Moneris™

BE PAYMENT READY

Hosted Tokenization Merchant Integration Guide

Version 1.1.0 – July 2023

Please Read Carefully

1. You have a responsibility to protect cardholder and merchant related confidential account information. Under no circumstances should ANY confidential information be sent via email while attempting to diagnose integration or production issues. When sending sample files or code for analysis by Moneris staff, all references to valid card numbers, merchant accounts and transaction tokens should be removed and or obscured. Under no circumstances should live cardholder accounts be used in the test environment.
2. The Transaction Risk Management Tool provides additional information to assist in identifying fraudulent transactions. In order to maximize the benefits from the Transaction Risk Management Tool it is highly recommended that you:
   a. Carefully consider the business logic and processes that you need to implement surrounding handling the response information the Transaction Risk Management Tool provides.
   b. Also implement the other fraud tools available through Moneris Gateway (e.g., AVS, CVD, Verified by Visa and MasterCard SecureCode).
3. When testing the Transaction Risk Management Tool there is specific test data that you will need to use. Please carefully review and follow the testing instructions and data provided in the document.

Getting Help

Moneris has help for you at every stage of the integration process.

Getting Started	During Development	Production
Contact our Client Integration Specialists: clientintegrations@moneris.com Hours: Monday – Friday, 8:30am to 8 pm ET	If you are already working with an integration specialist and need technical development assistance, contact our eProducts Technical Consultants: 1-866-319-7450 eproducts@moneris.com Hours: 8am to 8pm ET	If your application is already live and you need production support, contact Moneris Customer Service: onlinepayments@moneris.com 1-866-319-7450 Available 24/7

For additional support resources, you can also make use of our community forums at

http://community.moneris.com/product-forums/

Table of Contents

About this documentation

This document contains a guide for using the Hosted Tokenization configuration tool in the Merchant Resource Centre of Moneris Gateway. Also described are the methods for sending and processing a Hosted Tokenization transaction and managing the responses from the transaction.

Skills and System Requirements

In order to use Hosted Tokenization your system will need the following:

- A web server capable of sending and receiving an HTML POST/GET

In addition, you will need the following knowledge and/or skill set:

- Knowledge of creating an HTML web page and posting forms.
- Knowledge of iframes
- If you are selling more than one item, you will need some knowledge of a client-side scripting language (JavaScript, PHP, etc.) to calculate a final charge amount.
- If you want to create your own custom receipts and perform transaction verification, you will require knowledge of a server-side scripting language (PHP, Perl, ASP, etc.)

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). 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.

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.

1 Hosted Tokenization

The Moneris Hosted Tokenization (HT) was designed as a solution for online e-commerce merchants that do not wish to handle credit card information directly on their websites and also have the ability to fully customize their check-out webpage's appearance.

When HT is initiated, the Moneris Gateway will present and display on the merchant's behalf, a text-box on the check-out page for card number entry. Optionally, within the HT iframe, a text box for the expiration date and CVD data can be displayed as well along with the card number text box. The cardholder can then securely enter their credit card information into the text-boxes within the iframe; a number keypad is included in the iframe for assisting with entry as well.

Upon submission of the payment information on the check-out page, the Moneris Gateway will return a temporary token to the merchant, representing the credit card information. This token would then be used by the merchant to process a financial transaction via API with Moneris in order to charge the card. Upon receiving a response to the financial transaction, the merchant would then generate a receipt and allow the cardholder to continue on with the online shopping experience.

A benefit of integration with Moneris' Hosted Tokenization is the solution will reduce a merchant's PCI-compliance assessment scope due to the fact that credit card information is not captured nor stored by the merchant's site.

2 Introduction to Moneris Hosted Payment Solutions

Moneris' Hosted Tokenization Solution allows you as a merchant to securely accept payment information from your customers. This is one of the simplest integration methods to accept payments on your website. A few simple lines of coding will allow you to get started with your online payments.

3 Hosted Payment Page Configuration Tool

The Hosted Tokenization Configuration Tool, part of the Merchant Resource Center, is where you create and configure a profile for your hosted payment solution. Creating and configuring a hosted payment solution profile are the two first steps in the process of integrating the hosted payment solution with your system. To review the steps for integrating your hosted payment solution, refer to Integrating Your Hosted Solution (see page 1).

In order to use the Hosted Tokenization Configuration Tool to create and configure a hosted payment solution profile, you need to log in to the Merchant Resource Center test environment.

To log into the Merchant Resource Center test environment go to

https://esqa.moneris.com/mpg

and use one of the following login IDs.

Table 1 Test IDs for Merchant Resource Center

Store ID	Username	Password
store1	DemoUser	password
store2	DemoUser	password
store3	DemoUser	password
store5	DemoUser	password
moneris	DemoUser	password

Table 2 Test IDs for Merchant Resource Center – INTERAC® Online Payment

Store ID	Username	Password
store3	DemoUser	password

Table 3 Test IDs for Merchant Resource Center – Convenience Fee

Store ID	Username	Password
monca00392	DemoUser	password

3.1 Creating a Hosted Payment Solution Configuration Profile

The first step in the process of integrating your hosted payment solution is creating a configuration profile for it using the Merchant Resource Center Hosted Tokenization Configuration Tool.

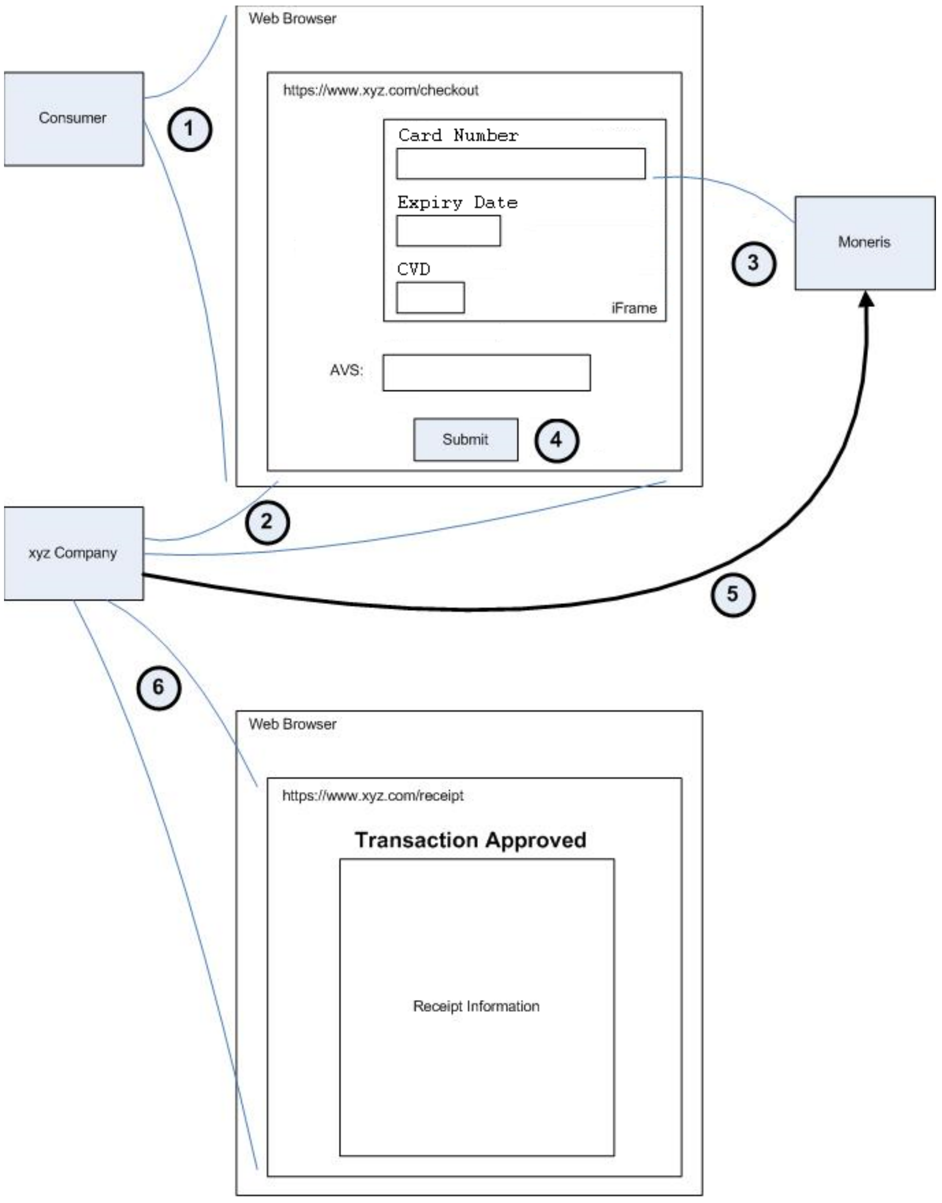
3.1.1 Creating a Hosted Tokenization Configuration

To create a Hosted Tokenization Configuration:

1. Log in to the Merchant Resource Center
    - QA: https://esqa.moneris.com/mpg
    - Production: https://www3.moneris.com/mpg
2. Click on **Admin** in the menu
3. Click on **Hosted Tokenization** in the sub-menu
4. (Optional) Enter the Source Domain page. This is the address of the main outer page that sends the transaction to the Moneris Gateway. Example from the process flow diagram above would be "https://www.xyz.com". This may be left blank for mobile solutions or if the profile is being used by multiple domains.
5. Click **Create Profile**
6. Make a note of the Profile ID that is generated since this will need to be included in your HTML iframe code

4 Developing for Hosted Tokenization

4.1 Hosted Tokenization Process Flow



1. The cardholder shops at a merchant site with their web-browser and ready to check out.
2. The check-out page is presented by the merchant’s server with Hosted Tokenization integration.

3. A small portion of the merchant's check-out page has an iframe that links to Moneris' Hosted Tokenization configuration. The text-boxes to collect the credit card data are presented by Moneris.
4. The cardholder enters the credit card data and other payment-related information that the merchant may need in order to process a financial transaction to charge the card. Once the cardholder presses the Submit button, the initial code behind the check-out page submits a request to the Moneris Moneris Gateway to obtain the temporary token that represents the credit card data. The latter code behind the check-out page then takes the token and other payment-related information from the check-out page and submits them to the merchant's choice of URL that collects the submitted information.
5. The merchant's server sends a Vault transaction to Moneris using the payment information collected by the URL in step 4. Information in the response to the Vault transaction is saved for reference. For more information on Vault, please refer to our Developer Portal at https://developer.moneris.com/
6. Result of the financial transaction is displayed to the cardholder.

4.2 Sending a Hosted Tokenization Request

- 4.2.1 Getting a Temporary Token
- 4.2.2 Forwarding a Temporary Token to Payment Processing Page
- 4.2.3 Processing the Payment - Hosted Tokenization

4.2.1 Getting a Temporary Token

To get a temporary token you will need to send a request to Moneris from within an iframe. A sample code is illustrated below. Note that the Profile ID in the HTML link below will need to be replaced with your own Profile ID from Creating a Hosted Tokenization Configuration.

EXAMPLE: If your Profile ID is htCCLFFI2H31LBK then replace the highlighted ID in the sample below with your ID:

NOTE: If you are integrating your hosted payment solution with Internet Explorer 7, refer to Appendix A Internet Explorer 7 Compatibility on page 1.

Sample Getting a Temporary Token

```
src="https://esqa.moneris.com/HPPtoken/index.php?id=ht4RXXBKV9T52A8&css_
body=background:green;&css_textbox=border-width:2px;&css_textbox_pan=width:140px;&enable_
```

Sample Getting a Temporary Token

```
exp=1&css_textbox_exp=width:40px;&enable_cvd=1&css_textbox_cvd=width:40px"
<html>
<head>
<title> Outer Frame - Merchant Page</title>
<script>
function doMonerisSubmit()
{
var monFrameRef = document.getElementById('monerisFrame').contentWindow;
monFrameRef.postMessage('tokenize','https://gatewayqa.moneris.com/HPPtoken/index.php');
//change link according to table above
return false;
}
var respMsg = function(e)
{
var respData = eval("(" + e.data + ")");
document.getElementById("monerisResponse").innerHTML = e.origin + " SENT " + " - " +
respData.responseCode + "-" + respData.dataKey + "-" + respData.errorMessage;
document.getElementById("monerisFrame").style.display = 'none';
}
window.onload = function()
{
if (window.addEventListener)
{
window.addEventListener ("message", respMsg, false);
}
else
{
if (window.attachEvent)
{
window.attachEvent("onmessage", respMsg);
}
}
}
</script>
</head>
<body>
<div>This is the outer page</div>
<div id=monerisResponse></div>
<iframe id=monerisFrame
src=https://gatewayqa.moneris.com/HPPtoken/index.php?id=htFTMQ8J63EYNZS&pmmsg=true&css_
body=background:green;&css_textbox=border-width:2px;&css_textbox_pan=width:140px;&enable_
exp=1&css_textbox_exp=width:40px;&enable_cvd=1&css_textbox_cvd=width:40px&enable_exp_
formatting=1&enable_cc_formatting=1 frameborder='0' width="200px" height="200px"></iframe>

<input type=button onClick=doMonerisSubmit() value="submit iframe">
</body>
</html>
```

The IFrame is opened by passing the below arguments as a query string to the following url:

QA: https://esqa.moneris.com/HPPtoken/index.php

Production: https://www3.moneris.com/HPPtoken/index.php

Variable name	Definition
Id	Required - Provided by the Hosted Tokenization profile configuration tool in the MRC.
css_body	Required - CSS applied to the body. By default margin and padding is set to 0.
css_textbox	Required - CSS applied to all text boxes in general.
css_textbox_pan	Optional - CSS applied to the pan text box specifically.
enable_exp	Optional - Must be set to 1 for expiry date text box to be displayed
css_textbox_exp	Optional - CSS applied to the expiry date text box specifically.
enable_cvd	Optional - Must be set to 1 for CVD text box to be displayed
css_textbox_cvd	Optional - CSS applied to the CVD text box specifically.
enable_exp_formatting	Optional - Formatting applied to expiry date field to display a slash between month and year (Format: MM/YY). Must be set to 1 to enable formatting.
enable_cc_formatting	Optional - Formatting applied to credit card number based on the card type (Format: Visa - 4242 4242 4242 4242, MC - 5454 5454 5454 5454, Amex - 333 666666 55555)
display_labels	Optional – 0 for no labels, 1 for traditional labels, 2 for place holder labels.
css_input_label	Optional – CSS for input labels
css_label_pan	Optional – CSS for card number label
css_label_exp	Optional – CSS for expiry date label
css_label_cvd	Optional – CSS for CVD label
pan_label	Optional – text for card number label (default is “Card Number”)
exp_label	Optional – text for expiry date label (default is “Expiry Date”)
cvd_label	Optional – text for CVD label (default is “CVD”)

The response will be returned as JSON. It will contain 4 arguments:

- responseCode - Indication whether the page-loading or card-submission was successful or why it failed. Please note, if expiry text box or CVD text box are enabled, the returned responseCode value will be in the form of a list (e.g. [“944”,”943”]), since there may be more than one failure. For example, in the case where both the card number entered and expiry date are invalid. If only the card number text box is displayed, the responseCode will be returned in the form of a string.
- errorMessage - Description of failure (This is a very generic description – see “responseCode Definitions” below for specific error code results).
- bin - BIN range of the submitted card number. Provides merchant ability to determine the card type and perform any card-specific processing.
- dataKey - Tokenized card number. This is what is used with the Vault API transaction.

Table 1 Error Codes - Hosted Tokenization

Code	Message/Description
001	approved
940	Invalid profile id (on tokenization request)
941	Error generating token
942	Invalid Profile ID, or source URL
943	Card data is invalid (not numeric, fails mod10, we will remove spaces)
944	Invalid expiration date (mmyy, must be current month or in the future)
945	Invalid CVD data (not 3-4 digits)

4.2.2 Forwarding a Temporary Token to Payment Processing Page

To charge the card using the temporary token you will need to send the temporary token to a page on your site that implements the Moneris Vault API.

The token will be received by the below JavaScript function in your code, this is the part of the page where you would put any code that forwards the token to a secondary page for processing.

Sample Forwarding a Temporary Token

```
var respMsg = function(e)
{
var respData = eval("(" + e.data + ")");
document.getElementById("monerisResponse").innerHTML = e.origin + " SENT " + " - " +
respData.responseCode + "-" + respData.dataKey + "-" + respData.errorMessage;
document.getElementById("monerisFrame").style.display = 'none';
// your token will be in the filed: respData.dataKey
// from this point in the Javascript you can have more code that posts the token to another
```

Sample Forwarding a Temporary Token

```
page that will actually process the payment.
}
```

4.2.3 Processing the Payment - Hosted Tokenization

To charge the card on your payment processing page you will need to use one of the Moneris Vault API's. The Moneris Vault API's are available in various programming languages. Below is a Java sample code outlining a Purchase transaction that utilizes the temporary token to charge the card.

Sample Processing the Payment - Hosted Tokenization

```
public class TestResPurchaseCCExpDate
{
public static void main(String args[]) throws IOException
{
/********************** Request Variables ****************************/
String host = "esqa.moneris.com";
String store_id = "store1";
String api_token = "yesguy";
/********************** Transaction Variables ***********************/
String data_key = "ot-pILqxjIp3BPZgLgN80roTHrAH"; //Temp Token from Hosted Tokenization
process.
String order_id; // Application will provide this unique value.
String cust_id = "Hilton_1";
String amount = "1.00";
String crypt_type = "7";
String exp_date = "1209";
/************************ Request Object ******************************/
ResPurchaseCC resPurchaseCC = new ResPurchaseCC(data_key, order_id, amount, crypt_type);
resPurchaseCC.setCustId(cust_id);
// IMPORTANT: ideally the expiration date should be set within the iframe (This is done by
setting the optional expiry text box field to enabled). If not enabled, the expiration date
will need to be set by the setExpdate call.
// resPurchaseCC.setExpdate(exp_date);
// IMPORTANT note: ideally if the CVD feature is used, the value should be set within the
iframe (This is done by setting the optional CVD text box field to enabled). If not
enabled, the CVD value can be passed through the setCvdInfo call. In production, if the CVD
feature will be used it needs to be enabled on the merchant account.
// CvdInfo cvd = new CvdInfo ("1", "789");
// resPurchaseCC.setCvdInfo(cvd);
ResolverHttpsPostRequest mpgReq =
new ResolverHttpsPostRequest(host, store_id, api_token, resPurchaseCC);
/************************ Receipt Object ******************************/
try
{
ResolverReceipt resreceipt = mpgReq.getResolverReceipt();
ResolveData resdata = resreceipt.getResolveData();
System.out.println("DataKey = " + resreceipt.getDataKey());
System.out.println("ReceiptId = " + resreceipt.getReceiptId());
System.out.println("ReferenceNum = " + resreceipt.getReferenceNum());
System.out.println("ResponseCode = " + resreceipt.getResponseCode());
System.out.println("AuthCode = " + resreceipt.getAuthCode());
System.out.println("Message = " + resreceipt.getMessage());
System.out.println("TransDate = " + resreceipt.getTransDate());
System.out.println("TransTime = " + resreceipt.getTransTime());
```

Sample Processing the Payment - Hosted Tokenization

```
System.out.println("TransType = " + resreceipt.getTransType());
System.out.println("Complete = " + resreceipt.getComplete());
System.out.println("TransAmount = " + resreceipt.getTransAmount());
System.out.println("CardType = " + resreceipt.getCardType());
System.out.println("TxnNumber = " + resreceipt.getTxnNumber());
System.out.println("TimedOut = " + resreceipt.getTimedOut());
System.out.println("ResSuccess = " + resreceipt.getResSuccess());
System.out.println("PaymentType = " + resreceipt.getPaymentType() + "\n");
//Contents of ResolveData
System.out.println("Cust ID = " + resdata.getResCustId());
System.out.println("Phone = " + resdata.getResPhone());
System.out.println("Email = " + resdata.getResEmail());
System.out.println("Note = " + resdata.getResNote());
System.out.println("MaskedPan = " + resdata.getResMaskedPan());
System.out.println("Exp Date = " + resdata.getResExpDate());
System.out.println("Crypt Type = " + resdata.getResCryptType());
System.out.println("Avs Street Number = " + resdata.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + resdata.getResAvsStreetName());
System.out.println("Avs Zipcode = " + resdata.getResAvsZipcode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
public class TestResPurchaseCC
{
public static void main(String args[]) throws IOException
{
/********************** Request Variables ****************************/
String host = "esplusqa.moneris.com";
String store_id = "monusqa002";
String api_token = "qatoken";
/********************** Transaction Variables ************************/
String data_key = " pILqxjIp3BPZgLgN80roTHrAH ";
String order_id; // will prompt for user inputs
String cust_id = "Hilton_1";
String amount = "1.00";
String crypt = "7";
String commcard_invoice = "123456789123456789";
String commcard_tax_amount = "1.00";
//String dynamic_descriptor = "123456";
String exp_date = "2109";

InputStreamReader isr = new InputStreamReader( System.in );
BufferedReader stdin = new BufferedReader( isr );
System.out.print( "Please enter an order ID: " );
order_id = stdin.readLine();
USResPurchaseCC usResPurchaseCC = new USResPurchaseCC(data_key, order_id, amount, crypt);

/************************ Optional Variables ******************************/
// IMPORTANT: ideally the expiration date should be set within the iframe (This is done by
setting the optional expiry text box field to enabled). If not enabled, the expiration date
will need to be set by the setExpdate call below.
// usResPurchaseCC.setExpdate(exp_date);
// IMPORTANT note: ideally, if the CVD feature is used, the value should be set within the
iframe (This is done by setting the optional CVD text box field to enabled). If not enabled
within the iframe, the CVD value can be passed through the setCvdInfo call below.
// CvdInfo cvd = new CvdInfo ("1", "789");
```

Sample Processing the Payment - Hosted Tokenization

```
// resPurchaseCC.setCvdInfo(cvd);
usResPurchaseCC.setCustId(cust_id);
usResPurchaseCC.setCommcardInvoice(commcard_invoice);
usResPurchaseCC.setCommcardTaxAmount(commcard_tax_amount);
//usResPurchaseCC.setDynamicDescriptor(dynamic_descriptor);

/************************ Request Object ******************************/
USResolverHttpsPostRequest mpgReq =
new USResolverHttpsPostRequest(host, store_id, api_token, usResPurchaseCC);
/************************ Receipt Object ******************************/
try
{
USResolverReceipt resreceipt = mpgReq.getResolverReceipt();
ResolveData resdata = resreceipt.getResolveData();

System.out.println("DataKey = " + resreceipt.getDataKey());
System.out.println("ReceiptId = " + resreceipt.getReceiptId());
System.out.println("ReferenceNum = " + resreceipt.getReferenceNum());
System.out.println("ResponseCode = " + resreceipt.getResponseCode());
System.out.println("AuthCode = " + resreceipt.getAuthCode());
System.out.println("Message = " + resreceipt.getMessage());
System.out.println("TransDate = " + resreceipt.getTransDate());
System.out.println("TransTime = " + resreceipt.getTransTime());
System.out.println("TransType = " + resreceipt.getTransType());
System.out.println("Complete = " + resreceipt.getComplete());
System.out.println("TransAmount = " + resreceipt.getTransAmount());
System.out.println("CardType = " + resreceipt.getCardType());
System.out.println("TxnNumber = " + resreceipt.getTxnNumber());
System.out.println("TimedOut = " + resreceipt.getTimedOut());
System.out.println("ResSuccess = " + resreceipt.getResSuccess());
System.out.println("PaymentType = " + resreceipt.getPaymentType() + "\n");

//Contents of ResolveData
System.out.println("Cust ID = " + resdata.getResCustId());
System.out.println("Phone = " + resdata.getResPhone());
System.out.println("Email = " + resdata.getResEmail());
System.out.println("Note = " + resdata.getResNote());
System.out.println("MaskedPan = " + resdata.getResMaskedPan());
System.out.println("Exp Date = " + resdata.getResExpDate());
System.out.println("Crypt Type = " + resdata.getResCryptType());
System.out.println("Avs Street Number = " + resdata.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + resdata.getResAvsStreetName());
System.out.println("Avs Zipcode = " + resdata.getResAvsZipcode());

}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResPurchaseCC
```

For more details on the Vault API please download the API and Integration Guide at https://developer-.moneris.com/.

5 Testing a Hosted Payment Solution

5.1 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 available 24/7; 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, user IDs, and Hosted Tokenization configurations 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/configurations that you created.

Using the logins in Hosted Payment Page Configuration Tool , you can create your own Hosted Tokenization Configuration ID and Token. You can use these to send transactions to our test environment and configure your Hosted Tokenization. Your Configuration ID and Token will be valid for 30 days. You may test as often as required.

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 transactions based on the penny value of the amount field.

EXAMPLE: a 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 $10.00. 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 for download at https://developer.moneris.com

When testing you may use the following test credit card numbers with any future expiry date.

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

Table 1 Test Card Numbers

Card Plan	Card Number
MasterCard	5454545454545454
Visa	4242424242424242 or 4005554444444403
Amex	373599005095005
Diners	36462462742008

Table 2 INTERAC® Online Payment Test Card Numbers

Card Plan	Card Number
INTERAC Online Track2	3728024906540591206=01121122334455000 5268051119993326=01121122334455000000 453781122255=011211223344550000000000

NOTE:
When testing INTERAC® Online Payment you will be forwarded to the INTERAC® Online Payment merchant testing tool. A screen will appear where certain fields need to be completed.

For an approved response you will need to enter the following data in to the fields, do not alter any of the other fields:

IDEBIT_TRACK2:3728024906540591206=01121122334455000

IDEBIT_ISSNAME:RBC

IDEBIT_ISSCONF:123456

For a declined response leave the fields blank.

Click **Post to Merchant**. Do **not** click **Validate Data** — it will return validation errors.

When testing ACH transactions you may use the following test bank account details:

Table 3 Test Account Details - ACH

Financial Institution	Routing Number	Account Number	Check Number
FEDERAL RESERVE BANK	011000015	Any number between 5-22 digits	Any number

Gift Card Test Card Numbers

For Gift Card test credentials please contact our Integration Support team at onlinepay-ments@moneris.com.

5.2 What Information Will I Get As a Response to My Transaction Request?

- 5.2.1 Response Fields for Transaction Request
- 5.2.2 Special Error Codes for Hosted Solutions

For each transaction you will receive a response message. The fields that will be included in the response are indicated in the table below.

The Receipt can be handled in two ways depending on how the "Response Method" has been con-figured.

1. Moneris Gateway can generate a receipt on your behalf and present it to the client. The receipt will be relatively generic in appearance and will be based on the settings from the Hosted Token-ization Configuration in the Merchant Resource Center. Please refer to What Do I Need to Include in the Receipt? (see page 1) to configure the receipt.
2. The receipt values will be sent back to the URL specified in the Hosted Tokenization Configuration settings from the Merchant Resource Center. You can then create a custom receipt or use it to initiate a secondary process. These values can be passed back appended to the URL in a query string format or as an HTTP POST.

5.2.1 Response Fields for Transaction Request

Table 1 Response Fields - Transaction Request

Variable name	Size/Type	Description
response_order_id	50-character alphanumeric	order_id specified in request or gen-erated by Hosted Tokenization
response_code	3-character alphanumeric	Transaction Response Code

Variable name	Size/Type	Description
		< 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
date_stamp	yyyy-mm-dd	Processing host date stamp
time_stamp	##:##:##	Processing host time stamp
bank_approval_code	8-character alphanumeric	Authorization code returned from the issuing institution
result	1-character numeric	1 = approved , 0 = declined, incomplete
trans_name	alphanumeric	Type of transaction that was performed purchase: cardholder was billed immediately preauth: funds were locked on the card – a capture will need to be performed to have the funds deposited into merchant's account (see Merchant Resource Centre User's Guide). A PreAuth transaction **must** be reversed if it is not to be captured. To reverse the full amount of the PreAuth, please use the Capture transaction with a dollar amount of "0.00". achdebit: bank account information is collected and funds are debited from this account cavv_purchase: similar to purchase but a VbV/MCSC authentication attempt was made. cavv_preauth: similar to preauth but a

Variable name	Size/Type	Description
		VbV/MCSC authentication attempt was made. idebit_purchase: similar to purchase but the transaction was performed using INTERAC® Online Payment
cardholder	40-character alphanumeric	Cardholder's name
charge_total	9-character decimal. Up to 7-character numeric + 2-character numeric after the decimal point **EXAMPLE:** 1234567.89	Amount of the transaction

card	2-character alphanumeric	Credit Card Type
		
		M = Mastercard
		
		V = Visa
		
		AX = American Express
		
		DC = Diners Card
		
		NO = Novus / Discover
		
		C = JCB
		
		SE = Sears
		
		P = INTERAC® Online Payment
		
		CQ = ACH (Online Check)

f4l4	####***#### ***####	First 4 and last 4 digits of the card # Last 4 digits of the bank account number (ACH/Online Check)
exp_month	2-character numeric	2 digit month (ex. 01, 02...). Will return expiry month entered on the Hosted Tokenization (Credit Card only)
exp_year	2-character numeric	2 digit year (ex. 01, 02...). Will return

Variable name	Size/Type	Description
		expiry year entered on the Hosted Tokenization (Credit Card only)
message	100-character alphanumeric	Response description returned from issuing institution or from Moneris Gateway if there is a system error.
CfStatus	2-character alphanumeric	Indicates the status of the merchant and convenience fee transactions. The CfStatus field provides details about the transaction behavior and should be referenced when contacting Moneris Customer Support. Possible values are: 1 or 1F = Completed 1st purchase transaction 2 or 2F = Completed 2nd purchase transaction 3 = Completed void transaction 4A or 4D = Completed refund transaction 7 or 7F = Completed merchant independent refund transaction 8 or 8F = Completed merchant refund transaction 9 or 9F = Completed 1st void transaction 10 or 10F = Completed 2nd void transaction 11A or 11D = Completed refund transaction
iso_code	2-character numeric	ISO response code
bank_transaction_id	18-character numeric	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.

Variable name	Size/Type	Description
		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. **EXAMPLE:** The following illustrates the breakdown of this field where "660123450010690030” is the reference number returned in the message, "66012345" is the terminal id, "001" is the shift number, "069" is the batch number and "003" is the transaction number within the batch.
transactionKey	100-character alphanumeric (optional)	This is an encrypted string that is returned when using the transaction verification feature. There is no need to decrypt the string. It needs to be passed back to Moneris Gateway to verify the authenticity of the transaction. **NOTE:** This variable applies only when using transaction verification functionality.
ticket	alphanumeric	The value returned from the preload data request. **NOTE:** This variable applies only when using data preload functionality.
rvarn	optional	These extra variables can be sent in the request and will be echoed back in the

Variable name	Size/Type	Description
		response. These variables must begin with “rvar” and then contain any alpha-numeric string (i.e. rvar1, rvarname, rvarMyVariable). If they are not posted in the request, they will not be included in the response.
eci	1-character numeric	Electronic Commerce Indicator that was sent with the transaction. Possible values are:

Crypt Type	TVisa/MCSC Definitions
5	- Fully authenticated - There is a liability shift and the merchant is protected from chargebacks.
6	- VbV/MCSC has been attempted - VbV -There is a liability shift and the merchant is protected from chargebacks -MCSV –No liability shift and the merchant is not protected from chargebacks.
7	- Non-VbV/MCSC transaction - Merchant is no longer protected from chargebacks

Variable name	Size/Type	Description
txn_num	20-character alphanumeric	Gateway Transaction identifier. This value is required if merchant decides to send automated captures, voids or refunds through an API.

Variable name	Size/Type	Description
recur_result	true	Indicates the Recurring Billing result. true: The Recurring Billing transaction was successfully registered. Any response other than "true" indicates that the recurring billing transaction was not properly registered.
avs_response_code	1-character alphanumeric	Indicates the address verification result. Refer to Appendix A Transaction Request Examples for further details. To test AVS you must create a configuration in "store5" and use that configuration for testing.
cvd_response_code	1-character alphanumeric	Indicates the CVD validation result. Refer to Appendix A Transaction Request Examples for further details. To test CVD you must create a configuration in "store5" and use that configuration for testing.
cavv_result_code	1-character alphanumeric	The Cardholder Authentication Verification Value (CAVV) is a value that allows VisaNet to validate the integrity of the VbV transaction data. These values are passed back from the issuer to the merchant after the VbV/SecureCode authentication has taken place. **EXAMPLE** If the eci returned is a "6" and the result code is a "B", it becomes liable for chargeback. Please see CAVV Result Codes - Verified by Visa (see page 1) for the CAVV result codes table
is_visa_debit	boolean	A value of 'true' or 'false' is sent back which indicates if the card provided by the cardholder was a Visa Debit card.

Variable name	Size/Type	Description

Table 2 Response Fields - INTERAC® Online Payment

Variable name	Size/Type	Description
Trans_name	alphanumeric	Type of transaction that was performed idebit_purchase: similar to purchase but the transaction was performed using INTERAC® Online Payment
ISSNAME	1-30 characters alphanumeric	Returned for an INTERAC® Online Payment transaction. This field identifies the name of the card issuer. This data must be displayed on a receipt.
INVOICE	1-20 characters alphanumeric	Returned for an INTERAC® Online Payment transaction. This field contains the invoice number used to identify the transaction. This data must be displayed on the receipt.
ISSCONF	1-15-character alphanumeric	Returned for an INTERAC® Online Payment transaction. This field is the confirmation number returned by the issuing bank. This data must be displayed on the receipt.

Table 3 Response Fields - Gift Card Transactions

Variable name	Size/Type	Description
gift_charge_total	9-character numeric	This is the total amount of the Purchase transaction. This must contain 3 digits with two penny values. The minimum value passed can be 0.01 and the maximum 9999999.99

Variable name	Size/Type	Description
rem_balance	9-character numeric	This is the remaining balance on the card after Deactivation. The balance will be in pennies.
display_text	82-character alphanumeric	This is the remaining balance on the card after Deactivation. The balance will be in pennies.
receipt_text	122-character alphanumeric	This is a message that, if present, is to be printed on the receipt
voucher_text	255-character alphanumeric	If the VoucherType field is non-zero, the text from this field should be printed in the body of the voucher.
ref_num	10-character numeric	This is the unique number that was assigned by the Moneris system to identify the trans-action. The maximum value of this parameter is 0xFFFFFFFF (4294967295). The host can not return reference numbers greater than this value. If this field is present, it is to be included on the receipt.
terminal_id	8-character numeric	Identifies the Terminal Iden-tifier which was used to pro-cess the transaction.
txn_num	30-character alphanumeric	Gateway Transaction identifier. This value is required if mer-chant decides to send auto-mated void/refund through an API.

NOTE: Multiple gift cards may be used to cover the full amount of the transaction. If two gift cards are submitted for processing, then there will be two sets of the above <gift_card> response fields within the response XML.

Table 4 Response Fields - Convenience Fee Transactions

Variable name	Size/Type	Description
convenience_fee	9-character decimal. Up to 7-character numeric + 2-character numeric after the decimal point **EXAMPLE:** 1234567.89	Charge the convenience fee amount. Please note the 'convenience_fee' must be less than the 'charge_total'.
cf_fee_rate	9-character decimal	The convenience fee rate that has been defined on the merchant's profile. For example: 1.00 – a fixed amount or 10.0 - a percentage amount
cf_fee_type	AMT / PCT	The type of convenience fee that has been defined on the merchant's profile. Available options are: AMT – fixed amount PCT – percentage
cf_success	true/false	Indicates whether the Convenience Fee transaction processed successfully.

Table 5 Response Fields - Transaction Risk Management Tool Transactions

Variable name	Size/Type	Description
risk_policy_score		The sum of all the risks weights from triggered rules within the selected policy in the range [-100…+100].
risk_request_result		success – ThreatMetrix was able to process the request successfully fail_access – ThreatMetrix was

Variable name	Size/Type	Description
		unable to process the request due to API verification failing fail_verification – API query limit reached fail_incomplete – ThreatMetrix was unable to process the request due to incomplete or incorrect input data fail_internal_error – ThreatMetrix encountered an error while processing the request fail_temporarily_unavailable – the request fail because the service is temporarily unavailable fail_invalid_email_address – the format of the supplied email address was invalid fail_invalid_telephone_number – the format of the supplied telephone number was invalid fail_invalid_device_id – the format of the supplied device_id was invalid fail_invalid_ip_address_parameter – the format of a supplied ip_address parameter was invalid
risk_reason_code		The codes of the rules verified from the selected policy that have triggered. Each rule code is returned as a separate name/value pair.
risk_reason_name		The names of rules verified from the selected policy that have triggered. Each rule name is returned as a separate name/value pair.
risk_reason_message_en		An English message description

Variable name	Size/Type	Description
		of the rule returned.
risk_reason_message_fr		A French message description of the rule returned.

Table 6 Response Fields - Loyalty Card Transactions

Variable name	Type	Description
request_amount	9-character decimal. Up to 7-character numeric + 2-character numeric after the decimal point **EXAMPLE:** 1234567.89	Identifies the amount for which loyalty points are to be awarded. This amount may be equal to or less than the total amount of the transaction.
transaction_points	9-character numeric	Amount processed on this loyalty card transaction. This value will be displayed in the number of points.
transaction_amount	9-character decimal. Up to 7-character numeric + 2-character numeric after the decimal point **EXAMPLE:** 1234567.89	Amount processed on this loyalty card transaction. This value will be displayed in the number of points.
current_balance	9-character numeric	This is the current balance for the card in points. If this field is present, it is to be printed on the receipt. If this field is not present, no balance information is to be printed on the receipt
lifetime_balance	9-character numeric	This is the lifetime balance for the card in points. If this field is present, it is to be printed on the receipt. If this field is not

Variable name	Type	Description
		present, no balance information is to be printed on the receipt.

Table 7 Response Fields - Hosted Vault Transactions

Variable name	Type	Description
data_key	max 50-character alphanumeric	The unique key to identify the client. This is the ID that will be used for subsequent transactions for the account, such as an update.
payment_type	alphanumeric	This identifies what type of payment was registered. Possible values are:
res_success	4-character alphanumeric	True: Card registered False: Card failed registration Null: Incomplete registration

NOTE: To determine if a transaction is approved, the response_code will have a value of less than 50. If it is declined the response_code will be 50 or greater. A value of NULL means the transaction was incomplete.

5.2.2 Special Error Codes for Hosted Solutions

The Hosted Tokenization is designed to generate special error codes when certain data is incorrect and/or the transaction couldn't be processed. The table below contains the information regarding the error codes. Each error will be accompanied by a message describing the problem.

Table 1 Special Error Codes

Code	Message/Description
914	Transaction cancelled by cardholder – The response code indicates that the cardholder pressed the <cancel> transaction button – This response is only returned if the enhanced cancel button functionality is enabled within the Hosted Tokenization configuration.
991	Invalid referrer URL - <referrer url> – If the Hosted Tokenization solution is configured to check the referring URL and it is incorrect this error will occur. The source URL will be included in the error. Please refer to the “Security Features” portion of the Hosted Tokenization configuration for a list of all Allowed Referring URLs.
992	VbV / Secure Code authentication failed – This error will occur if your merchant account is configured for VbV/MCSC and the cardholder failed to enter the proper PIN during the authentication process.
993	Data error - unable to store data – This error will occur if too much request data was passed in the transaction request or if the database failed to store the request. This may occur if unsupported characters were included in one of the posted fields.
N/A	Invalid store credentials – There is no code generated and a blank page is loaded with the above information. The ps_store_id and/or hpp_key did not match an existing store.
N/A	Card Issuer returned corrupt data. Unable to proceed with the transaction. Please return to the site where you initiated the transaction and try again. Your card has not been charged. – There is no code generated and a blank page is loaded with the above information. This error will occur if the cardholder’s issuing bank did not return the correct data in the VbV/MCSC authentication process.

Table 2 Error Codes - Convenience Fee Responses

Code	Message/Description
973	Unable to locate merchant CF details

Code	Message/Description
977	Invalid amount
978	Failed CF transaction
984	Data error: (optional: field name)
987	Invalid transaction
Null	Error: Malformed XML

6 Moving to Production

Once you have completed the necessary steps of creating a profile for your solution, configuring the solution profile, developing and testing the solution, you are ready to move your solution into production.

6.1 How Do I Activate My Store?

Once you have received your activation letter/fax go to https://www3.moneris.com/connect/en/activate/index.php 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**. Once this is confirmed you will need to create an administrator account that you will use to log into the Merchant Resource Center to access and administer your Moneris Gateway store.

NOTE: The API TOKEN that you receive during Activation is NOT the token that you require for the Hosted Tokenization request.

6.2 How Do I Configure My Store for Production?

Once you have activated your store, the next step is to point your store to the production host.

To point your store to the production host:

1. In your HTML FORM POST, change the <FORM METHOD="POST" ACTION=https://esqa.moneris.com/HPPDP/index.php> to contain the production URL: <FORM METHOD="POST" ACTION=https://www3.moneris.com/HPPDP/index.php>.
2. Change the ps_store_id and hpp_key to reflect your production store configuration.

Once you are in production you will access the Merchant Resource Center at https://www3.moneris.com/mpg. You can use the store administrator ID you created during the activation process and then create additional users as needed.

For INTERAC® Online Payment Solution:

Third-party Service/Shopping-cart Provider

In your product documentation, please ensure that the clients are properly instructed to create the Production HPP configuration as outlined above. They should also be instructed to provide Moneris Solutions with screen-shots of their check-out process showing examples of approved and declined transactions using the INTERAC® Online Payment service and provide the completed Merchant Checklist of Appendix B. Detailed descriptions of the requirements for the checklist can be found in the INTERAC

Online Merchant Guidelines document. Once completed, they can be faxed or emailed to the Moneris Gateway Integration Support group for review.

When you are ready to move into production please contact the Integration Support Team at eproduct-s@moneris.com.