut = " + receipt.GetTimedOut());


                    }
                    else
                    {

                         foreach ( string ecr in receipt.GetTerminalIDs() )
                            {
                                Console.WriteLine("ECR: " + ecr);
                                foreach ( string cardType in receipt.GetCreditCards(ecr) )
                                {
                                    Console.WriteLine("\tCard Type: " + cardType);

                                    Console.WriteLine("\t\tPurchase: Count = "
                                                    + receipt.GetPurchaseCount(ecr,cardType)
                                                    + " Amount = "
                                                    + receipt.GetPurchaseAmount(ecr,
                                                                        cardType));
                                    Console.WriteLine("\t\tRefund: Count = "
                                                    + receipt.GetRefundCount(ecr,cardType)
                                                    + " Amount = "
                                                    + receipt.GetRefundAmount(ecr,cardType));
```

```
                              Console.WriteLine("\t\tCorrection: Count = "
                                          + receipt.GetCorrectionCount(ecr,cardType)
                                          + " Amount = "
                                          + receipt.GetCorrectionAmount(ecr,
                                                        cardType));
                        }
                    }
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e);
        }
        }
```

6. Process Flow for US Visa and MasterCard Level 2/3



Option 1

```
<request>
        <store_id></store_id>
        <api_token></api_token>
        <us_preauth>
                <order_id></order_id>
                <amount></amount>
                <pan></pan>
                <expdate></expdate>
                <crypt_type></crypt_type>
        </us_preauth>
</request>

<request>
        <store_id></store_id>
        <api_token></api_token>
        <us_completion>
                <order_id></order_id>
                <comp_amount></comp_amount>
                <txn_number></txn_number>
                <crypt_type></crypt_type>
        </us_completion>
</request>

<request>
        <store_id></store_id>
        <api_token></api_token>
        <us_l23data>
                <order_id></order_id>
                <txn_number></txn_number>
                <addendum1>
                        <customer_code/>
                        <local_tax_amount/>
                        <discount_amount/>
                        <freight_amount/>
                        <duty_amount/>
                        <national_tax_amount/>
                        <other_tax_amount/>
                        <vat_invoice_ref_num/>
                        <customer_vat_registration_num/>
                        <vat_tax_amount/>
                        <vat_tax_rate/>
                        <destination_zip/>
                        <ship_from_zip/>
                </addendum1>
                <addendum2>
                        <item_description/>
                        <product_code/>
                        <commodity_code/>
                        <quantity/>
                        <unit_cost/>
                        <ext_amount/>
                        <uom/>
                        <tax_collected_ind/>
                        <item_discount_amount/>
                        <item_local_tax_amount/>
                        <item_other_tax_amount/>
                        <item_other_tax_type/>
                        <item_other_tax_rate/>
                        <item_other_tax_id/>
                </addendum2>
                <addendum2>
                        <item_description/>
                        <product_code/>
                        <commodity_code/>
                        <quantity/>
                        <unit_cost/>
                        <ext_amount/>
                        <uom/>
                        <tax_collected_ind/>
                        <item_discount_amount/>
                        <item_local_tax_amount/>
                        <item_other_tax_amount/>
                        <item_other_tax_type/>
```

Option 2

```
<request>
        <store_id></store_id>
        <api_token></api_token>
        <us_preauth>
                <order_id></order_id>
                <amount></amount>
                <pan></pan>
                <expdate></expdate>
                <crypt_type></crypt_type>
        </us_preauth>
</request>

<request>
        <store_id></store_id>
        <api_token></api_token>
        <us_l23completion>
                <order_id></order_id>
                <comp_amount></comp_amount>
                <txn_number></txn_number>
                <crypt_type></crypt_type>
                <us_l23data>
                    <addendum1>
                        <customer_code/>
                        <local_tax_amount/>
                        <discount_amount/>
                        <freight_amount/>
                        <duty_amount/>
                        <national_tax_amount/>
                        <other_tax_amount/>
                        <vat_invoice_ref_num/>
                        <customer_vat_registration_num/>
                        <vat_tax_amount/>
                        <vat_tax_rate/>
                        <destination_zip/>
                        <ship_from_zip/>
                    </addendum1>
                    <addendum2>
                        <item_description/>
                        <product_code/>
                        <commodity_code/>
                        <quantity/>
                        <unit_cost/>
                        <ext_amount/>
                        <uom/>
                        <tax_collected_ind/>
                        <item_discount_amount/>
                        <item_local_tax_amount/>
                        <item_other_tax_amount/>
                        <item_other_tax_type/>
                        <item_other_tax_rate/>
                        <item_other_tax_id/>
                    </addendum2>
                    <addendum2>
                        <item_description/>
                        <product_code/>
                        <commodity_code/>
                        <quantity/>
                        <unit_cost/>
                        <ext_amount/>
                        <uom/>
                        <tax_collected_ind/>
                        <item_discount_amount/>
                        <item_local_tax_amount/>
                        <item_other_tax_amount/>
                        <item_other_tax_type/>
                        <item_other_tax_rate/>
                        <item_other_tax_id/>
                    </addendum2>
                </us_l23data>
        </us_l23completion>
</request>
```

7. Transaction Types and Transaction Flow for Visa and MasterCard Level 2/3

When support for Level2/3 transactions is enabled for Visa and Mastercard all Level 1 and Level 2/3 Visa and Mastercard transactions must be sent using the following transaction set. This set includes a suite of financial transactions as well as a transaction that allows for the passing of Level 2/3 data. Batch Close, Open Totals and PreAuth are identical to the non-level 2/3 transactions outlined in Sections 4 & 5. When the L23IsCorporateCard response contains CorporateCard equal to true then you can submit a Level2/3 data transaction. If CorporateCard is false then the card does not support Level 2/3 data.

*** Note: If you do not wish to send any Level 2/3 data then you may submit Visa and MasterCard transactions using the transaction set outlined in Section 4 & 5 (Non-level 2/3 transactions).*

L23IsCorporateCard – The Level 2/3 isCorporateCard verifies if the card is available for Corporate Purchasing. When CorporateCard is returned true then Level 2/3 data may be submitted.

PreAuth – (authorisation / preauthorisation) The PreAuth verifies and locks funds on the customer's credit card. The funds are locked for a specified amount of time, based on the card issuer. To retrieve the funds from a PreAuth so that they may be settled in the merchant account a capture must be performed. Level 2/3 data submission is not supported as part of a preauth as a preauth is not settled.

L23Completion – (Capture/Preauth Completion) Once a PreAuth is obtained the funds that are locked need to be retrieved from the customer's credit card. The Level 2/3 Capture retrieves the locked funds and readies them for settlement in to the merchant account. Prior to performing a level23Completion a PreAuth must be performed.

Level23PurchaseCorrection – (Void, Correction) Purchase and Captures can be voided the same day* that they occur. A void must be for the full amount of the transaction and will remove any record of it from the cardholder statement.

L23Refund – (Credit) A Refund can be performed against a Purchase or a Capture to refund any part, or all of the transaction.

L23IndependentRefund – (Credit) A Refund can be performed against a Purchase or a Capture to refund any part, or all of the transaction. Independent Refund is used when the originating transaction was not performed through eSelectplus.

L23Data – (Level 2/3 Data) The Level 2/3 Data will contain all the required and optional data fields for Level 2/3 data. L23Data can be sent when the card has been identified in the transaction request as being a corporate card.

L23ForcePost – A Level 2/3 ForcePost is a completion through the eSelect plus system; however, forcepost is used when the original preauth or voice auth was not performed through eSelect plus.

* A Level23PurchaseCorrection can be performed against a transaction as long as the batch that contains the original transaction remains open. When using the automated closing feature batch close occurs daily between 10 – 11 pm EST.

8. Transaction Examples for Visa and MasterCard Level 2/3

Included below is the sample code that can be found in the “Examples” folder of the API download.

L23IsCorporateCard

The L23IsCorporateCard requires several variables. store_id, api_token, order_id, pan (credit card number), expiry date. Please refer to Appendix A. Definition of Request Fields for variable definitions.

```
namespace USMoneris
{
        using System;
        public class TestL23IsCorporateCard
        {
                public static void Main(string[] args)
                {
                        /****************** REQUEST VARIABLES******************************/

                        string host = "esplusqa.moneris.com";
                        string store_id = "monus00001";
                        string api_token = "montoken";

                        /***************** TRANSACTION VARIABLES ****************************/

                        string pan = "5454545442424242";
                        string expiry_date = "1212";

                        USL23IsCorporateCard corpCard =
                                new USL23IsCorporateCard(pan, expiry_date);


                        HttpsPostRequest mpgReq =
                                        new HttpsPostRequest(host, store_id, api_token, corpCard);




                        try
                        {
                                Receipt receipt = mpgReq.GetReceipt();

                                Console.WriteLine("CardType = " + receipt.GetCardType());
                                Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                                Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                                Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                                Console.WriteLine("TransType = " + receipt.GetTransType());
                                Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                                Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                                Console.WriteLine("Message = " + receipt.GetMessage());
                                Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                                Console.WriteLine("Complete = " + receipt.GetComplete());
                                Console.WriteLine("TransDate = " + receipt.GetTransDate());
                                Console.WriteLine("TransTime = " + receipt.GetTransTime());
                                Console.WriteLine("Ticket = " + receipt.GetTicket());
                                Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                                Console.WriteLine("Corporate Card = " + receipt.GetIsCorporateCard());

                        }
                        catch (Exception e)
                        {
                                Console.WriteLine(e);
                        }
                }
        }
}
```

L23PreAuth

The PreAuth (USPreAuth) requires several variables. store_id, api_token, order_id, amount, pan (credit card number), expiry date, and crypt type. Please refer to Appendix A. Definition of Request Fields for variable definitions.

```
namespace USMoneris
{
   using System;

                public class TestPreAuth
                {
                  public static void Main(string[] args)
                  {

                                string host = "esplusqa.moneris.com";
                                string store_id = "monus00001";
                                string api_token = "montoken";
      //string status = "true";

      string order_id;                   //will prompt for user input
                                string amount = "9.00";
                                string card = "5550080000122007";
                                string exp_date = "1411";
                                string crypt = "7";

                                Console.Write ("Please enter an order ID: ");
                                order_id = Console.ReadLine();

                        HttpsPostRequest mpgReq =
                           new HttpsPostRequest(host, store_id, api_token,
                                  new USPreAuth(order_id, amount, card, exp_date, crypt));

         /*Status Check Example
               HttpsPostRequest mpgReq =
              new HttpsPostRequest(host, store_id, api_token, status,
                     new USPreAuth(order_id, amount, card, exp_date, crypt));
         */


        /**********************   REQUEST  ***********************/

         try
                   {
                      Receipt receipt = mpgReq.GetReceipt();

                                            Console.WriteLine("CardType = " + receipt.GetCardType());
                                            Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                                            Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                                            Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                                            Console.WriteLine("TransType = " + receipt.GetTransType());
                                            Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                                            Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                                            Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                                            Console.WriteLine("Message = " + receipt.GetMessage());
                                            Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                                            Console.WriteLine("Complete = " + receipt.GetComplete());
                                            Console.WriteLine("TransDate = " + receipt.GetTransDate());
                                            Console.WriteLine("TransTime = " + receipt.GetTransTime());
                                            Console.WriteLine("Ticket = " + receipt.GetTicket());
                                            Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
           Console.WriteLine("CardLevelResult = " + receipt.GetCardLevelResult());
           //Console.WriteLine("StatusCode = " + receipt.GetStatusCode());
           //Console.WriteLine("StatusMessage = " + receipt.GetStatusMessage());

                   }
                   catch (Exception e)
                   {
                      Console.WriteLine(e);
                   }
                  }

                }
}
```

L23Completion

The Level 2/3 Completion (L23Completion) transaction is used to secure the funds locked by an 'USPreAuth' transaction. When sending a Capture request you will need two pieces of information from the original 'USPreAuth' – the order_id and the txn_number from the returned response. Please refer to Appendix A. Definition of Request Fields for variable definitions.

```
namespace USMoneris
{
        using System;
        public class TestL23Completion
        {
                public static void Main(string[] args)
                {
                        /****************** REQUEST VARIABLES*******************************/

                        string host = "esplusqa.moneris.com";
                        string store_id = "monus00001";
                        string api_token = "montoken";

                        /***************** TRANSACTION VARIABLES *****************************/

                        string order_id;              //will prompt user for input
                        string txn_number;
                        string amount = "5.00";
                        string crypt = "7";
                        string dynamic_descriptor = "123456";

                        Console.Write ("Please enter an order ID: ");
                        order_id = Console.ReadLine();

                        Console.Write ("Please enter a txn number: ");
                        txn_number= Console.ReadLine();

                        /******************** Addendum1  ****************************/
                        string customer_code = "ID12345";
                        string local_tax_amount = "1.00";
                        string discount_amount = "0.50";
                        string freight_amount = "0.50";
                        string duty_amount = "0.50";
                        string national_tax_amount = "0.00";
                        string other_tax_amount = "0.00";
                        string vat_invoice_ref_num = "123456789012345";
                        string customer_vat_registration_num = "1234567890123";
                        string vat_tax_amount = "1.00";
                        string vat_tax_rate = "0.00";
                        string destination_zip = "90210";
                        string ship_from_zip = "90210";

                        Addendum1 addendum1 = new Addendum1 (customer_code, local_tax_amount,
discount_amount, freight_amount,
                                duty_amount, national_tax_amount, other_tax_amount, vat_invoice_ref_num,
                                customer_vat_registration_num, vat_tax_amount, vat_tax_rate,
destination_zip, ship_from_zip);

                        /******************** Addendum2  ****************************/
                        string item_description1 = "Item1";
                        string product_code1 = "PROD00001";
                        string commodity_code1 = "IT1";
                        string quantity1 = "1.00";
                        string unit_cost1 = "1.00";
                        string ext_amount1 = "1.00";
                        string uom1 = "EA";
                        string tax_collected_ind1 = "Y";
                        string item_discount_amount1 = "0.25";
                        string item_local_tax_amount1 = "1.00";
                        string item_other_tax_amount1 = "1.00";
                        string item_other_tax_type1 = "0.00";
                        string item_other_tax_rate1 = "0.00";
                        string item_other_tax_id1 = "VAT";
                        Addendum2 addendum2 = new Addendum2 (item_description1, product_code1,
commodity_code1, quantity1,
                                unit_cost1, ext_amount1, uom1, tax_collected_ind1, item_discount_amount1,
                                item_local_tax_amount1, item_other_tax_amount1, item_other_tax_type1,
                                item_other_tax_rate1, item_other_tax_id1);

                        string item_description2 = "Item2";
                        string product_code2 = "PROD00002";
                        string commodity_code2 = "IT2";
                        string quantity2 = "2.00";
                        string unit_cost2 = "2.00";
                        string ext_amount2 = "2.00";
                        string uom2 = "EA";
                        string tax_collected_ind2 = "Y";
                        string item_discount_amount2 = "0.25";
                        string item_local_tax_amount2 = "2.00";
                        string item_other_tax_amount2 = "2.00";
                        string item_other_tax_type2 = "0.00";
                        string item_other_tax_rate2 = "0.00";
                        string item_other_tax_id2 = "VAT";

                        addendum2.AddAddendum2 (item_description2, product_code2, commodity_code2,
quantity2,
                                unit_cost2, ext_amount2, uom2, tax_collected_ind2, item_discount_amount2,
                                item_local_tax_amount2, item_other_tax_amount2, item_other_tax_type2,
                                item_other_tax_rate2, item_other_tax_id2);


                        USL23Completion c = new USL23Completion(order_id, amount, txn_number, crypt,
addendum1, addendum2);

                        c.SetDynamicDescriptor(dynamic_descriptor);

                        HttpsPostRequest mpgReq = new HttpsPostRequest(host, store_id, api_token, c);

                        try
                        {
                                Receipt receipt = mpgReq.GetReceipt();

                                Console.WriteLine("CardType = " + receipt.GetCardType());
                                Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                                Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                                Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                                Console.WriteLine("TransType = " + receipt.GetTransType());
                                Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                                Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                                Console.WriteLine("ISO = " + receipt.GetISO());
                                Console.WriteLine("Message = " + receipt.GetMessage());
                                Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                                Console.WriteLine("Complete = " + receipt.GetComplete());
                                Console.WriteLine("TransDate = " + receipt.GetTransDate());
                                Console.WriteLine("TransTime = " + receipt.GetTransTime());
                                Console.WriteLine("Ticket = " + receipt.GetTicket());
                                Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                                Console.WriteLine("CorporateCard = " + receipt.GetIsCorporateCard());

                        }
                        catch (Exception e)
                        {
                                Console.WriteLine(e);
                        }
                }

        }
}
```

L23PurchaseCorrection

The Level 2/3 Purchase Correction (L23PurchaseCorrection) transaction is used to cancel a transaction that was performed in the current batch. No amount is required because a void is always for 100% of the original transaction. The only transaction that can be voided is completion. To send a void the order_id and TxnNumber from the capture or purchase are required. Please refer to Appendix A. Definition of Request Fields for variable definitions.

```
namespace USMoneris
{
                using System;
        public class TestL23PurchaseCorrection
        {
                public static void Main(string[] args)
                {
                        /****************** REQUEST VARIABLES******************************/

                        string host = "esplusqa.moneris.com";
                        string store_id = "monusqa002";
                        string api_token = "qatoken";

                        /***************** TRANSACTION VARIABLES ****************************/

                        string order_id;              //will prompt user for input
                        string txn_number;
                        string crypt = "7";
                        string dynamic_descriptor = "123456";

                        Console.Write ("Please enter an order ID: ");
                        order_id = Console.ReadLine();

                        Console.Write ("Please enter a txn number: ");
                        txn_number= Console.ReadLine();

                        USL23PurchaseCorrection pc = new USL23PurchaseCorrection(order_id, txn_number,
crypt);

                        pc.SetDynamicDescriptor(dynamic_descriptor);

                        HttpsPostRequest mpgReq = new HttpsPostRequest(host, store_id, api_token, pc);

                        try
                        {
                                Receipt receipt = mpgReq.GetReceipt();

                                Console.WriteLine("CardType = " + receipt.GetCardType());
                                Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                                Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                                Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                                Console.WriteLine("TransType = " + receipt.GetTransType());
                                Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                                Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                                Console.WriteLine("ISO = " + receipt.GetISO());
                                Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                                Console.WriteLine("Message = " + receipt.GetMessage());
                                Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                                Console.WriteLine("Complete = " + receipt.GetComplete());
                                Console.WriteLine("TransDate = " + receipt.GetTransDate());
                                Console.WriteLine("TransTime = " + receipt.GetTransTime());
                                Console.WriteLine("Ticket = " + receipt.GetTicket());
                                Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                                Console.WriteLine("Corporate Card = " + receipt.GetIsCorporateCard());

                        }
                        catch (Exception e)
                        {
                                Console.WriteLine(e);
                        }
                }

        }
```

L23Refund

The Level 2/3 Refund (L23Refund) will credit a specified amount to the cardholder's credit card. A Refund can be sent up to the full value of the original Capture or Purchase. To send a Refund you will require the order_id and txn_number from the original capture or purchase. Please refer to Appendix A. Definition of Request Fields for variable definitions.

```
namespace USMoneris
{
        using System;
        public class TestL23Refund
        {
                public static void Main(string[] args)
                {
                        /****************** REQUEST VARIABLES*******************************/

                        string host = "esplusqa.moneris.com";
                        string store_id = "monusqa002";
                        string api_token = "qatoken";
                                        //string status = "true";

                        /***************** TRANSACTION VARIABLES ****************************/

                        string order_id;              //will prompt user for input
                        string amount = "1.00";
                        string txn_number;
                        string crypt = "7";
                                        string dynamic_descriptor = "123456";

                        Console.Write ("Please enter an order ID: ");
                        order_id = Console.ReadLine();

                        Console.Write ("Please enter a txn number: ");
                        txn_number= Console.ReadLine();

                        /******************** Addendum1  ****************************/
                        string customer_code = "ID12345";
                        string local_tax_amount = "1.00";
                        string discount_amount = "0.50";
                        string freight_amount = "0.50";
                        string duty_amount = "0.50";
                        string national_tax_amount = "0.00";
                        string other_tax_amount = "0.00";
                        string vat_invoice_ref_num = "123456789012345";
                        string customer_vat_registration_num = "1234567890123";
                        string vat_tax_amount = "1.00";
                        string vat_tax_rate = "0.00";
                        string destination_zip = "90210";
                        string ship_from_zip = "90210";

                        Addendum1 addendum1 = new Addendum1 (customer_code, local_tax_amount,
discount_amount, freight_amount,
                                duty_amount, national_tax_amount, other_tax_amount, vat_invoice_ref_num,
                                customer_vat_registration_num, vat_tax_amount, vat_tax_rate,
destination_zip, ship_from_zip);

                        /******************** Addendum2  ****************************/
                        string item_description1 = "Item1";
                        string product_code1 = "PROD00001";
                        string commodity_code1 = "IT1";
                        string quantity1 = "1.00";
                        string unit_cost1 = "1.00";
                        string ext_amount1 = "1.00";
                        string uom1 = "EA";
                        string tax_collected_ind1 = "Y";
                        string item_discount_amount1 = "0.25";
                        string item_local_tax_amount1 = "1.00";
                        string item_other_tax_amount1 = "1.00";
                        string item_other_tax_type1 = "0.00";
                        string item_other_tax_rate1 = "0.00";
                        string item_other_tax_id1 = "VAT";
                        Addendum2 addendum2 = new Addendum2 (item_description1, product_code1, 
commodity_code1, quantity1,
                                unit_cost1, ext_amount1, uom1, tax_collected_ind1, item_discount_amount1,
                                item_local_tax_amount1, item_other_tax_amount1, item_other_tax_type1,
                                item_other_tax_rate1, item_other_tax_id1);

                        string item_description2 = "Item2";
                        string product_code2 = "PROD00002";
                        string commodity_code2 = "IT2";
                        string quantity2 = "2.00";
                        string unit_cost2 = "2.00";
                        string ext_amount2 = "2.00";
                        string uom2 = "EA";
                        string tax_collected_ind2 = "Y";
                        string item_discount_amount2 = "0.25";
                        string item_local_tax_amount2 = "2.00";
                        string item_other_tax_amount2 = "2.00";
                        string item_other_tax_type2 = "0.00";
                        string item_other_tax_rate2 = "0.00";
                        string item_other_tax_id2 = "VAT";

                        addendum2.AddAddendum2 (item_description2, product_code2, commodity_code2, 
quantity2,
                                unit_cost2, ext_amount2, uom2, tax_collected_ind2, item_discount_amount2,
                                item_local_tax_amount2, item_other_tax_amount2, item_other_tax_type2,
                                item_other_tax_rate2, item_other_tax_id2);

                        USL23Refund r = new USL23Refund(order_id, amount, txn_number, crypt, addendum1, 
addendum2);

                        r.SetDynamicDescriptor(dynamic_descriptor);

                        HttpsPostRequest mpgReq = new HttpsPostRequest(host, store_id, api_token, r);


                        try
                        {
                                Receipt receipt = mpgReq.GetReceipt();

                                Console.WriteLine("CardType = " + receipt.GetCardType());
                                Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                                Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                                Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                                Console.WriteLine("TransType = " + receipt.GetTransType());
                                Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                                Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                                Console.WriteLine("ISO = " + receipt.GetISO());
                                Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                                Console.WriteLine("Message = " + receipt.GetMessage());
                                Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                                Console.WriteLine("Complete = " + receipt.GetComplete());
                                Console.WriteLine("TransDate = " + receipt.GetTransDate());
                                Console.WriteLine("TransTime = " + receipt.GetTransTime());
                                Console.WriteLine("Ticket = " + receipt.GetTicket());
                                Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                                Console.WriteLine("Corporate Card = " + receipt.GetIsCorporateCard());

                        }
                        catch (Exception e)
                        {
                                Console.WriteLine(e);
                        }
                }

        }
} // end TestL23Refund
```

L23IndependentRefund

The Level 2/3 Independent Refund (L23IndependentRefund) will credit a specified amount to the cardholder's credit card. The Independent Refund does not require an existing order to be logged in the eSelectplus gateway; however, the credit card number and expiry date will need to be passed. The transaction format is almost identical to a PreAuth. Please refer to Appendix A. Definition of Request Fields for variable definitions

```
namespace USMoneris
{
        using System;
        public class TestL23IndependentRefund
        {
                public static void Main(string[] args)
                {

                        /****************** REQUEST VARIABLES*******************************/

                        string host = "esplusqa.moneris.com";
                        string store_id = "monus00001";
                        string api_token = "montoken";

                        /***************** TRANSACTION VARIABLES ****************************/

                        string order_id;              //will prompt user for input
                        string cust_id = "";
                        string amount = "5.00";
                        string card = "4242424242424242";
                        string exp = "1212";
                        string crypt= "7";

                        Console.Write ("Please enter an order ID: ");
                        order_id = Console.ReadLine();

                        /******************** Addendum1  ****************************/
                        string customer_code = "ID12345";
                        string local_tax_amount = "1.00";
                        string discount_amount = "0.50";
                        string freight_amount = "0.50";
                        string duty_amount = "0.50";
                        string national_tax_amount = "0.00";
                        string other_tax_amount = "0.00";
                        string vat_invoice_ref_num = "123456789012345";
                        string customer_vat_registration_num = "1234567890123";
                        string vat_tax_amount = "1.00";
                        string vat_tax_rate = "0.00";
                        string destination_zip = "90210";
                        string ship_from_zip = "90210";

                        Addendum1 addendum1 = new Addendum1 (customer_code, local_tax_amount,
discount_amount, freight_amount,
                                duty_amount, national_tax_amount, other_tax_amount, vat_invoice_ref_num,
                                customer_vat_registration_num, vat_tax_amount, vat_tax_rate,
destination_zip, ship_from_zip);

                        /******************** Addendum2  ****************************/
                        string item_description1 = "Item1";
                        string product_code1 = "PROD00001";
                        string commodity_code1 = "IT1";
                        string quantity1 = "1.00";
                        string unit_cost1 = "1.00";
                        string ext_amount1 = "1.00";
                        string uom1 = "EA";
                        string tax_collected_ind1 = "Y";
                        string item_discount_amount1 = "0.25";
                        string item_local_tax_amount1 = "1.00";
                        string item_other_tax_amount1 = "1.00";
                        string item_other_tax_type1 = "0.00";
                        string item_other_tax_rate1 = "0.00";
                        string item_other_tax_id1 = "VAT";
```

```
                      Addendum2 addendum2 = new Addendum2 (item_description1, product_code1,
commodity_code1, quantity1,
                             unit_cost1, ext_amount1, uom1, tax_collected_ind1, item_discount_amount1,
                             item_local_tax_amount1, item_other_tax_amount1, item_other_tax_type1,
                             item_other_tax_rate1, item_other_tax_id1);

                      string item_description2 = "Item2";
                      string product_code2 = "PROD00002";
                      string commodity_code2 = "IT2";
                      string quantity2 = "2.00";
                      string unit_cost2 = "2.00";
                      string ext_amount2 = "2.00";
                      string uom2 = "EA";
                      string tax_collected_ind2 = "Y";
                      string item_discount_amount2 = "0.25";
                      string item_local_tax_amount2 = "2.00";
                      string item_other_tax_amount2 = "2.00";
                      string item_other_tax_type2 = "0.00";
                      string item_other_tax_rate2 = "0.00";
                      string item_other_tax_id2 = "VAT";

                      addendum2.AddAddendum2 (item_description2, product_code2, commodity_code2,
quantity2,
                             unit_cost2, ext_amount2, uom2, tax_collected_ind2, item_discount_amount2,
                             item_local_tax_amount2, item_other_tax_amount2, item_other_tax_type2,
                             item_other_tax_rate2, item_other_tax_id2);


                      HttpsPostRequest mpgReq =
                                     new HttpsPostRequest(host, store_id, api_token,
                                                   new USL23IndependentRefund(order_id, cust_id, amount,
card, exp, crypt, addendum1, addendum2));

                      try
                      {
                             Receipt receipt = mpgReq.GetReceipt();

                             Console.WriteLine("CardType = " + receipt.GetCardType());
                             Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                             Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                             Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                             Console.WriteLine("TransType = " + receipt.GetTransType());
                             Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                             Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                             Console.WriteLine("ISO = " + receipt.GetISO());
                             Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                             Console.WriteLine("Message = " + receipt.GetMessage());
                             Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                             Console.WriteLine("Complete = " + receipt.GetComplete());
                             Console.WriteLine("TransDate = " + receipt.GetTransDate());
                             Console.WriteLine("TransTime = " + receipt.GetTransTime());
                             Console.WriteLine("Ticket = " + receipt.GetTicket());
                             Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                             Console.WriteLine("Corporate Card = " + receipt.GetIsCorporateCard());

                      }
                      catch (Exception e)
                      {
                             Console.WriteLine(e);
                      }
               }

        }
} // end TestL23IndependentRefund
```

L23Data

The MCLevel23 transaction request passes the Level 2 and 3 data for processing. The MCLevel23 request must be preceded by a financial transaction (completion, refund . . .) and the Corporate Card flag must be set to “true” in the response. The MCLevel23 request will need to contain the order_id of the financial transaction as well and the txn_number. Please see the appendices for a description of the Level 2 and Level 3 fields.

```
namespace USMoneris
{
        using System;
        public class TestL23Data
        {
                public static void Main(string[] args)
                {
                        /****************** REQUEST VARIABLES*******************************/

                        string host = "esplusqa.moneris.com";
                        string store_id = "monus00001";
                        string api_token = "montoken";

                        /***************** TRANSACTION VARIABLES *****************************/

                        string order_id;              //will prompt user for input
                        string txn_number;

                        Console.Write ("Please enter an order ID: ");
                        order_id = Console.ReadLine();

                        Console.Write ("Please enter a txn number: ");
                        txn_number= Console.ReadLine();

                        /******************** Addendum1  ****************************/
                        string customer_code = "ID12345";
                        string local_tax_amount = "1.00";
                        string discount_amount = "0.50";
                        string freight_amount = "0.50";
                        string duty_amount = "0.50";
                        string national_tax_amount = "0.00";
                        string other_tax_amount = "0.00";
                        string vat_invoice_ref_num = "123456789012345";
                        string customer_vat_registration_num = "1234567890123";
                        string vat_tax_amount = "1.00";
                        string vat_tax_rate = "0.00";
                        string destination_zip = "90210";
                        string ship_from_zip = "90210";

                        Addendum1 addendum1 = new Addendum1 (customer_code, local_tax_amount,
discount_amount, freight_amount,
                                duty_amount, national_tax_amount, other_tax_amount, vat_invoice_ref_num,
                                customer_vat_registration_num, vat_tax_amount, vat_tax_rate,
destination_zip, ship_from_zip);

                        /******************** Addendum2  ****************************/
                        string item_description1 = "Item1";
                        string product_code1 = "PROD00001";
                        string commodity_code1 = "IT1";
                        string quantity1 = "1.00";
                        string unit_cost1 = "1.00";
                        string ext_amount1 = "1.00";
                        string uom1 = "EA";
                        string tax_collected_ind1 = "Y";
                        string item_discount_amount1 = "0.25";
                        string item_local_tax_amount1 = "1.00";
                        string item_other_tax_amount1 = "1.00";
                        string item_other_tax_type1 = "0.00";
                        string item_other_tax_rate1 = "0.00";
                        string item_other_tax_id1 = "VAT";

                        Addendum2 addendum2 = new Addendum2 (item_description1, product_code1,
commodity_code1, quantity1,
                                unit_cost1, ext_amount1, uom1, tax_collected_ind1, item_discount_amount1,
                                item_local_tax_amount1, item_other_tax_amount1, item_other_tax_type1,
                                item_other_tax_rate1, item_other_tax_id1);

                        string item_description2 = "Item2";
                        string product_code2 = "PROD00002";
                        string commodity_code2 = "IT2";
                        string quantity2 = "2.00";
                        string unit_cost2 = "2.00";
                        string ext_amount2 = "2.00";
                        string uom2 = "EA";
                        string tax_collected_ind2 = "Y";
                        string item_discount_amount2 = "0.25";
                        string item_local_tax_amount2 = "2.00";
                        string item_other_tax_amount2 = "2.00";
                        string item_other_tax_type2 = "0.00";
                        string item_other_tax_rate2 = "0.00";
                        string item_other_tax_id2 = "VAT";

                        addendum2.AddAddendum2 (item_description2, product_code2, commodity_code2,
quantity2,
                                unit_cost2, ext_amount2, uom2, tax_collected_ind2, item_discount_amount2,
                                item_local_tax_amount2, item_other_tax_amount2, item_other_tax_type2,
                                item_other_tax_rate2, item_other_tax_id2);

                        USL23Data d = new USL23Data(order_id, txn_number, addendum1, addendum2);

                        HttpsPostRequest mpgReq = new HttpsPostRequest(host, store_id, api_token, d);

                        try
                        {
                                Receipt receipt = mpgReq.GetReceipt();

                                Console.WriteLine("CardType = " + receipt.GetCardType());
                                Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                                Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                                Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                                Console.WriteLine("TransType = " + receipt.GetTransType());
                                Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                                Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                                Console.WriteLine("ISO = " + receipt.GetISO());
                                Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                                Console.WriteLine("Message = " + receipt.GetMessage());
                                Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                                Console.WriteLine("Complete = " + receipt.GetComplete());
                                Console.WriteLine("TransDate = " + receipt.GetTransDate());
                                Console.WriteLine("TransTime = " + receipt.GetTransTime());
                                Console.WriteLine("Ticket = " + receipt.GetTicket());
                                Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                                Console.WriteLine("Corporate Card = " + receipt.GetIsCorporateCard());

                        }
                        catch (Exception e)
                        {
                                Console.WriteLine(e);
                        }
                }

        }
} //
```

L23ForcePost

The Level 2/3 ForcePost (L23ForcePost) transaction will need to contain the order_id of the financial transaction as well and the txn_number. Please refer to Appendix A. Definition of Request Fields for variable definitions

```
namespace USMoneris
{
        using System;

        public class TestL23ForcePost
        {
                public static void Main(string[] args)
                {
                        string host = "esplusqa.moneris.com";
                        string store_id = "monus00001";
                        string api_token = "montoken";

                        string order_id;
                        string cust_id = "customer1";
                        string amount = "10.00";
                        string pan = "4242424242424242";
                        string expiry_date = "1212";
                        string auth_code = "AU4R6";
                        string crypt_type = "1";

                        Console.Write ("Please enter an order ID: ");
                        order_id = Console.ReadLine();

                        /********************* Addendum1  ****************************/
                        string customer_code = "ID12345";
                        string local_tax_amount = "1.00";
                        string discount_amount = "0.50";
                        string freight_amount = "0.50";
                        string duty_amount = "0.50";
                        string national_tax_amount = "0.00";
                        string other_tax_amount = "0.00";
                        string vat_invoice_ref_num = "123456789012345";
                        string customer_vat_registration_num = "1234567890123";
                        string vat_tax_amount = "1.00";
                        string vat_tax_rate = "0.00";
                        string destination_zip = "90210";
                        string ship_from_zip = "90210";

                        Addendum1 addendum1 = new Addendum1 (customer_code, local_tax_amount, 
discount_amount, freight_amount,
                                duty_amount, national_tax_amount, other_tax_amount, vat_invoice_ref_num,
                                customer_vat_registration_num, vat_tax_amount, vat_tax_rate, 
destination_zip, ship_from_zip);

                        /******************** Addendum2  ****************************/
                        string item_description1 = "Item1";
                        string product_code1 = "PROD00001";
                        string commodity_code1 = "IT1";
                        string quantity1 = "1.00";
                        string unit_cost1 = "1.00";
                        string ext_amount1 = "1.00";
                        string uom1 = "EA";
                        string tax_collected_ind1 = "Y";
                        string item_discount_amount1 = "0.25";
                        string item_local_tax_amount1 = "1.00";
                        string item_other_tax_amount1 = "1.00";
                        string item_other_tax_type1 = "0.00";
                        string item_other_tax_rate1 = "0.00";
                        string item_other_tax_id1 = "VAT";

                        Addendum2 addendum2 = new Addendum2 (item_description1, product_code1, 
commodity_code1, quantity1,
                                unit_cost1, ext_amount1, uom1, tax_collected_ind1, item_discount_amount1,
                                item_local_tax_amount1, item_other_tax_amount1, item_other_tax_type1,
                                item_other_tax_rate1, item_other_tax_id1);

                        string item_description2 = "Item2";
```

```
                      string product_code2 = "PROD00002";
                      string commodity_code2 = "IT2";
                      string quantity2 = "2.00";
                      string unit_cost2 = "2.00";
                      string ext_amount2 = "2.00";
                      string uom2 = "EA";
                      string tax_collected_ind2 = "Y";
                      string item_discount_amount2 = "0.25";
                      string item_local_tax_amount2 = "2.00";
                      string item_other_tax_amount2 = "2.00";
                      string item_other_tax_type2 = "0.00";
                      string item_other_tax_rate2 = "0.00";
                      string item_other_tax_id2 = "VAT";

                      addendum2.AddAddendum2 (item_description2, product_code2, commodity_code2,
quantity2,
                             unit_cost2, ext_amount2, uom2, tax_collected_ind2, item_discount_amount2,
                             item_local_tax_amount2, item_other_tax_amount2, item_other_tax_type2,
                             item_other_tax_rate2, item_other_tax_id2);

                      /******************** Create Request  ***************************/


                      HttpsPostRequest mpgReq =
                             new HttpsPostRequest(host, store_id, api_token,
                                    new USL23Forcepost(order_id, cust_id, amount, pan, expiry_date,
auth_code, crypt_type, addendum1, addendum2));

                      try
                      {
                             Receipt receipt = mpgReq.GetReceipt();

                             Console.WriteLine("CardType = " + receipt.GetCardType());
                             Console.WriteLine("TransAmount = " + receipt.GetTransAmount());
                             Console.WriteLine("TxnNumber = " + receipt.GetTxnNumber());
                             Console.WriteLine("ReceiptId = " + receipt.GetReceiptId());
                             Console.WriteLine("TransType = " + receipt.GetTransType());
                             Console.WriteLine("ReferenceNum = " + receipt.GetReferenceNum());
                             Console.WriteLine("ResponseCode = " + receipt.GetResponseCode());
                             Console.WriteLine("ISO = " + receipt.GetISO());
                             Console.WriteLine("BankTotals = " + receipt.GetBankTotals());
                             Console.WriteLine("Message = " + receipt.GetMessage());
                             Console.WriteLine("AuthCode = " + receipt.GetAuthCode());
                             Console.WriteLine("Complete = " + receipt.GetComplete());
                             Console.WriteLine("TransDate = " + receipt.GetTransDate());
                             Console.WriteLine("TransTime = " + receipt.GetTransTime());
                             Console.WriteLine("Ticket = " + receipt.GetTicket());
                             Console.WriteLine("TimedOut = " + receipt.GetTimedOut());
                             Console.WriteLine("CorporateCard = " + receipt.GetIsCorporateCard());
                      }
                      catch (Exception e)
                      {
                             Console.WriteLine(e);
                      }
               }
        }
} // end TestL23IndependentRefund
```

9. What Information will I get as a Response to My Transaction Request?

For each transaction you will receive a response message. For a full description of each field please refer to

Visa and MasterCardLevel 3 Request Fields

Req	Variable Name	Field Name	Size/Type	Description
Y	product_code	Product Code	12 A/N	The product code of the individual item purchased **Mandatory, cannot contain all spaces or all zeroes.**
Y	item_description	Item Description	26 A/N – VS 35 A/N – MC	The description of the individual item purchased. **Mandatory, cannot contain all spaces or all zeroes.**
Y	quantity	Item Quantity	13 decimal	The quantity of the individual item purchased. **Mandatory, cannot contain all spaces or all zeroes.**
Y	uom	Unit of Measure	12 A/N	A three-position unit of measurement code. **Mandatory, cannot contain all spaces or all zeroes.**
Y	ext_amount	Extended item amount	9 decimal	The amount of the item that is normally calculated as price times quantity. **Mandatory, cannot contain all spaces or all zeroes. Must contain 2 decimals.**
N	unit_cost	Item unit cost	12 decimal	Individual line item cost per unit. Must contain 2 decimal places. **Cannot contain all spaces.**
Y	commodity_code	Item Commodity Code	12 A/N	Commodity Code used to classify items purchased. **Cannot contain all zeros.**
Y	item_discount_amount	Discount amount	9 decimal	Discount Amount applied to the item. Required but amount can be zeroes with 2 decimals.
N	tax_collected_ind	Tax Collected Indicator		An indicator used to reflect local sales tax captured and reported. Values are: **1** = Extended Tax amount included in total purchase amount. **0** = Extended Tax amount not included in total purchase amount **Space** = information is unknown
N	item_local_tax_amount	Local Tax Amount	9 decimal	Sales tax amount applied to an individual item. **Must contain 2 decimals.**
N	item_other_tax_amount	Other tax Amount	9 decimal	Specific tax amount collected. **Must contain 2 decimals.**
N	item_other_tax_type	Other Tax Type	4 A/N	Indicator used to further define tax categories applicable to specific domestic processing arrangements.
N	item_other_tax_rate	Other Tax Rate	5 decimal	Specific tax rate in relationship to a type of tax collected. **Must contain 2 decimals.**
N	item_other_tax_id	Other Tax ID	15 A/N	Identification number used to a specific tax amount.

Appendix E. Definitions of Response Fields.

To determine whether a transaction is successful or not the field that must be checked is Response Code. See the table below to determine the transaction result.

Response Code	Result
0 – 49 (inclusive)	Approved
50 – 999 (inclusive)	Declined
Null	Incomplete

For a full list of response codes and the associated message please refer to the Response Code document available for download at https://developer.moneris.com

10. How Do I Test My Solution?

A testing environment is available for you to connect to while you are integrating your site to our payment gateway. The test environment is generally available 7x24; however since it is a development environment we cannot guarantee 100% availability. Also, please be aware that other merchants are using the testing environment so you may see transactions and user ids that you did not create. As a courtesy to others that are testing we ask that when you are processing refunds, changing passwords and/or trying other functions that you use only the transactions/users that you created.

When using the APIs in the test environment you will need to use test store, api_token and user ids. These are different than your production ids. The ids that you can use in the test environment are in the table below.

Test IDs			
store_id	api_token	Username	Password
monusqa002	qatoken	demouser	abc1234
monusqa003	qatoken	demouser	abc1234
monusqa004	qatoken	demouser	abc1234

To To access the Merchant Resource Centre in the test environment go to https://esplusqa.moneris.com/usmpg. And use the logins provided in the previous table.

The test environment has been designed to replicate our production environment as closely as possible. One major difference is that we are unable to send test transactions onto the production authorization network and thus Issuer responses are simulated. Additionally, the requirement to emulate approval, decline and error situations dictates that we use certain transaction variables to initiate various response and error situations.

The test environment will approve and decline credit card transactions based on the penny value of the amount field.
For example, a credit card transaction made for the amount of $9.00 or $1.00 will approve since the .00 penny value is set to approve in the test environment. Transactions in the test environment should not exceed $11.00. This limit does not exist in the production environment. For a list of all current test environment responses for various penny values, please see the Test Environment Penny Response table as well as the Test Environment eFraud Response table, available at https://developer.moneris.com

NOTE These responses may change without notice. Moneris Solutions recommends you regularly refer to our website to check for possible changes.

When testing level 2/3 transactions, you can use any valid card number with any expiry date. The following test card numbers can be used as well with any expiry date.

Test Card Numbers	
Card Plan	Card Number
MasterCard	5454545454545454
Visa	4242424242424242
Amex	373599005095005 (Amex will approve on .37 and .70)
Diners	36462462742008

To access the Merchant Resource Center in the test environment go to https://esplus.moneris.com/usmpg. And use the logins provided in the previous table.

The test environment has been designed to replicate our production environment as closely as possible. One major difference is that we are unable to send test transactions onto the authorization network and thus Issuer responses are simulated. Additionally, the requirement to emulate approval, decline and error situations dictates that we use certain transaction variables to initiate various response and error situations.

11. What Do I Need to Include in the Receipt?

Visa and MasterCard expect certain variables be returned to the cardholder and presented as a receipt when a transaction is approved. These 12 fields are listed below. A sample receipt is provided in Appendix F. Sample Receipt.

1. Amount
2. Transaction Type
3. Date and Time
4. AuthCode
5. ResponseCode
6. ISO Code
7. Response Message
8. Reference Number
9. Goods and Services Order
10. Merchant Name
11. Merchant URL
12. Cardholder Name

12. How Do I Activate My Store?

Once you have received your activation letter/fax go to https://esplus.moneris.com/usmpg/activate/ as instructed in the letter/fax. You will need to input your store ID and merchant ID then click on 'Activate'. In this process you will need to create an administrator account that you will use to log into the Merchant Resource Centre to access and administer your eSELECTplus store. You will need to use the Store ID and API Token to send transactions through the API.

Once you have created your first Merchant Resource Centre user, please log on to the Interface by clicking the "eSELECTplus" button. Once you have logged in please proceed to ADMIN and then STORE SETTINGS. At the top of the page you will locate your production API Token.

13. How Do I Configure My Store For Production?

Once you have completed your testing you are ready to point your store to the production host. You will need to change the "host" to be esplus.moneris.com. You will also need to change the store_id to reflect your production store ID and well the api_token must be changed to your production token to reflect the token that you received during activation.

Once you are in production you will access the Merchant Resource Centre at https://esplus.moneris.com/usmpg. You can use the store administrator id you created during the activation process and then create additional users as needed.

For further information on how to use the Merchant Resource Centre please see the eSELECTplus Merchant Resource Centre User's Guide which is available at https://developer.moneris.com

14. How Do I Get Help?

If you require technical assistance while integrating your store, please contact the eSELECTplus Support Team:

For technical support:
Phone: 1-866-319-7450 (Technical Difficulties)

For financial support:
Phone: 1-800-471-9511
Email: eselectplus@moneris.com

For integration support:
Phone: 1-866-562-4354
Email: eselectplus@moneris.com

When sending an email support request please be sure to include your name and phone number, a clear description of the problem as well as the type of API that you are using. **For security reasons, please do not send us your API Token combined with your store ID, or your merchant number and device number in the same email.**

15. Appendix A. Definition of Request Fields

Request Fields		
Variable Name	Size/Type	Description
order_id	50 / an	Mechant defined unique transaction identifier - must be unique for every Purchase, PreAuth and Independent Refund attempt. For Refunds, Completions and Voids the order_id is must reference the original transaction.
pan	20 / variable	Credit Card Number - no spaces or dashes. Most credit card numbers today are 16 digits in length but some 13 digits are still accepted by some issuers. This field has been intentionally expanded to 20 digits in consideration for future expansion and/or potential support of private label card ranges.
expdate	4 / num	Expiry Date - format YYMM no spaces or slashes. PLEASE NOTE THAT THIS IS REVERSED FROM THE DATE DISPLAYED ON THE PHYSICAL CARD WHICH IS MMYY
amount	9 / decimal	Amount of the transaction. This must contain 3 digits with two penny values. The minimum value passed can be 0.01 and the maximum 9999999.99
crypt	1 / an	E-Commerce Indicator: 1 - Mail Order/Telephone Order - Single 2 - Mail Order/Telephone Order - Recurring 3 - Mail Order Telephone Order - Installment 4 - Mail Order Telephone Order - Unknown Classification 5 - Authenticated Ecommerce Transaction (VBV or MCSC) 6 – Non Authenicated Ecommerce Transaction (VBV or MCSC) 7 - SSL enabled merchant 8 - Non Secure Transaction (Web or Email Based) 9 - SET nonAuthenticated transaction
txn_number	255 / varchar	Used when performing follow on transactions - this must be filled with the value that was return as the trans_id in the response of the original transaction. When performaing a capture this must reference the Preauth. When performing a refund or a void this must reference the capture or the purchase.
cust_id	99/an	This is an optional field that can be sent as part of a purchase or preauth request. IT is searchable from the Moneris Merchant Interface. It is commonly used for policy number, membership number, student id or invoice number.
cavv		This is a value that is provided by the Moneris MPI or by a third party MPI. It is part of a VBV transaction.
commcard_invoice	17/an	Level 2 Invoice Number for the transaction. Used for Corporate Credit Card transactions (commercial Purchasing Cards). Characters allowed for commcard_invoice: **a-z A-Z 0-9 spaces**
commcard_tax_amount	9/decimal	Level 2 tax Amount of the transaction. Used for Corporate Credit Card transactions (Commercial Purchasing Cards). This much contain 3 digits with two penny values. The minimum value passed can be 0.00 and the maximum is 9999999.99
dynamic_descriptor	25/an	Merchant defined description sent on a per-transaction basis that will appear on the credit card statement appended to the merchant's business name. Please not, the combined length of the merchant's business name and dynamic_descriptor may not exceed 25 characters.

16. Appendix B. Definition of Visa and MasterCard Level 2/3 Request Fields

Visa and MasterCard Level 2 Request Fields

Req	Variable Name	Field Name	Size/Type	Description
Y	customer_code	Customer Code	17 A/N – VS 25 A/N – MC	A control number, such as purchase order number, project number, department allocation number or name that the purchaser supplied the merchant. **Cannot contain all spaces**.
Y	local_tax_amount	Local Tax Amount	9 N	Mandatory. The sales tax amount on the total purchase. Sales tax on Visa transactions must be between 0.1%--22% of Total transaction amount. Sales tax on MasterCard transactions must be between 0.1%--30% of Total transaction amount. **Must have 2 decimals.** i.e. 1234567.89
Y	freight_amount	Freight Amount	9 N	The total freight/shipping amount applied to the total purchase. **Must have 2 decimals. Cannot contain all zeroes if exists.**
Y	destination_zip	Ship To Zip Code	10 A/N	The ZIP code where goods will be delivered. Must be all spaces if does not exist.
Y	ship_from_zip	Ship From Zip Code	10 A/N	The ZIP code from which items were shipped. Must be all spaces if does not exist.
Y	duty_amount	Duty Amount	9 N	The total duty amount applied to the total purchase. **Must have 2 decimals. Cannot contain all zeroes if exists.**
Y	discount_amount	Discount Amount	9 N	The total discount amount applied to the total purchase. **Must have 2 decimals. Cannot contain all zeroes if exists.**
N	other_tax_amount	Alternat Tax Amount	9 N	The alternat sales tax amount. **Must have 2 decimals.**
N	national_tax_amount	National Tax Amount	9 N	The national sales tax amount. **Must have 2 decimals.**
N	customer_vat_registration_num	Customer VAT Registration Number	13 A/N	The Customer Value Added Tax (VAT) Registration Number. Must be all spaces if does not exist.
N	vat_invoice_ref_num	VAT Invoice Reference Number	15 A/N	The unique Value Added Tax (VAT) invoice number. Must be all spaces if does not exist.
Y	vat_tax_amount	Tax Amount	9 N	VAT/Tax Amount on freight or shipping only. **Must contain 2 decimal places.**
Y	vat_tax_rate	Tax Rate	4 N	VAT/Tax rate applied to freight or shipping only. **Must contain 2 decimal places.**

Visa and MasterCardLevel 3 Request Fields

Req	Variable Name	Field Name	Size/Type	Description
Y	product_code	Product Code	12 A/N	The product code of the individual item purchased **Mandatory, cannot contain all spaces or all zeroes.**
Y	item_description	Item Description	26 A/N – VS 35 A/N – MC	The description of the individual item purchased. **Mandatory, cannot contain all spaces or all zeroes.**
Y	quantity	Item Quantity	13 decimal	The quantity of the individual item purchased. **Mandatory, cannot contain all spaces or all zeroes.**
Y	uom	Unit of Measure	12 A/N	A three-position unit of measurement code. **Mandatory, cannot contain all spaces or all zeroes.**
Y	ext_amount	Extended item amount	9 decimal	The amount of the item that is normally calculated as price times quantity. **Mandatory, cannot contain all spaces or all zeroes. Must contain 2 decimals.**
N	unit_cost	Item unit cost	12 decimal	Individual line item cost per unit. Must contain 2 decimal places. **Cannot contain all spaces.**
Y	commodity_code	Item Commodity Code	12 A/N	Commodity Code used to classify items purchased. **Cannot contain all zeros.**
Y	item_discount_amount	Discount amount	9 decimal	Discount Amount applied to the item. Required but amount can be zeroes with 2 decimals.
N	tax_collected_ind	Tax Collected Indicator		An indicator used to reflect local sales tax captured and reported. Values are: **1** = Extended Tax amount included in total purchase amount. **0** = Extended Tax amount not included in total purchase amount **Space** = information is unknown
N	item_local_tax_amount	Local Tax Amount	9 decimal	Sales tax amount applied to an individual item. **Must contain 2 decimals.**
N	item_other_tax_amount	Other tax Amount	9 decimal	Specific tax amount collected. **Must contain 2 decimals.**
N	item_other_tax_type	Other Tax Type	4 A/N	Indicator used to further define tax categories applicable to specific domestic processing arrangements.
N	item_other_tax_rate	Other Tax Rate	5 decimal	Specific tax rate in relationship to a type of tax collected. **Must contain 2 decimals.**
N	item_other_tax_id	Other Tax ID	15 A/N	Identification number used to a specific tax amount.

17. Appendix E. Definitions of Response Fields

Response Fields		
Variable Name	Size/Type	Description
ReceiptId	50 / an	order_id specified in request
ReferenceNum	18 / num	The reference number is an 18 character *string* that references the terminal used to process the transaction as well as the shift, batch and sequence number, This data is typically used to reference transactions on the host systems and must be displayed on any receipt presented to the customer. This information should be stored by the merchant. The following illustrates the breakdown of this field where "640123450010690030" is the reference number returned in the message, "64012345" is the terminal id, "001" is the shift number, "069" is the batch number and "003" is the transaction number within the batch. Moneris Host Transaction identifier
ResponseCode	3 / num	Transaction Response Code < 50: Transaction approved >= 50: Transaction declined NULL: Transaction was not sent for authorization * If you would like further details on the response codes that are returned please see the Response Codes document available for download at https://developer.moneris.com.
ISO	2 / num	ISO response code
AuthCode	8 / an	Authorization code returned from the issuing institution
TransTime	##:##:##	Processing host time stamp
TransDate	yyyy-mm-dd	Processing host date stamp
TransType	an	Type of transaction that was performed
Complete	True/False	Transaction was sent to authorization host and a response was received
Message	100 / an	Response description returned from issuing institution.
TransAmount	9 / decimal	amount specified in request.
CardType	2 / alpha	Credit Card Type
Txn_number	20 / an	Gateway Transaction identifier
TimedOut	True/False	Transaction failed due to a process timing out
Ticket	n/a	reserved
CorporateCard	True/False/Null	Will return true if the card is a Corporate Card – this card is capable of accepting the subsequent Level 2/3 data transaction types. If it is false the card is not corporate, if the result is Null then you are not enrolled in Level2/3
MessageId	15 num / Null	This value is returned with a Visa Level 2/3 corporate card transaction. It is a unique transaction identifier.

18. Appendix F. Sample Receipt

Your order has been Approved
Print this receipt for your records

QA Merchant #1
3250 Bloor St West
Toronto Ontario
M8X2X9

1 800 987 1234
www.moneris.com

Transaction Type: Purchase

Order ID:	mhp3495435587		
Date/Time:	2002-10-18 11:27:48	Approval Code:	030012
Sequence Number:	660021630012090020	Response / ISO Code:	028/04
Amount:	12.04	APPROVED * =	

Item	Description	Qty	Amount	Subtotal
cir-001	Med Circle	1	2.01	2.01
tri-002	Big triangle	1	1.01	1.01
squ-003	small square	2	1.01	3.02
			Shipping:	4.00
			GST :	1.00
			PST :	1.00
			Total:	12.04 USD

Bill To:	Ship To:
Test Customer	Test
123 Main St	1 King St
Springfield	Bakersville
ON	ON
Canada	Canda
M1M 1M1	M1M 1M1
tel: 416 555 1111	tel: 416 555 2222
fax: 416 555 1111	fax: 416 555 2222

Special Instructions
Knock on Back door when delivering
E-Mail Address:eselectsupport@moneris.com
Refund Policy
30 Days - Must be unopened, 10% restocking charge.