SnapPay HPP Integration

Updated: 7/12/22

Table of Contents

Overview

SnapPay provides fast, easy and reliable credit card processing services for online payments with and without managed service fee (MSF). This document gives the detailed information required to integrate with SnapPay payment processing solutions through hosted payment page. This document also provides detailed information on other APIs that are available.

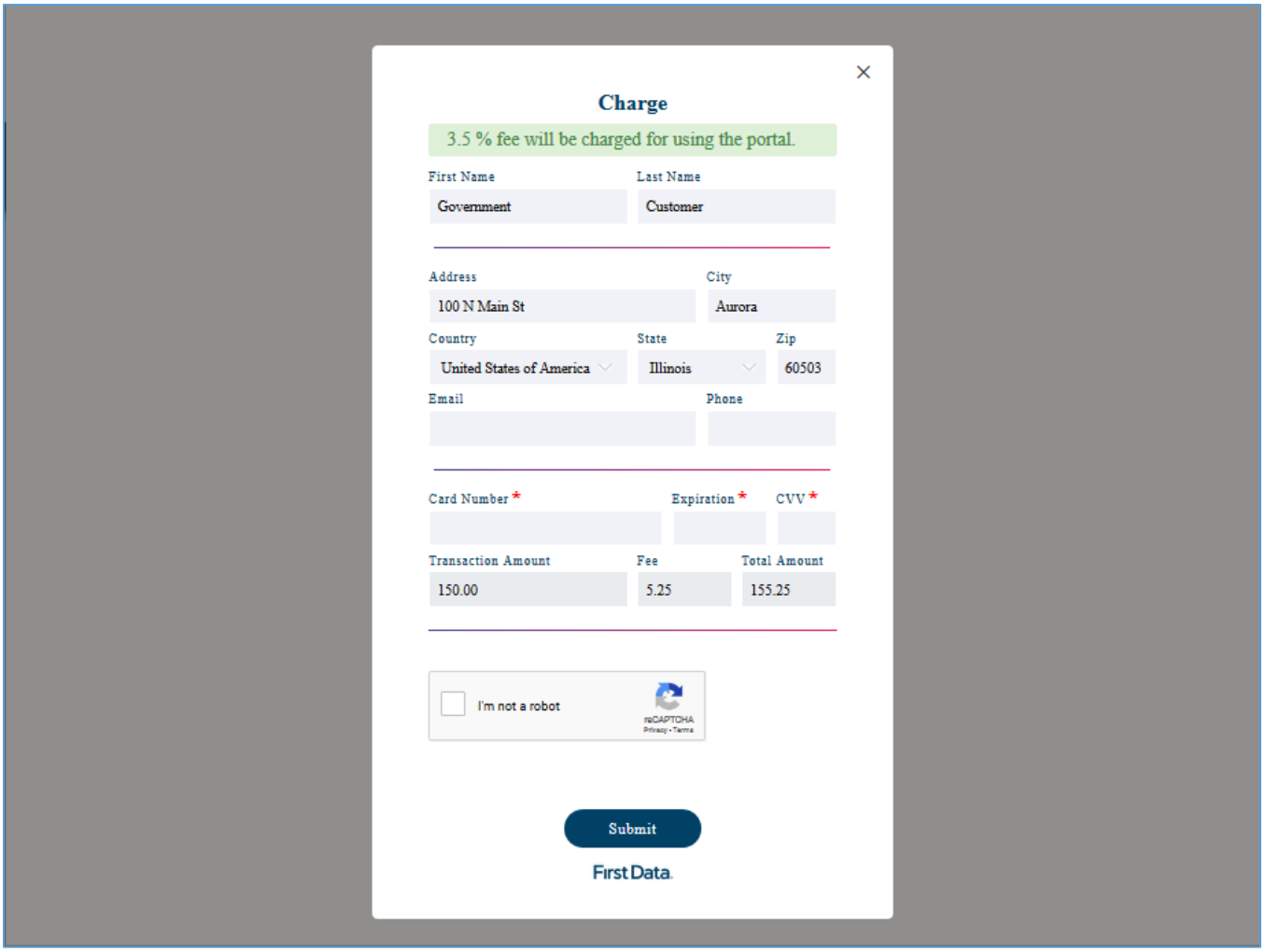
Hosted Payment Page using Form Post Parameters

The SnapPay Hosted Payment Page or SnapPay HPP is the payment solution for an online application to implement a simple, customizable payment page that fits their need. SnapPay HPP requires little developer involvement, so setup is quick and simple.

Merchant will post the request with the parameters listed below to the URL provided in the Post URLs section. User will be redirected to the SnapPay hosted payment page where the user will submit the payment with a credit card details. Upon clicking the submit button, SnapPay will process the payment and post the result parameters back to the calling system.

Post URLs

Environment	**URL**
Staging	https://stage.SnapPayGlobal.com/Interop/HostedPaymentPage
Production	https://www.SnapPayGlobal.com/Interop/HostedPaymentPage



Hosted Payment Page with Credit Card

Request Parameters

Parameter	Data Type	Size	Reqd.	Notes
accountid	numeric	10	Y	SnapPay Account ID
customerid	string	20	Y	Customer ID
currencycode	string	3	Y	Currency Code – USD
transactionamount	double		Y	Amount with 2 decimals
merchantid	string	20	Y	Merchant ID
paymentmode	string	3	Y	CC – for credit card ACH – for bank details

cvvrequired	string	1	N	Y/N – determines if verification code is required on the form
enableemailreceipt	string	1	N	Y/N – determines if emailing receipt is enabled after transaction is completed. Email should be provided on the form for emailing receipt.
redirectionurl	string	256	Y	Redirection URL – SnapPay will redirect the user to the URL specified here after the transaction is successful.
signature	string		Y	HMAC Signature
firstname	string	40	N	First Name
lastname	string	40	N	Last Name
addressline1	string	40	N	Address Line 1 (Street Address)
city	string	100	N	City
state	string	100	N	State
zip	string	20	N	Zip/Postal Code
country	string	100	N	Country
email	string	100	N	Email
phonenumber	string	26	N	Phone Number
udf0	string	20	N	User Defined Fields
udf1	string	20	N	User Defined Fields
udf2	string	20	N	User Defined Fields
udf3	string	20	N	User Defined Fields
udf4	string	20	N	User Defined Fields
udf5	string	20	N	User Defined Fields
udf6	string	20	N	User Defined Fields
udf7	string	20	N	User Defined Fields
udf8	string	20	N	User Defined Fields
udf9	string	20	N	User Defined Fields

Response Parameters

Parameter	Data Type	Size	Notes
accountid	numeric	10	SnapPay Account ID
customerid	string	20	Customer ID
companycode	string	20	Company Code
currencycode	string	3	Currency Code – USD
merchantid	string	20	Merchant ID
paymenttransactionid	numeric	10	SnapPay Transaction ID
pgtransactionid	string	80	Gateway Transaction ID for reference
transactionamount	double		Amount with 2 decimals
authorizationnumber	string	10	Authorization Number from gateway
returnmessage	string	500	Return message from SnapPay. Validation errors are mentioned here.
transactionstatus	string	1	Y – Transaction Successful N – Transaction Failed
pgreturncode	string	10	Return code from gateway
pgreturndescription	string	400	Return description from gateway
transactiondate	date time		Transaction Date and Time
respproc	string	8	Processor Response Code
userfields	string	20	Custom Fields will be included in this parameter
pgtext	string	600	Payment Gateway additional message on failure
paymentmethodtype	string	20	Credit Card / Account type used in transaction
paymentmethodlast4	string	4	Last 4 of the bank account or credit card number used in transaction
paymentmethodtoken	string	50	Payment Gateway Token ID for Credit Card or ACH
paymentmethodfirstname	string	40	First Name
paymentmethodlastname	string	40	Last Name

paymentmethodexpirationmonth	Numeric		Expiration Month
paymentmethodexpirationyear	Numeric		Expiration Year
paymentmethodroutingnumber	string	20	Bank Routing Number
paymentmethodaddressline1	string	40	Address Line 1
paymentmethodcity	string	100	City
paymentmethodstate	string	100	State
paymentmethodzip	string	20	Zip
paymentmethodcountry	string	100	Country
paymentmethodemail	string	100	Email
paymentmethodphonenumber	string	26	Phone Number
feepgtransactionid	string	80	Gateway Transaction ID for fee transaction
feeamount	double		Fee amount with 2 decimals
feeauthorizationnumber	string	10	Authorization Number from gateway for fee transaction
feetransactionstatus	string	1	Y – Fee Transaction Successful N – Fee Transaction Failed.
feepgreturncode	string	10	Return code from gateway for fee transaction
feepgreturndescription	string	400	Return description from gateway for fee transaction
feetransactiondate	date time		Transaction Date and Time for fee transaction
feerespproc	string	8	Processor Response Code for fee transaction
feepgtext	string	600	Payment Gateway additional message on failure for fee transaction
feepaymenttransactionid	Numeric		Payment Transaction ID for fee transaction
feeformat	string	10	Fee Format (flat or percentage). Service fee (MCF) supports both formats but convenience fee supports flat format only.
feetype	string	15	Fee Type, e.g. Service
feevalue	Numeric		Fee Value in %

signature	string		HMAC Signature created with response parameter values
udf0	string	20	User Defined Fields
udf1	string	20	User Defined Fields
udf2	string	20	User Defined Fields
udf3	string	20	User Defined Fields
udf4	string	20	User Defined Fields
udf5	string	20	User Defined Fields
udf6	string	20	User Defined Fields
udf7	string	20	User Defined Fields
udf8	string	20	User Defined Fields
udf9	string	20	User Defined Fields

Note: If the value of transactionstatus in response parameter is Y, then the transaction is considered to be successful. If the value of transactionstatus in response parameter is N, then the transaction is considered as failed transaction. pgreturncode and pgreturndescription parameters will have more information on the failure.

Hosted Payment Page using JavaScript

The SnapPay Hosted Payment Page or SnapPay HPP is the payment solution for online application to implement a simple, customizable payment page that fits their need. SnapPay HPP requires little developer involvement, so setup is quick and simple.

The calling application will open SnapPay HPP as a popup with below parameters embedded in JavaScript. User will submit the payment with credit card or bank details. Upon clicking the submit button, SnapPay will process the payment and post the result parameters back to Merchant's system.

Please refer the sample JavaScript that you need to add on click of a button to open SnapPay HPP.

URL that you provide for parameter "data-snappayurl" is listed under Post URLs section above.

Sample JavaScript

```
<div id="dvsnappay_hppform">
  <!-- HTML is dynamically placed here -->
</div>
<input id="snappayhppform_response" type="hidden" name="snappayhppform_response" />

<script id="snappay_hppform"
    src='https://stage.snappayglobal.com/Areas/Interop/Scripts/HPPForm.js'
    data-target="#snappayhppform_response"
    data-callback="submit_external_ecommerce"
    data-accountid='1000000001'
    data-customerid='100'
    data-currencycode='USD'
    data-transactionamount='123.45'
    data-merchantid='496695144562'
    data-paymentmode='CC' // CC for credit card
    data-cvvrequired='Y'
    data-enableemailreceipt='Y'
    data-redirectionurl='<redirection URL>'
    data-snappayurl='https://stage.snappayglobal.com/Interop/HostedPaymentPage'
    data-udf0='ord-July08'
    data-udf1='inv-July08 '
    data-udf2=''
    data-udf3=''
    data-udf4=''
    data-udf5=''
    data-udf6=''
    data-udf7=''
    data-udf8=''
    data-udf9=''
    data-firstname='John'
    data-lastname='Smith'
    data-addressline1='850 Maple Ave'
    data-city='Aurora'
    data-state='IL'
    data-zip='60504'
    data-country='US'
    data-email='john.smith@abcd.com'
    data-phonenumber='6301234567'
```

```
        data-signature='wYNLEzZFE4 +e9FAk5Rvifnn7hWwFgBGYcHA9+v2Y4dg'>
    </script>
```

Note: Use following URLs for production environment.

```
src='https://www.snappayglobal.com/Areas/Interop/Scripts/HPPForm.js'
data-snappayurl='https://www.snappayglobal.com/Interop/HostedPaymentPage'
```

Request Parameters

Parameter	Data Type	Size	Reqd.	Notes
data-target	string	50	Y	ID of the hidden field where SnapPay will respond with success or error. The response will be in JSON format. In the sample JavaScript the value for this field will be "#snappayhppform_response" – same as the ID of the hidden field.
data-callback	string	50	Y	Name of a JavaScript function in the calling application that will read the response given in data-target element. In the sample given above the value is set to "submit_external_ecommerce".
data-accountid	numeric	10	Y	SnapPay Account ID
data-customerid	string	20	Y	Customer ID
data-currencycode	string	3	Y	Currency Code – USD
data-transactionamount	double		Y	Amount with 2 decimals
data-merchantid	string	20	Y	Merchant ID
data-paymentmode	string	3	Y	CC – for credit card ACH – for bank details
data-cvvrequired	string	1	N	Y/N – determines if verification code is required on the form
data-enableemailreceipt	string	1	N	Y/N – determines if emailing receipt is enabled after transaction is completed. Email should be provided on the form for emailing receipt.
data-redirectionurl	string	256	Y	Redirection URL – SnapPay will redirect the user to the URL specified here after the transaction is successful.
data-signature	string		Y	HMAC Signature
data-firstname	string	40	N	First Name
data-lastname	string	40	N	Last Name
data-addressline1	string	40	N	Address Line 1 (Street Address)
data-city	string	100	N	City
data-state	string	100	N	State
data-zip	string	20	N	Zip/Postal Code

data-country	string	100	N	Country
data-email	string	100	N	Email
data-phonenumber	string	26	N	Phone Number
data-signature	string		Y	HMAC Signature
data-udf0	string	20	N	User Defined Fields
data-udf1	string	20	N	User Defined Fields
data-udf2	string	20	N	User Defined Fields
data-udf3	string	20	N	User Defined Fields
data-udf4	string	20	N	User Defined Fields
data-udf5	string	20	N	User Defined Fields
data-udf6	string	20	N	User Defined Fields
data-udf7	string	20	N	User Defined Fields
data-udf8	string	20	N	User Defined Fields
data-udf9	string	20	N	User Defined Fields

Response Parameters

Parameter	Data Type	Size	Notes
accountid	numeric	10	SnapPay Account ID
customerid	string	20	Customer ID
companycode	string	20	Company Code
currencycode	string	3	Currency Code – USD
merchantid	string	20	Merchant ID
paymenttransactionid	numeric	10	SnapPay Transaction ID
pgtransactionid	string	80	Gateway Transaction ID for reference
transactionamount	double		Amount with 2 decimals
authorizationnumber	string	10	Authorization Number from gateway
returnmessage	string	500	Return message from SnapPay. Validation errors are mentioned here.
transactionstatus	string	1	Y – Transaction Successful N – Transaction Failed
pgreturncode	string	10	Return code from gateway
pgreturndescription	string	400	Return description from gateway
transactiondate	date time		Transaction Date and Time
respproc	string	8	Processor Response Code
userfields	string	20	Custom Fields will be included in this parameter
pgtext	string	600	Payment Gateway additional message on failure
paymentmethodtype	string	20	Credit Card / Account type used in transaction
paymentmethodlast4	string	4	Last 4 of the bank account or credit card number used in transaction
paymentmethodtoken	string	50	Payment Gateway Token ID for Credit Card or ACH

paymentmethodfirstname	string	40	First Name
paymentmethodlastname	string	40	Last Name
paymentmethodexpirationmonth	Numeric		Expiration Month
paymentmethodexpirationyear	Numeric		Expiration Year
paymentmethodroutingnumber	string	20	Bank Routing Number
paymentmethodaddressline1	string	40	Address Line 1
paymentmethodcity	string	100	City
paymentmethodstate	string	100	State
paymentmethodzip	string	20	Zip
paymentmethodcountry	string	100	Country
paymentmethodemail	string	100	Email
paymentmethodphonenumber	string	26	Phone Number
feepgtransactionid	string	80	Gateway Transaction ID for fee transaction
feeamount	double		Fee amount with 2 decimals
feeauthorizationnumber	string	10	Authorization Number from gateway for fee transaction
feetransactionstatus	string	1	Y – Fee Transaction Successful N – Fee Transaction Failed.
feepgreturncode	string	10	Return code from gateway for fee transaction
feepgreturndescription	string	400	Return description from gateway for fee transaction
feetransactiondate	date time		Transaction Date and Time for fee transaction
feerespproc	string	8	Processor Response Code for fee transaction
feepgtext	string	600	Payment Gateway additional message on failure for fee transaction
feepaymenttransactionid	Numeric		Payment Transaction ID for fee transaction
feeformat	string	10	Fee Format (flat or percentage). Service fee (MCF) supports both formats but convenience fee supports flat format only.
feetype	string	15	Fee Type, e.g. Service
feevalue	Numeric		Fee Value in %
signature	String		HMAC Signature created with response parameter values
signature	String		HMAC Signature created with response parameter values
udf0	string	20	User Defined Fields
udf1	string	20	User Defined Fields
udf2	string	20	User Defined Fields
udf3	string	20	User Defined Fields
udf4	string	20	User Defined Fields
udf5	string	20	User Defined Fields
udf6	string	20	User Defined Fields

udf7	string	20	User Defined Fields
udf8	string	20	User Defined Fields
udf9	string	20	User Defined Fields

Note: If the value of transactionstatus in response parameter is Y, then the transaction is considered to be successful. If the value of transactionstatus in response parameter is N, then the transaction is considered as failed transaction. pgreturncode and pgreturndescription parameters will have more information on the failure.

HMAC Validation In Response

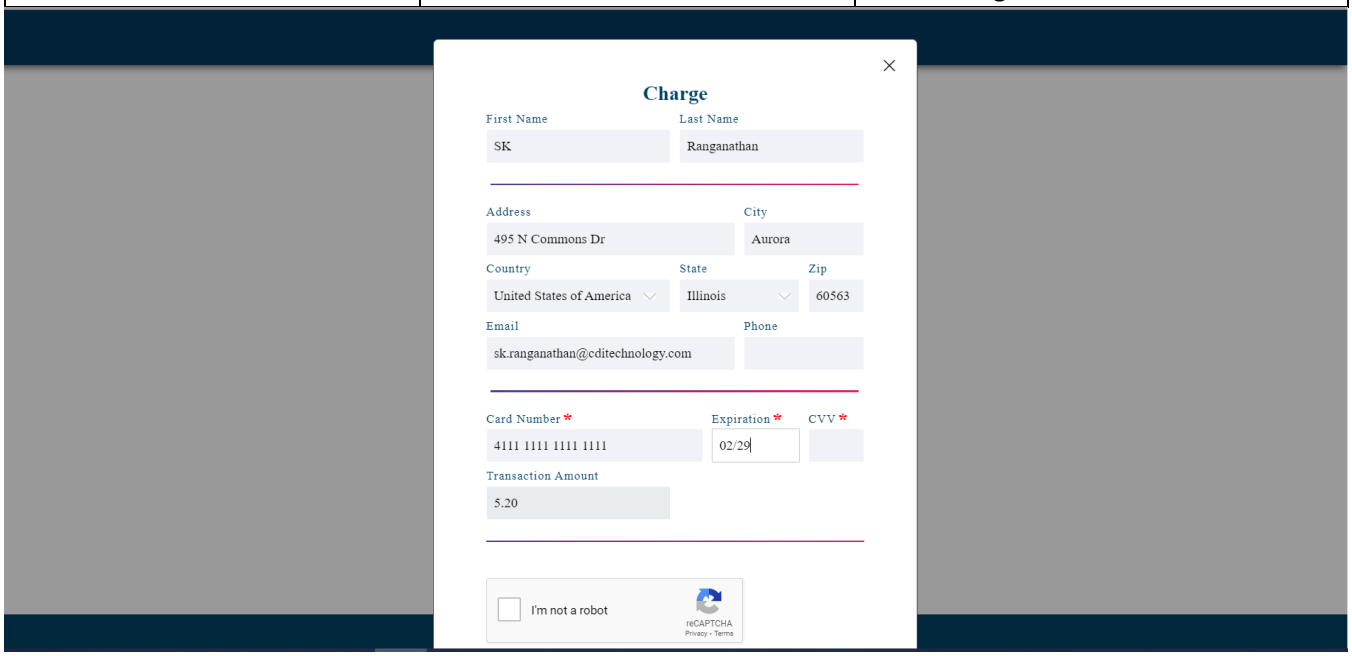
To validate whether the response came from SnapPay (Fiserv) or not, we need to validate the Signature in the response. This is an optional step.

Steps to turn on Signature in response

For Form Post implementation please turn on **EnableHMACForHPPFormPost**

For JS Post implementation please turn on **EnableHMACForHPPJS**

Parameter Name	Value	Description
HPPHMACResponseParameters	accountid,customerid,merchantid,transactionamount,currencycode,paymentmode,pgtransactionid,paymentmethodemail,nonce,timestamp	Parameter that defines HMAC definition for HPP transactions. Use comma to add multiple parameters. Each Parameter listed here should be part of the HPP form post or JS.
EnableHMACForHPPFormPost	Y/N	Enables HMAC authentication in HPP (Hosted Payment Page) for form post integration.
EnableHMACForHPPJS	Y/N	Enables HMAC authentication in HPP (Hosted Payment Page) for JavaScript based integration.



Hosted Payment Page with Credit Card

Once user clicks on Submit button on HPP form user will be redirected with response parameters including signature that need to be validated by the caller.

Example	
HMAC Response Parameter used to calculate HMAC (**HPPHMACResponseParameters**)	accountid,customerid,merchantid,transactionamount,currencycode,paymentmode,pgtransactionid,paymentmethodemail,nonce,timestamp
Response Value (**signature**)	ZTN5NERQNDlNdDc3cXF4Y1ZBUkJ3bGR1MUs3dkxsZW1aSFBNZ0E0Znp1cz06ZjMwNjQwZWM0MjRiNDMyNzlkYmRjMGE0NGFlNWY3MTE6MTY1NzU0Njg3MQ==

Optional: GetTransaction API

GetTransaction API can be called as a callback API to verify the transaction response (in case if the eCommerce site didn't get any response for more than 2 minutes). This callback will help the eCommerce understand if the payment happened successfully in SnapPay or not. This will avoid dual payments. For this callback to work the eCommerce has to pass a unique reference ID against the transaction. The reference ID can be an Order ID or Invoice ID. Following steps details on how to accomplish this.

Admin Configuration

Hosted Payment Page Custom Field attribute. Set the ordered and/or invoiceid to respective udf0 and udf1. It's fine if there is just Order ID – assign the Order ID as Label and description against UDF0 and mark it as Display and Required.

Export To Excel

Custom Field Name	Label	Description	Max Length	Data Type	Display	Required	Sequence	Default Value	Is Read Only
firstname	First Name	Name on payment method	40	String	☑	☐	1		☐
lastname	Last Name	Name on payment method	40	String	☑	☐	2		☐
address	Address	Address on payment method	40	String	☑	☐	3		☐
city	City	City of payment method	20	String	☑	☐	4		☐
state	State	State of payment method	20	String	☑	☐	5		☐
country	Country	County of payment method	15	String	☑	☐	6		☐
zip	Zip	Zip code of payment method	20	String	☑	☐	7		☐
phone	Phone	Phone number of payment method holder	20	String	☑	☐	8		☐
email	Email	Email address of payment method holder	100	String	☑	☐	9		☐
udf1	invoice ID	Invoice ID	20	String	☑	☐	10		☐
udf2	custome2	custom 2	20	String	☐	☐	11		☐
udf3	Custom 3	Custom 3	20	String	☐	☐	12		☐
udf4	Custom 4	Custom 4	20	String	☐	☐	13		☐
udf5	Custom 5	Custom 5	20	String	☐	☐	14		☐
udf6	Custom 6	Custom 6	20	String	☐	☐	15		☐
udf7	Custom 7	Custom 7	20	String	☐	☐	16		☐
udf8	Custom 8	Custom 8	20	String	☐	☐	17		☐
udf9	Custom 9	Custom 9	20	String	☐	☐	18		☐
udf0	Order Id	Order ID	20	String	☑	☑	19		☐

Following parameters are applicable to map the UDF fields to respective reference fields (Order Id, Invoice ID in SnapPay)

Parameter Name	Value	Description
EnableL2FieldsAsCustomFieldsHPP	Y/N	Determines if level 2 fields like order ID, invoice ID can be used in custom fields like udf0, udf1 etc. for Hosted Payment Page.
CustomFieldForOrderIDHPP	udf0 – udf9	Determines which custom field can be used to send order ID to payment gateway for Hosted Payment Page.
CustomFieldForInvoiceIDHPP	udf0 – udf9	Determines which custom field can be used to send invoice ID to payment gateway for Hosted Payment Page.

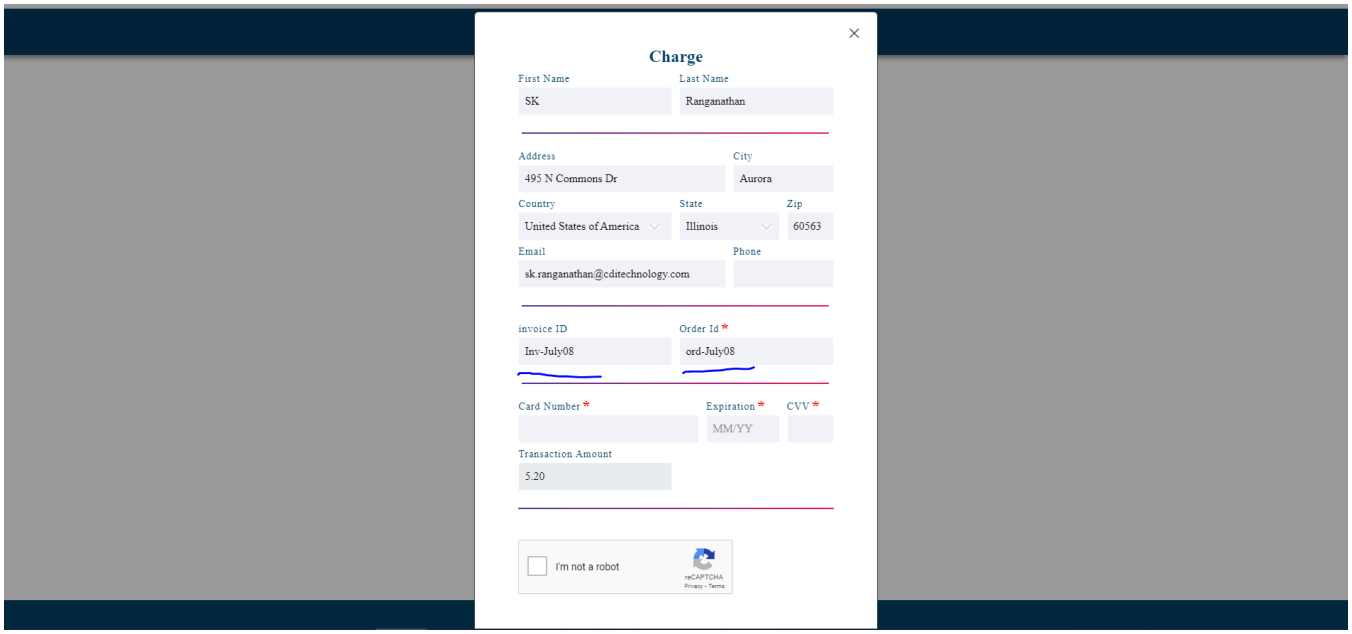
Note: Turn on the EnableL2FieldsAsCustomFieldsHPP parameter for sending order ID to gateway and set the CustomFieldForOrderIDHPP to udf0 and /or CustomFieldForInvoiceIDHPP to udf1

How to pass UDF fields in request

Enter the unique orderid and invoiceid for sale transaction using data-udf0 and date-udf1 fields.

Integration	Parameter	Value
Form Post	udf0	Ord-July08
Form Post	udf1	Inv-July08
JS	data-udf0	Ord-July08
JS	data-udf1	Inv-July08

Customer will be able to see the provided Order ID and/ or Invoice ID in HPP form.



Once the transaction completes, the response will show the UDF fields and signature.

Optional Step: Call the GetTransactionAPI using orderId and/or invoiceId. eCommerce site will be able to get the payment transaction details based on information passed.

Request	Response
{ "accountid":"1000000001", "orderid":"ord-July08" }	{ "accountid": 1000000001, "message": "Success", "transactions": [{ "transaction": { "transactionstatus": "Success", "merchantid": "496695144562", "returncode": "000", "returndescription": "Approval", "pgtransactionid": "189776240479", "pgtransactiontype": "Charge", "paymenttransactionid": "75847", "authorizationcode": "PPS776", "transactiondate": "07/08/2022 11:14:39 AM", "procrespcode": "RPCT", "transactionamount": 5.2, "currency": "USD", "totaltransactionamount": 5.2, "feeamount": 0, "feevalue": 0 }, "paymentmethod": { "paymentmethodid": 41166, "type": "VISA", "last4": "1111", "tokenid": "9417085548851111", "firstname": "SK", "lastname": "Ranganathan", "expirationmonth": "2", "expirationyear": "2029", "addressline1": "495 N Commons Dr", "city": "Aurora", "state": "IL", "zip": "60563", "country": "US", "email": "sk.ranganathan@cditechnology.com" } }] }

Appendix A: Test Data

Test Cards

Note: Remove XX from the Card # and use any date in future for expiration date.

Card Number	Type
4111XX111111111111	Visa
4637XX090000158588	Visa
5399XX104611689124	MasterCard
5454XX545454545454	MasterCard
6011XX000991300009	Discover
3714XX49635398431	AMEX

Appendix B: How to generate HMAC signature for SnapPay HPP Form Post.

This appendix provides details for creating HMAC signature for opening SnapPay HPP by posting form parameters. This is not applicable if you are using Virtual terminal or SnapPay API as described above in this document.

What's needed for creating HMAC signature for SnapPay HPP.

Following information is needed to generate HMAC signature.

- API Authentication Code (*wYNLEzZFE4+e9FAk5Rvifnn7hWwFgBGYcHA9+v2Y4dg=*)
- SnapPay Account ID – 10 digit Account ID (*1000000002*)
- Value of HPPHMACParameters parameter – elements separated by comma will be considered to create the HMAC signature as shown below (*accountid,customerid,merchantid,transactionamount,currencycode,paymentmode,email*)

Content in green text are sample values for documentation purpose, please check with your account manager to get your account specific values.

Steps to Create HMAC signature

1. Create a unique identifier (mostly GUID) for each request

   Sample C#.Net Code

   ```
   string nonce = Guid.NewGuid().ToString("N");
   ```

2. Create Request Time Stamp - this is to make sure that the incoming request is within certain time limit.

   Sample C#.Net Code

   ```
   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();
   ```

3. Create the raw signature string in the given format. Take the list of parameters value of

   Sample C#.Net Code

   ```
   string signatureRawData =  accountid  + customerid + merchantid + transactionamount + currencycode +
   paymentmode + email + nonce
   ```

4. Convert API Authentication Code to base 64 string

   Sample C#.Net Code

   var secretKeyByteArray = Convert.FromBase64String(apiAuthCode);

5. Convert raw signature data to UTF 8 encoded byte array.

   Sample C#.Net Code

   byte[] signature = Encoding.UTF8.GetBytes(signatureRawData);

6. Create HMAC signature from API Authentication Code which is converted to base 64 string

   Sample C#.Net Code

```
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);
}
```

7. Get base 64 encoded string

   Sample C#.Net Code

```
string signatureData = string.Format("{0}:{1}", requestSignatureBase64String, nonce);
string HmacValue = Convert.ToBase64String(Encoding.UTF8.GetBytes(signatureData));
```

C#.Net Sample Code to create HMAC Signature

```
string apiAuthCode = "wYNLEzZFE4 +e9FAk5Rvifnn7hWwFgBGYcHA9+v2Y4dg=";
string accountid = "1000000002";
string customerid = "12234";
string merchantid = "454464567567";
string transactionamount = "105.23";
string currencycode = "USD";
string paymentmode = "CC"; // or "ACH"
string email = "abc@xyz.com";

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();

signatureRawData =  accountid  + customerid + merchantid +  transactionamount + currencycode +
paymentmode + email + 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));
```

Appendix C: How to generate HMAC signature for SnapPay JavaScript HPP.

This appendix provides details for creating HMAC signature for opening SnapPay HPP using JavaScript. This is not applicable if you are using Virtual terminal or SnapPay API as described above in this document.

What's needed for creating HMAC signature for SnapPay HPP.

Following information is needed to generate HMAC signature.

- API Authentication Code (*wYNLEzZFE4+e9FAk5Rvifnn7hWwFgBGYcHA9+v2Y4dg=*)
- SnapPay Account ID – 10 digit Account ID (*1000000002*)
- Value of HPPHMACParameters parameter – elements separated by comma will be considered to create the HMAC signature as shown below (*accountid,customerid,merchantid,transactionamount,currencycode,paymentmode,email,nonce*)

Content in green text are sample values for documentation purpose, please check with your account manager to get your account specific values.

Steps to Create HMAC signature

1. Create a unique identifier (mostly GUID) for each request

   Sample JavaScript Code

```
var txtnonce = createGuid();

Add the below function in your JavaScript
function createGuid()
{
   return 'xxxxxxxxxxxx4xxxyxxxxxxxxxxxxxxx'.replace(/[xy]/g, function(c) {
      var r = Math.random()*16|0, v = c === 'x' ? r : (r&0x3|0x8);
      return v.toString(16);
   });
}
```

2. Create Request Time Stamp - this is to make sure that the incoming request is within certain time limit.

   Sample C#.Net Code

```
var txttimestamp = "";
var epochStart = new Date(Date.UTC(1970, 01, 01, 0, 0, 0, 0));
var timeSpan = Date.now() - epochStart;
txttimestamp = timeSpan;
```

3. Create the raw signature string in the given format.

Sample JavaScript Code

```
var signatureRawData = accountid + customerid + merchantid + transactionamount +
currencycode + paymentmode + email + nonce + txttimestamp;
```

4. Convert API Authentication Code to base 64 string

Sample JavaScript Code

```
var secretkey = CryptoJS.enc.Base64.parse(apiAuthCode);
```

5. Convert raw signature data to UTF 8 encoded byte array.

Sample JavaScript Code

```
var prehash = CryptoJS.enc.Utf8.parse(signatureRawData);
```

6. Create HMAC signature from API Authentication Code which is converted to base 64 string

Sample JavaScript Code

```
var hash = CryptoJS.HmacSHA256(prehash, secretkey);
var signature = hash.toString(CryptoJS.enc.Base64);
```

7. Get base 64 encoded string

Sample JavaScript Code

```
var signatureData =  signature + ":" + nonce;
var prehash1 = CryptoJS.enc.Utf8.parse(signatureData);
var HmacValue = prehash1.toString(CryptoJS.enc.Base64);
```

JavaScript Sample Code to create HMAC Signature

Add reference to “crypto-js.js” file in your JavaScript by using following line.

Stage: https://stage.snappayglobal.com/HMAC/crypto-js.js
Production: https://www.snappayglobal.com/HMAC/crypto-js.js

```
var apiAuthCode = "wYNLEzZFE4 +e9FAk5Rvifnn7hWwFgBGYcHA9+v2Y4dg=";
var accountid = "1000000002";
var customerid = "12234";
var merchantid = "454464567567";
var transactionamount = "105.23";
var currencycode = "USD";
var paymentmode = "CC";
var email = "abc@xyz.com";
var txtnonce = createGuid();
var txttimestamp = "";
```

```
var epochStart = new Date(Date.UTC(1970, 01, 01, 0, 0, 0, 0));
var timeSpan = Date.now() - epochStart;
txttimestamp = timeSpan;

var signatureRawData = accountid + customerid + merchantid + transactionamount +
currencycode + paymentmode + email + nonce + txttimestamp;
var secretkey = CryptoJS.enc.Base64.parse(apiAuthCode);
var prehash = CryptoJS.enc.Utf8.parse(signatureRawData);
var hash = CryptoJS.HmacSHA256(prehash, secretkey);
var signature = hash.toString(CryptoJS.enc.Base64);
var signatureData =  signature + ":" + nonce + ":" + txttimestamp;
var prehash1 = CryptoJS.enc.Utf8.parse(signatureData);
var HmacValue = prehash1.toString(CryptoJS.enc.Base64);


function createGuid()
{
   return 'xxxxxxxxxxxx4xxxyxxxxxxxxxxxxxxx'.replace(/[xy]/g, function(c) {
      var r = Math.random()*16|0, v = c === 'x' ? r : (r&0x3|0x8);
      return v.toString(16);
   });
}
```

Appendix D: How to Validate HMAC signature in response.

This appendix provides details for validating HMAC signature.

Two steps validation require at client side.

1) Need to have a web method which will validate response signature based on account parameters.
2) Need to have a Javascript function to access the web method

What's needed for validating HMAC signature.

Following information is needed to generate HMAC signature.

- API Authentication Code (*wYNLEzZFE4+e9FAk5Rvifnn7hWwFgBGYcHA9+v2Y4dg=*)
- SnapPay Account ID – 10 digit Account ID (*1000000002*)
- SnapPay HMAC Signature - `It will be sent by SnapPay in the response` (*ZTN5NERQNDlNdDc3cXF4Y1ZBUkJ3bGR1MUs3dkxsZW1aSFBNZ0E0Znp1cz06ZjMwNjQwZWM0MjRiNDMyNzlkYmRjMGE0NGFlNWY3MTE6MTY1NzU0Njg3MQ==*)
- Value of HPPHMACResponseParameters parameter – elements separated by comma will be considered to create the HMAC signature as shown below (*accountid,customerid,merchantid,transactionamount,currencycode,paymentmode,pgtransactionid,paymentmethodemail,nonce,timestamp*)

Content in green text are sample values for documentation purpose, please check with your account manager to get your account specific values.

The incoming signature from SnapPay should match the signature generated by following code.

C#.Net Sample Code to validate HMAC Signature

```
        [WebMethod]
         public static string ValidateHMAC(string param)
         {
             string hmacHeader = string.Empty;
             try
             {
                 string[] arr = param.Split('|');
                 string signatureFromSnapPay = arr[2];//It will be sent by SnapPay in the
response
                 Encoding encoding = Encoding.GetEncoding("iso-8859-1");
                 //Retrieve the actual signature, nonce and timestamp from the response
                 var rawAuthzHeader =
encoding.GetString(Convert.FromBase64String(signatureFromSnapPay));
                 var authorizationHeaderArray = rawAuthzHeader.Split(':');
                 if (authorizationHeaderArray != null)
                 {
                     var incomingBase64Signature = authorizationHeaderArray[0];
                     var nonce = authorizationHeaderArray[1];
                     var timestamp = authorizationHeaderArray[2];

                     //Now generate signature to compare with signature given by SnapPay
```

```
                    //To generate the signature concatenate all the fields value mentioned in
HPPHMACResponseParameters except nonce and timestamp. ex.1000000007100172714966951445625.2USDCC
                    string hppHMACParamValue = arr[0];
                    //Now append nonce and timestamp retrieved above to this value
                    string data = String.Format("{0}{1}{2}", hppHMACParamValue, nonce,
timestamp);
                    byte[] signature = Encoding.UTF8.GetBytes(data);

                    //It should be APIAuthenticationCode config parameter value.
ex.ctwJhiZ6BYgo5o9M5ImcDDj+koPSN3r5LwsC+5UNXgj05U02E8PjrA==
                    string secretkey = arr[1];
                    var secretKeyByteArray = Convert.FromBase64String(secretkey);
                    string convertedInputString = string.Empty;
                    using (HMACSHA256 hmac = new HMACSHA256(secretKeyByteArray))
                    {
                        byte[] signatureBytes = hmac.ComputeHash(signature);
                        convertedInputString = Convert.ToBase64String(signatureBytes);
                    }

                    //Now compare this signature with SnapPay signature
                    bool isValid = incomingBase64Signature.Equals(convertedInputString,
StringComparison.Ordinal);
                    hmacHeader = isValid ? "Valid signature." : "Invalid signature.";
                }
            }
            catch (Exception ex)
            {
                hmacHeader = ex.Message;
            }
            hmacHeader = JsonConvert.SerializeObject(hmacHeader);
            return hmacHeader;
        }
```

Javascript Sample Code to call the web method validate HMAC Signature

```
            function VerifyHMACFromSnapPay() {
                        //Validate client side url which will point to webmethod for verify
            hmac
            var url =
            "https://demo1.cditechnology.com/ExternalApplication/SnapPayResponse.aspx/Validate
            HMAC";

                        var output = "";

            var hpphmacparam = $('#hpphmacresponseparameters').val();//
            HPPHMACResponseParameters config parameter.
                        if (hpphmacparam != "" && hpphmacparam != undefined) {
              var hpphmacparamarr = hpphmacparam.split(",");
              var input = "";
              $.each(hpphmacparamarr, function (i, obj) {
                  var paramval = $("#" + hpphmacparamarr[i]).text();
                  if (hpphmacparamarr[i] != "nonce" && hpphmacparamarr[i] != "timestamp") {
                      if (paramval != "" && paramval != undefined) {
                          input += paramval;
```

```
                    }
                }
            });
            input += "|" + $("#secretkey").val();// API Authentication Code
            input += "|" + $("#signature").text(); // HMAC signature generated during
transaction
            $.ajax({
                cache: false,
                url: url,
                type: "POST",
                async: false,
                dataType: "json",
                contentType: "application/json",
                data: '{param: "' + input + '"}',
                success: function (response) {
                    output = JSON.parse(response.d);
                    $("#signaturestatus").text(output);
                    if (output.startsWith("Invalid")) {
                        $('#signaturestatus').css("color", "red");
                    }
                    else {
                        $('#signaturestatus').css("color", "green");
                    }
                },
                error: function (request, status, error) {
                }
            });
            return false;
        }
    }
```