ELECTRONIC REVENUE COLLECTION FOR COMMONWEALTH AGENCIES

SnapPay Integration Guide

V 1.4

Table of Contents

1 Version Control

Name	Description	Version	Date
SnapPay Integration Team / Office of Budget	Original	1.0	6/13/2023
SnapPay Integration Team / Office of Budget	Merchant ID for staging environment for credit card no fee to customer setup (charge to agency)	1.1	10/13/2023
SnapPay Integration Team / Office of Budget	Step 14 to provide link to OB web page staging environment	1.2	12/1/2023
SnapPay Integration Team / Office of Budget	Remove email from HMAC signature, changed HPP screenshots to reflect the latest HPP and clarification items for Use Case #2	1.3	4/5/2024
SnapPay Integration Team / Office of Budget	GetTransaction API usage	1.4	7/11/2024

2 Strategy

2.1 Vision

Align with the Governor's initiative to maximize and simplify customer experience by implementing an enterprise-wide solution for remitting electronic payments.

2.2 Goals and Objectives

- Maximize the potential for revenue collections and reduce manual processing efforts, ultimately reducing the amount of time it takes for revenues to reach Treasury investment pools.
- Promote flexibility in the ways the Commonwealth can collect receivables and revenues through Credit Card, Automated Clearing House (ACH), or Check.
- Simplify system support by maintaining a common payment gateway for electronic payment solution.
- Leverage the investment in Commonwealth's Enterprise Wide Financial and Reporting System (SAP ERP) and its integration with Treasury.
- Reduce the need for Commonwealth staff necessary to process and deposit paper checks.

3 Description

Office of Budget (OB), in association with Office of Administration (OA) / Integrated Enterprise Systems (IES) and the Commonwealth's merchant services contractor, Fiserv, hosts an enterprise-wide solution for accepting electronic payments into the Commonwealth's financial system (SAP) to eliminate manual effort in processing paper checks by the Office of Comptroller Operations (OCO) and other state agencies receiving incoming revenues from external customers. The solution is called SnapPay, a product of Fiserv, offering out of the box integration with SAP and thus integration with Treasury.

4 Payments against SAP Accounts Receivables

Any customer, established in the SAP system, can log in to the OB Payment Center (SnapPay) to review their outstanding invoices and remit payments (Accounts Receivables for the Commonwealth) via Automated Clearing House (ACH) or Credit Card.

5 Payments outside of SAP Accounts Receivables

The OB Payment Center (SnapPay) also offers a hosted payment page solution which commonwealth agencies can leverage for collecting payments from customers who may not be established in SAP.

Broadly, there are two different use cases:

5.1 Use Case #1

In Use Case #1, agencies need to collect revenue or refunds of expenditures from customers without any additional requirement to integrate with their internal application.

Agencies can provide their customers guidance on payment center use and direct their customers to the OCO website, https://www.bpp.ob.pa.gov/Customer/PaymentForm, where customers are able to make payments against the specific agency program they are interested in.

5.2 Use Case #2

In Use Case #2, agencies need to collect revenue or refunds of expenditures **and have a requirement to connect to a payment gateway from within their agency application**.

In this case, the agency application already has the customer details and requires a direct connection to the payment gateway from within the agency application, or there is a need to capture such payment receipts within the agency application immediately.

6 Integration Setup

This document provides detailed guidelines for both Use Cases previously described and is to be read in conjunction with the "Snap Pay HPP Integration" document, which provides technical details behind the integration.

The following steps are involved for agencies to set up the integration with the SnapPay Payment Gateway. **Please note:** Only steps 1-4 are required for Use Case #1:

1. Agencies interested in the solution should email RA-OBCO_PYMTCENTER@pa.gov with a brief description of their program.
2. The Office of Budget (OB) will correspond with the agency to understand the program requirements.
3. **The agency provides the details of the revenue or refund of expenditure program in the Phase 2 Coding Template.xlsx**. Such information includes the program description, SAP Account codes like GL Account, Cost Center, Internal Order and Fund. Agencies will decide if credit card fees will be paid by the customer or borne by the agency. ACH payments do not incur any fees.
4. OB will establish the revenue program(s) in the SAP production system (Use Case #1) or staging system (Use Case #2) and provide the unique program ID(s) to the agency.
5. Agencies interested in integration (Use Case #2), will review the technical information/steps included throughout the remainder of this document. Interested agencies will contact the OA resource account at RA-OA_SNAPPAY@pa.gov and copy RA-OBCO_PYMTCENTER@pa.gov. The connection details including the API Authorization Code, AccountID, CustomerID, MerchantID, etc., for the staging system are provided in the Appendix of this document, which will be required to achieve the integration. The access for production will be provided once successful testing is complete and when the agency is ready to go-live in the production environment. **Agencies are strongly encouraged to complete the above steps before they begin integration.**
6. Agencies will work with their IT delivery centers to integrate their application to invoke the hosted payment page, based on the SnapPay integration guide. Questions regarding the integration can be addressed to the OA resource account at RA-OA_SNAPPAY@pa.gov and copy RA-OBCO_PYMTCENTER@pa.gov. OA will reach out to SnapPay if the issue cannot be resolved internally.
7. Agencies will complete testing in the staging environment. The results will be reviewed with the Office of Administration and Office of Budget and the implementation schedule for the production environment will then be finalized. OB will establish the revenue program(s) in the SAP production system (Use Case #2).
8. The agency will pass customer details like Name, Address, Customer Email Address, and Total Amount, along with the individual program ID/payment amount, to the hosted payment page from within their application and invoke the hosted payment page.

9. The SnapPay hosted payment page has ten (10) user defined fields (UDF), which have been proposed, as detailed below, by the Commonwealth. The definition of these UDF fields therefore cannot be changed.

User Defined Fields on Hosted Page	Data Element	Mandatory or Optional
UDF2	Program ID 1	Mandatory
UDF3	Amount 1	Mandatory
UDF4	Payment Reference Text 1	Optional*
UDF5	Program ID 2	Optional
UDF6	Amount 2	Optional
UDF7	Payment Reference Text 2	Optional
UDF8	Program ID 3	Optional
UDF9	Amount 3	Optional
UDF0	Payment Reference Text 3	Optional
UDF1	SAP Vendor	Optional; Required only when there is a refund of expenditure from vendor

* To successfully use the GetTransaction API functionality described in item 12 below, the following must be implemented by the agency:

- While UDF4 is configured in the system as an optional field, it is expected that for Use Case #2, agencies always pass a unique reference that identifies the specific transaction. UDF4 is used as unique transaction/order id to identify and fetch details of a transaction. This is a 20-character field, and the desired nomenclature is as below:
  - Characters 1-3: The program id (same as UDF 2)
  - Character 4: Hyphen (-)
  - Character 5-14: A running sequence number, as desired by the agency
  - Character 15-20: transaction date in YYMMDD format
  - Ex: 289-PA1234567824071l
  - If UDF7 and UDF0 are used, they can use similar conventions.

10. Agency customers have the option to remit payments for up to three (3) different agency programs (program ID) within a single payment transaction. Agency applications will ensure the total amount is the sum of the individual payment for all three (3) programs the customer chooses to pay for on the single transaction.
11. The customer enters the ACH or credit card information on the hosted payment page. Please see the screenshots in the next section for reference.

12. Upon submitting the payment on the hosted payment page, the page returns the success or failure of the payment to the agency application. Agencies can capture the response information for further use, as required.

    A **GetTransaction API** is available, and recommended, as a callback API to verify the success or failure of the transaction, in case the agency finds the user attempting to make a second payment, or if the agency application has not received a response after a lapse of time. This callback would help the agency application understand if the original payment submission occurred successfully in SnapPay, to avoid duplicate payments and to ensure the agency application is in sync for all transactions with SnapPay. For this callback API, the agency application must pass an Order ID/Unique reference against the transaction, which will be the UDF4.

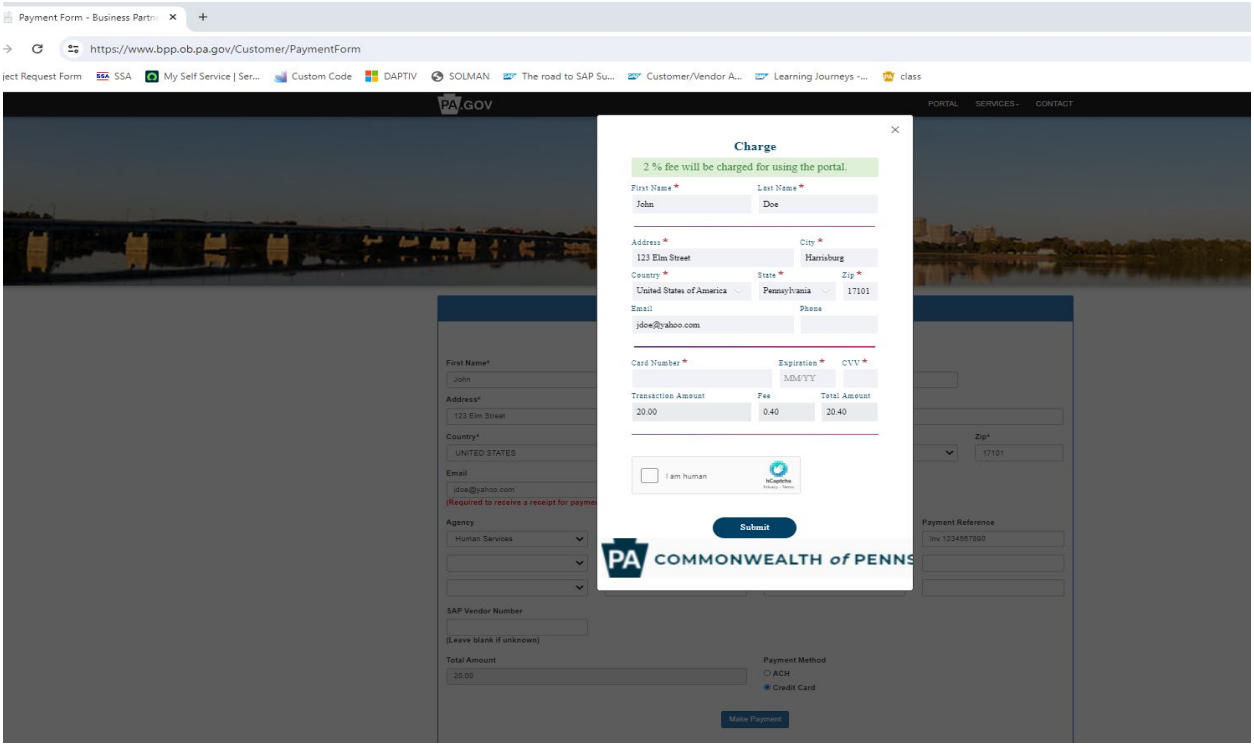
    The sample coding for the GetTransaction API is provided in the Appendix of this guide.

13. Agencies also have access to financial reports within SAP to review revenue receipts for any given period.
14. Agencies intending integration with their website may find it useful to go through the Office of Budget application in the staging environment at https://www.bpp.ob.beta.pa.gov/Customer/PaymentForm. This application uses the Hosted Payment Page solution internally.
15. Agencies requiring a solution/integration/API outside the above laid out process, can email RA-OA_SNAPPAY@pa.gov and RA-OBCO_PYMTCENTER@pa.gov. This will be evaluated on a case-by-case basis with Fiserv for feasibility.

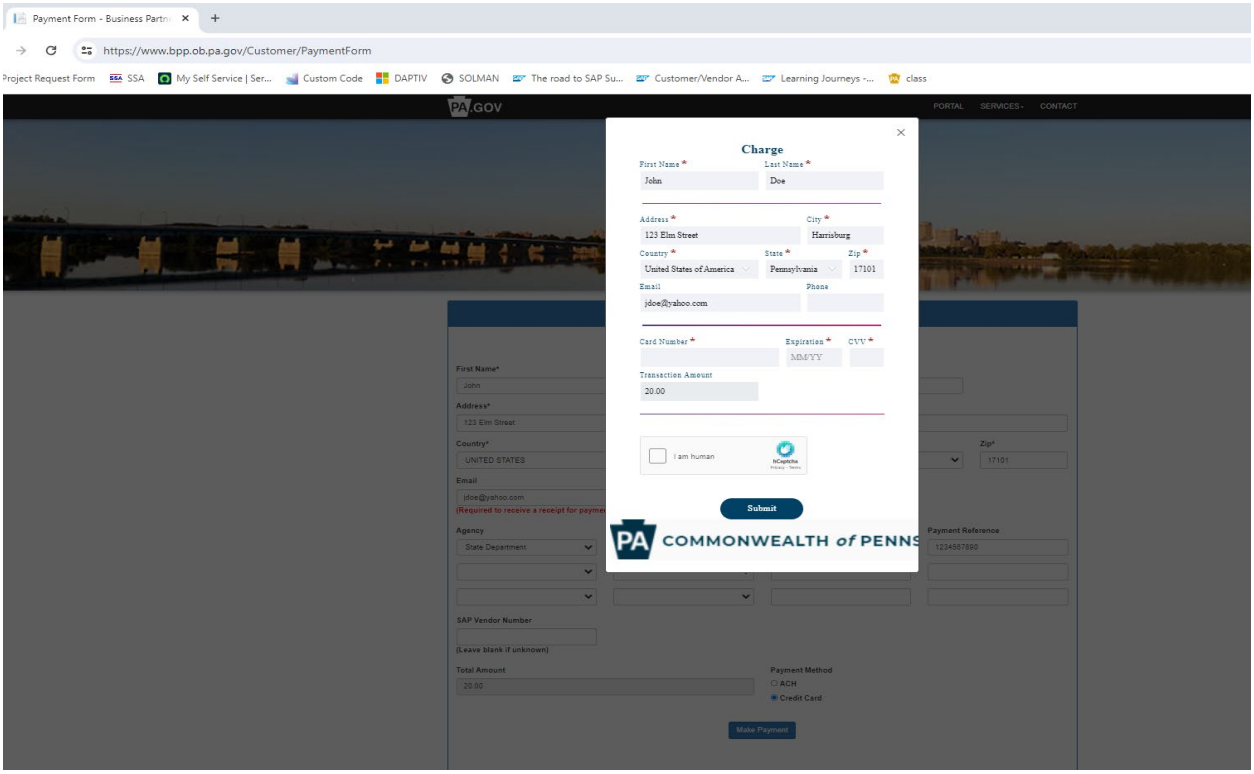
After the solution is implemented, agencies will be responsible for addressing customer queries for payments that originate from the agency application.

7 Hosted Payment Page - Screenshots

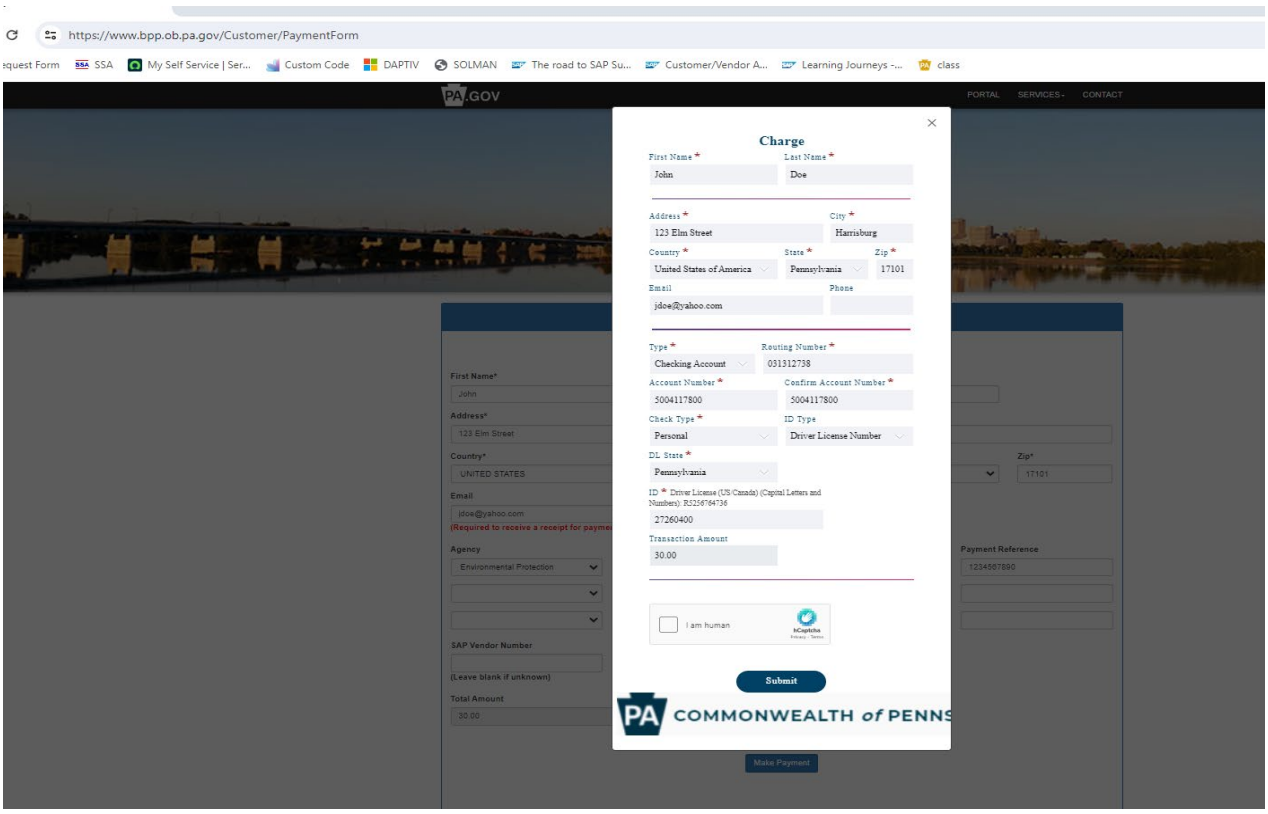
7.1 Payment by Card (CC Fee borne by Customer)



7.2 Payment by Credit Card (No charge to customer, borne by agency)



7.3 Payment by ACH



8 Appendix

8.1 Supporting Documents

1. SnapPay Integration Guide
2. Phase 2 Coding Template.xlsx

8.2 Integration Parameters

Attribute	Staging	Production
URL API for SnapPay	https://stage.snappayglobal.com/Interop/HostedPaymentPage	Will be provided
API AuthCode	XkZnJhylhOc7KQsBvPLnpbWMRR5mgLr7BFxVNwwpg6I=	Will be provided
Merchant ID- Credit Card (Charge to Customer)	700000009421	Will be provided
Merchant ID – Credit Card (Charge to Agency)	700000009422	Will be provided
Merchant ID – ACH	44010399	Will be provided
Account ID	1001111488	Will be provided
Customer ID	999	Will be provided

8.3 Test Credit Cards and ACH Account Information

8.3.1 Test Card Details

Credit Card details for testing purposes are detailed in the chart below.

Please note: Use any future date (MM/YY) as the credit card expiration date while testing.

Card Number	Type
4111111111111111	Visa
4637090000158588	Visa
5399104611689124	MasterCard
5454545454545454	MasterCard
6011000991300009	Discover
371449635398431	AMEX

8.3.2 Test ACH Details

Automated Clearing House (ACH) details for testing purposes are detailed in the chart below:

Routing Number	Account Number
071904779	123456789

8.4 Code Snippets

This section provides sample code which agencies can reference to assist with the integration of the Hosted Payment Page:

Please note: If you have a previous version of the user guide, you will notice that the email has been removed from the HMAC signature creation in this document. The email address is not always required by an agency, so it has been removed from the HMAC signature creation.

8.4.1 ASP.NET MVC – C# Solution

Payment page .cshtml file code:

On the submit button – getSignature():

```
<script type="text/javascript">
   function loadScript() {
      var state;
      var s = document.getElementById('ddlState').value;
      var fs = document.getElementById('fstate').value;
      var ctry = document.getElementById('ddlCountry').value;
      if (ctry == "US") {
         state = s;
      }
      else {
         state = fs
      }
      var paytype = paymentType();
      var merchid;
      if (paytype == "ACH") {
         //staging
         //merchid = "44010399";
      }
      else {
         //staging
         //merchid = "700000009421";
         }
      var script = document.createElement('script');
      script.setAttribute('id', 'snappay_hppform');
      //staging
      //script.setAttribute('src', 'https://stage.snappayglobal.com/Areas/Interop/Scripts/HPPForm.js');
       script.setAttribute('data-target', '#snappayhppform_response');
      script.setAttribute('data-callback', 'submit_external_ecommerce'); //response information
      //staging
```

```
        //script.setAttribute('data-accountid', '1001111488');
        script.setAttribute('data-customerid', '999');
        script.setAttribute('data-currencycode', 'USD');
        script.setAttribute('data-transactionamount', document.getElementById('totalamt').value);
        script.setAttribute('data-merchantid', merchid);
        script.setAttribute('data-paymentmode', paytype);
        script.setAttribute('data-cvvrequired', 'Y');
        script.setAttribute('data-enableemailreceipt', 'Y');
        //staging
script.setAttribute('data-redirectionurl', 'https://www.bpp.ob.betapa.gov/Customer/PaymentRedirect');
        //staging
 script.setAttribute('data-snappayurl', 'https://stage.snappayglobal.com/Interop/HostedPaymentPage');
        script.setAttribute('data-udf0', document.getElementById('paymentref3').value);
        script.setAttribute('data-udf1', document.getElementById('sapvendorno').value);
        script.setAttribute('data-udf2', document.getElementById('ddlProgs').value);
        script.setAttribute('data-udf3', document.getElementById('amount1').value);
        script.setAttribute('data-udf4', document.getElementById('paymentref1').value);
        script.setAttribute('data-udf5', document.getElementById('ddlProgs2').value);
        script.setAttribute('data-udf6', document.getElementById('amount2').value);
        script.setAttribute('data-udf7', document.getElementById('paymentref2').value);
        script.setAttribute('data-udf8', document.getElementById('ddlProgs3').value);
        script.setAttribute('data-udf9', document.getElementById('amount3').value);
        script.setAttribute('data-firstname', document.getElementById('fname').value);
        script.setAttribute('data-lastname', document.getElementById('lname').value);
        script.setAttribute('data-addressline1', document.getElementById('address').value);
        script.setAttribute('data-city', document.getElementById('city').value);
        script.setAttribute('data-state', state);
        script.setAttribute('data-zip', document.getElementById('zip').value);
        script.setAttribute('data-country', document.getElementById('ddlCountry').value);
        script.setAttribute('data-email', document.getElementById('email').value);
        script.setAttribute('data-phonenumber', '');
        script.setAttribute('data-signature', document.getElementById('GUID').value);

        document.body.appendChild(script);
    }
</script>

<script type="text/javascript">
    function submit_external_ecommerce() {
        $.ajax({
            type: 'POST',
            url: "/Customer/SnapPayResponseSuccess",
            datatype: "json",
```

```
        data: {
            resp: $('#snappayhppform_response').val(), ba1: $('#ddlBA').val(), ba2: $('#ddlBA2').val(), ba3: $('#ddlBA3').val(), ta: $('#totalamt').val(),
            pg1: $('#ddlProgs').val(), pg2: $('#ddlProgs2').val(), pg3: $('#ddlProgs3').val(), pr1: $('#paymentref1').val(), pr2: $('#paymentref2').val(),
            pr3: $('#paymentref3').val(), vn: $('#sapvendorno').val(), am1: $('#amount1').val(), am2: $('#amount2').val(), am3: $('#amount3').val()
        },
        traditional: true,
        success: function (response) {
            if (response == "Transaction successful.") {
                //staging
                var redirectionurl = "https://www.bpp.ob.beta.pa.gov/Customer/PaymentRedirect";
                window.location.href = redirectionurl;
            }
            else {
                //staging
                var redirectionurl = "https://www.bpp.ob.beta.pa.gov/Customer/PaymentForm";
                window.location.href = redirectionurl;
            }
        }
    })
  }
</script>

<script type="text/javascript">
  function getSignature() {
    var ptype = paymentType();
    var merchant;
    var signature;
    if (ptype == "ACH") {
        //staging
        //merchid = "44010399";
    }
    else {
        //staging
        //merchid = "700000009421";
    }
    $.ajax({
        type: 'GET',
        url: "/Customer/CreateHMAC",
        datatype: "json",
        data: { merchant: merchant, amount: $('#totalamt').val(), paymethod: ptype },
```

```
            success: function (response) {
                $("#GUID").val(response);
loadScript();
            },
            error: function (ex) {
                alert('Failed to retrieve HMAC value.' + ex);
            }
        });
        return false;
    }
</script>
```

8.4.2 Controller C# code

Create the HMAC signature and return to the view (.cshtml)

```
[HttpGet]
public ActionResult CreateHMAC(string merchant, string amount, string paymethod)
{
  string apiAuthCode = “XkZnJhylhOc7KQsBvPLnpbWMRR5mgLr7BFxVNwwpg6I= “;
  //staging
  //string accountid = "1001111488";
  string customerid = "999";
  string merchantid = merchant;
  string transactionamount = amount;
  string currencycode = "USD";
  string paymentmode = paymethod; // or "ACH"
  string nonce = Guid.NewGuid().ToString("N");
  DateTime epochStart = new DateTime(1970, 01, 01, 0, 0, 0, 0, DateTimeKind.Utc);
  TimeSpan timeSpan = DateTime.UtcNow - epochStart;

  string requestTimeStamp = Convert.ToUInt64(timeSpan.TotalSeconds).ToString();
  string signatureRawData = accountid + customerid + merchantid + transactionamount + currencycode + paymentmode + nonce + requestTimeStamp;
  var secretKeyByteArray = Convert.FromBase64String(apiAuthCode);
  byte[] signature = Encoding.UTF8.GetBytes(signatureRawData);
  string requestSignatureBase64String = string.Empty;
  using (HMACSHA256 hmac = new HMACSHA256(secretKeyByteArray))
  {
    //convert encoded utf-8 signature to hash
    byte[] signatureBytes = hmac.ComputeHash(signature);
    //convert hash signature to base 64 string
    requestSignatureBase64String = Convert.ToBase64String(signatureBytes);
  }
  string signatureData = string.Format("{0}:{1}:{2}", requestSignatureBase64String, nonce, requestTimeStamp);
  string HmacValue = Convert.ToBase64String(Encoding.UTF8.GetBytes(signatureData));

  return Json(HmacValue, JsonRequestBehavior.AllowGet);
}
```

Return success response to the view:

```
[HttpPost]
```

```
public JsonResult SnapPayResponseSuccess(string resp, string ba1, string ba2, string ba3, string ta, string pg1, string pg2, string pg3, string pr1, string pr2, string pr3, string vn, string am1, string am2, string am3)
    {
        string busarea1 = null;
        string busarea2 = null;
        string busarea3 = null;
        string totamt = null;
        string responseSP = null;
        string prg1 = null;
        string prg2 = null;
        string prg3 = null;
        string amt1 = null;
        string amt2 = null;
        string amt3 = null;
        string pref1 = null;
        string pref2 = null;
        string pref3 = null;
        string vendno = null;
        responseSP = resp;
        busarea1 = ba1;
        busarea2 = ba2;
        busarea3 = ba3;
        totamt = ta;
        prg1 = pg1;
        prg2 = pg2;
        prg3 = pg3;
        amt1 = am1;
        amt2 = am2;
        amt3 = am3;
        pref1 = pr1;
        pref2 = pr2;
        pref3 = pr3;
        vendno = vn;

        var strrespSP = responseSP;

        var charsToRemove = new string[] { @"\", "\"", "{", "}", "null" };

        foreach (var c in charsToRemove)
        {
            strrespSP = strrespSP.Replace(c, string.Empty);
        }
```

```
    string[] arr = strrespSP.Split(',');
    char[] chDelimiter = { ':' };

    string rtnmsg = arr[7];

    string[] arrrtnmsg = rtnmsg.Split(':');

    var rtnm = arrrtnmsg[1];

    Session["Message"] = rtnm;
    Session["Response"] = responseSP;
    Session["BusArea1"] = busarea1;
    Session["BusArea2"] = busarea2;
    Session["BusArea3"] = busarea3;
    Session["TotalAmt"] = totamt;
    Session["Prg1"] = prg1;
    Session["Prg2"] = prg2;
    Session["Prg3"] = prg3;
    Session["Amt1"] = amt1;
    Session["Amt2"] = amt2;
    Session["Amt3"] = amt3;
    Session["PayRef1"] = pref1;
    Session["PayRef2"] = pref2;
    Session["PayRef3"] = pref3;
    Session["VendorNo"] = vendno;

    return Json(rtnm, JsonRequestBehavior.AllowGet);
}
```

8.4.3 GetTransaction API

The detailed documentation for the callback API (GetTransaction) is available through this path:

https://developer.fiserv.com/product/SnapPay/api/?type=post&path=/api/interop/GetTransaction&branch=main&version=3.0.0

Please note: The credentials for the API will be provided on request.

```
using Newtonsoft.Json;
using System;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Web.Mvc;

namespace SnapPayTestMVCApplication.Controllers
{
   public class HomeController : Controller
   {
      public ActionResult Index()
      {
         return View();
      }

      [HttpPost]
      public async Task<ActionResult> GetTransAsync()
      {
         //Staging values
         string apiUserId = "1001111488";
         string apiPassword = "";
         string apiAuthCode = "XkZnJhylhOc7KQsBvPLnpbWMRR5mgLr7BFxVNwwpg6I=";
         string data = "{\"accountid\":1001111488,\"orderid\":\"288-PA2345\"}";
         string requestUri = "https://restapi-stage.snappayglobal.com/api/Interop/GetTransaction";
         string accountid = "1001111488";

         string requestHttpMethod = "POST";

         string requestContentBase64String = string.Empty;

         DateTime epochStart = new DateTime(1970, 01, 01, 0, 0, 0, 0, DateTimeKind.Utc);
         TimeSpan timeSpan = DateTime.UtcNow - epochStart;
         string requestTimeStamp = Convert.ToUInt64(timeSpan.TotalSeconds).ToString();
         string nonce = Guid.NewGuid().ToString("N");

         //Checking if the request contains body, usually will be null with HTTP GET and DELETE
         HttpRequestMessage req = new HttpRequestMessage(HttpMethod.Post, requestUri);
```

```
        req.Content = new StringContent(data, Encoding.UTF8, "application/json");
        if (req.Content != null)
        {
           byte[] content = req.Content.ReadAsByteArrayAsync().Result;
           MD5 md5 = MD5.Create();
           //Hashing the request body, any change in request body will result in different hash
           byte[] requestContentHash = md5.ComputeHash(content);
           requestContentBase64String = Convert.ToBase64String(requestContentHash);
        }

        string signatureRawData = accountid + requestHttpMethod + requestUri + requestTimeStamp +
nonce + requestContentBase64String;
        var secretKeyByteArray = Convert.FromBase64String(apiAuthCode);
        byte[] signature = Encoding.UTF8.GetBytes(signatureRawData);
        string requestSignatureBase64String = string.Empty;

        using (HMACSHA256 hmac = new HMACSHA256(secretKeyByteArray))
        {
           //convert encoded utf-8 signature to hash
           byte[] signatureBytes = hmac.ComputeHash(signature);
           //convert hash signature to base 64 string
           requestSignatureBase64String = Convert.ToBase64String(signatureBytes);
        }

        string signatureData = string.Format("{0}:{1}:{2}:{3}", accountid, requestSignatureBase64String,
nonce, requestTimeStamp);
        string HmacValue = Convert.ToBase64String(Encoding.UTF8.GetBytes(signatureData));
        string credentials = Convert.ToBase64String(ASCIIEncoding.ASCII.GetBytes(apiUserId + " : " +
        apiPassword));

        //Setting HMAC signature in the header with prefix Hmac.
        req.Headers.Add("Signature", "Hmac " + HmacValue);
        //Setting other values in the header.
        req.Headers.Add("Authorization", "Basic " + credentials);
        req.Headers.Add("AccountId", accountid);

        var client = new HttpClient();
        string resptransact = string.Empty;
        using (var response = await client.SendAsync(req))
        {
           response.EnsureSuccessStatusCode();
           var body = await response.Content.ReadAsStringAsync();
           resptransact = body;
           Console.WriteLine(body);
        };

        return Json(resptransact, JsonRequestBehavior.AllowGet);
```

```
      }
   }
}
```

//Response from SnapPay//

//{"accountid":1001111488,"message":"Success","status":"Y","transactions":[{"transaction":{"transactionstatus":"Success","merchantid":"700000009421","returncode":"000","returndescription":"Approval","pgtransactionid":"720665263271","pgtransactiontype":"Charge","paymenttransactionid":"4513265","authorizationcode":"PPS652","transactiondate":"8/7/2024 05:34:25 PM","procrespcode":"RPCT","transactionamount":21,"currency":"USD","totaltransactionamount":21.42,"feeamount":0.42,"feeauthcode":"PPS650","feeformat":"Percentage","feevalue":2,"feestatus":"Y","feetransactionid":"4513266","feetype":"Service"},"paymentmethod":{"paymentmethodid":599553,"type":"VISA","last4":"1111","tokenid":"9415818374071111","firstname":"Sudhakar","lastname":"Venkataraman","expirationmonth":"8","expirationyear":"2026","addressline1":"112 Joe Dr","city":"Harrisburg","state":"PA","zip":"17101","country":"US","phonenumber":"","email":"svenkatara@pa.gov"}}]}