# Moneris MPI – Verified by Visa / MasterCard SecureCode

PHP API - v1.2.7

| **Revision Number** | **Date** | **Description** |
|---|---|---|
| **V1.2.6** | November 13, 2013 | -Document edited for coherence<br>-Section 2. Verified by Visa and MasterCard Secure Code<br>- Merged Verified By Visa and MasterCard Secure Code and changed description.<br>-Section 3. System and Skill requirement<br>- Updated PA-DSS description<br>-Updated download link in various locations to https://developer.moneris.com<br>-Section 4. Transaction Flow.<br>- Updated content<br>-Section 5. How do I send Transactions?<br>- Updated sample code<br>-Section 6. What does VbV/MCSC Response mean?<br>- Updated Response table<br>-Section 7 How do I Test My Solution?<br>- Updated Test Card number table<br>-Section 9. How do I get Help?<br>- Updated support email to eproducts@moneris.com<br>Appendix B: Definition of Response Fields<br>- Updated response fields description |

**Table of Contents**

## ****** PLEASE READ CAREFULLY******

**You have a responsibility to send only Visa or MasterCard to the Moneris MPI. Under no circumstances should ANY other card type be sent to the VBV/MCSC MPI.**

## 1. About this Documentation

This documentation contains the basic information for using the PHP API for sending a Verified by Visa or MasterCard SecureCode request to the Moneris MPI. In particular it describes the format for sending the requests and handling the corresponding responses you will receive.

To help prevent fraudulent activity on online transactions it is highly recommended that you also implement the eSELECTplus eFraud features which consist of AVS and CVD.

*Address Verification Service (AVS)* – Verifies the cardholder's billing address information.
*Card validation Digit (CVD)* – Validates that cardholder has a genuine credit card in their possession during the transaction.

## 2. Verified by Visa and MasterCard Secure Code

Verified by Visa (VbV) and MasterCard Secure Code (MCSC) are programs based on the 3-D Secure Protocol to improve the security of online transactions. These programs involve authentication of the cardholder during an online e-Commerce transaction. Authentication is based on the issuer's selected method of authentication. The following are examples of authentication method; risk-based authentication, dynamic passwords, and static passwords. Some of the benefits of these programs are reduced risk of fraudulent transactions and protection against chargebacks for certain fraudulent transactions. Enrollment is required in order to participate in the VbV and Secure Code programs. Merchants must contact the Moneris Sales/Support Helpdesk to enroll into these programs.

## 3. System and Skill Requirements

In order to use the PHP API your system will need to have the following:

1. PHP 4 or later
2. Port 443, 9443 (for testing) open
3. OpenSSL
4. cURL - PHP interface - this can be downloaded from http://curl.haxx.se/download.html

As well, you will need to have the following knowledge and/or skill set:

1. PHP

**Note:**

It is important to note that all Merchants and Service Providers that store, process, or transmit cardholder data must comply with PCI DSS and the Card Association Compliance Programs. However, validation 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 2.0 may result in a Merchant being subject to fines, fees or assessments and/or termination of processing services. Non-compliant solutions may prevent merchants boarding with Moneris Solutions.

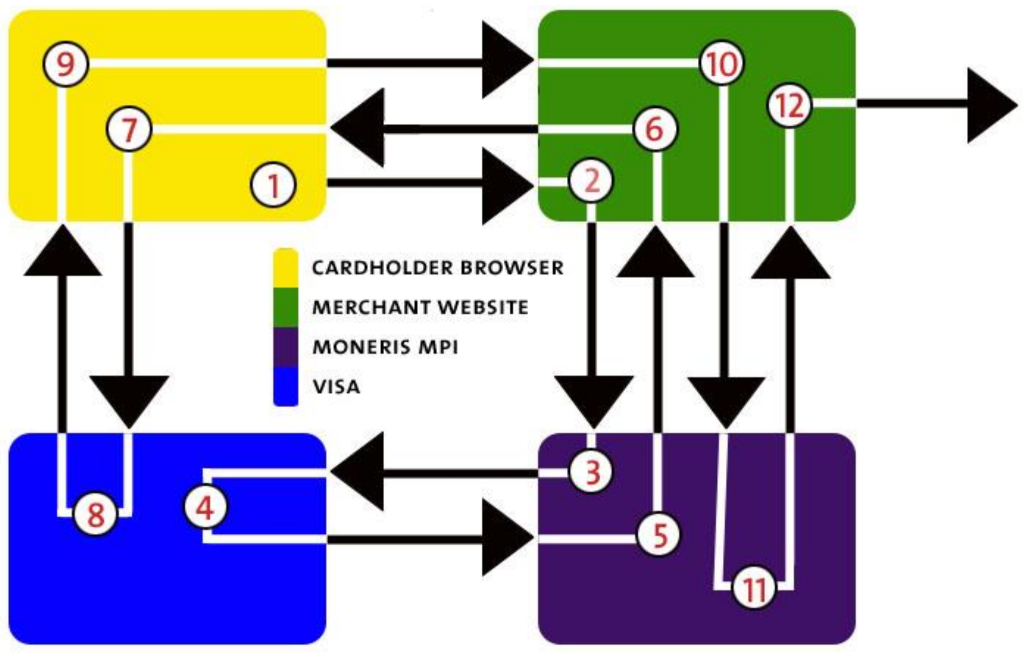
As a Moneris Solutions client or partner using this method of integration, your solution must demonstrate compliance to the Payment Card Industry Data Security Standard (PCI DSS) and/or the Payment Application Data Security Standard (PA DSS) 2.0. These standards are designed to help the cardholders and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

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

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

## 4. Transaction Flow

Below is a diagram with explanations about the flow of a VBV/MCSC transaction.



1. Cardholder enters their credit card number and submits their transaction information to the merchant.

2. Upon receiving the transaction request the merchant calls the Moneris MPI API and passes a TXN type request.

3. The Moneris MPI receives the request and authenticates the merchant and sends the transaction information to Visa/MasterCard.

4. Visa/MasterCard verifies if the card is enrolled and returns a URL for the issuer.

5. Moneris MPI receives the response from Visa or /MasterCard and forwards the information to the merchant.

6. The Moneris MPI API installed at the merchant receives the response from the Moneris MPI. If the card is enrolled the response (getMpiMessage()) would be “Y” and the merchant makes a call to the API, which opens a popup/inline window in the Cardholder browser. If the response was an “N” for not enrolled a transaction could be sent to the processor identifying it as VBV/MCSC attempted with ECI value of 6. If the response was “U” (Unable to Authenticate) or the response times out, the transaction can be sent to the processor with ECI value of 7; however, the merchant in this case would be liable to a chargeback. Otherwise merchant can chose not to continue with the transaction.

7. The cardholder browser uses the URL returned from Visa/MasterCard via the merchant to communicate directly to the bank. The contents of the popup are loaded and the cardholder enters their pin.

8. The information is submitted to the bank and authenticated. A response is then returned to the client browser.

9. The client browser receives the response from the bank and forwards it to the merchant.

10. The merchant receives the response information from the cardholder browser and makes a call to the Moneris MPI API and passes an ACS request type.

11. Moneris MPI receives the ACS request and authenticates the information. The Moneris MPI then provides a CAVV value (getMpiCavv()) to the merchant. If the getMpiSuccess() of the response is “true”, the merchant may proceed with the cavv purchase or cavv preauth. If the response is “false” and the getMpiMessage() is “N”, the transaction must be cancelled as cardholder failed to authenticate. If the response is “false” and the getMpiMessage is “U” or the response times out, the transaction can be processed as a normal purchase or PreAuth; however in this case the merchant assumes liability to a chargeback.

12. The merchant retrieves the CAVV value and formats a cavv purchase or a cavv preauth request using the method that is normally used. As part of this transaction method the merchant must pass the CAVV value. For more information on sending cavv-purchase and cavv-preauth, please refer to the main API available from our Developer Portal: https://developer.moneris.com.

## 5. How do I send Transactions?

### A. The TXN Request (Step 2 of Transaction Flow)

The txn request sends the initial transaction data to the Moneris MPI to verify if the card is enrolled. Below is the ASP code that is used to make a request to the Moneris MPI API. This code can be found from the mpistore.php sample included in the download.

```
$HTTP_ACCEPT = getenv("HTTP_ACCEPT");
$HTTP_USER_AGENT = getenv("HTTP_USER_AGENT");

//these are form variable gotten after cardholder hits buy button on merchant site
//(purchase_amount,expiry)

$txnArray=array(type=>'txn',
                    xid=>$xid,
                    amount=>$purchase_amount,
                    pan=>$pan,
                    expdate=>$expiry,
                    MD=>  "xid=" . $xid   //MD is merchant data that can be passed along
                              ."&amp;expiry=".$expiry
                              ."&amp;amount=" .$purchase_amount,
                    merchantUrl=>$merchUrl,
                    accept =>$HTTP_ACCEPT,
                    userAgent =>$HTTP_USER_AGENT
                    );

$mpiTxn = new MpiTransaction($txnArray);

$mpiRequest = new MpiRequest($mpiTxn);

$mpiHttpPost  = new MpiHttpsPost($storeid,$apitoken,$mpiRequest);
```

### B. The TXN Response & Creating the Popup (Step 6 of Transaction Flow)

The txn request will return a response with several values. The "getMpiMessage" will contain "Y", "U", "N". If the message is "N" the purchase or preauth can be sent as a crypt type 6 – attempted authentication. If the message is "Y" then a call to the API to create the VBV form is made. Finally, if the message is "U" the merchant can send the transaction with crypt_type 7; however, the merchant would be liable for chargebacks. "U" is returned for non-participating cards, such as corporate cards.

```
$mpiResponse = $mpiHttpPost->getMpiResponse();

if($mpiResponse->getMpiMessage() == 'Y')
{
     $vbvInLineForm = $mpiResponse->getMpiInLineForm();
     print "$vbvInLineForm\n";
}
elseif ($mpiResponse->getMpiMessage() == 'N')
{
     //Send transaction using the mpg API in this case merchant
     //Use $crypt_type='6';
}
else //corporate cards, unable to authenticate or times out (e.g. MPI is down)
     {
     //optional to send transaction using the mpg API in this case merchant assumes
     //liability, use $crypt_type='7';
}
```

### C. The ACS Request (Step 10 of Transaction Flow)

After the cardholder completes the authentication the browser will return a response to the URL specified in the TermURL field that was provided in the original request. This will contain a PARes as well as a success field. Once the PARes is received it must be passed to the Moneris MPI API as a type ACS.

```
$txnArray = array ( type=>'acs',
                    PaRes=>$PaRes,
                    MD=>$MD
                    );

$mpiTxn = new MpiTransaction($txnArray);

$mpiRequest = new MpiRequest($mpiTxn);

$mpiHttpPost  = new MpiHttpsPost($storeid,$apitoken,$mpiRequest);
```

## D. The ACS Response and forming a transaction (Step 11 of Transaction Flow)

The ACS response will contain the CAVV value. This value needs to be passed to the transaction engine along with the rest of the purchase or preauth request. Please see the documentation provided by your payment solution. Outlined below is how to send a transaction to eSELECTplus – requires the eSELECTplus API installed as well. Please refer to the eSELECTplus PHP API doc for response definitions.

```
$mpiResponse=$mpiHttpPost->getMpiResponse();

parse_str($MD);     //this function will parse MD field as if it were a query string
                    //and bring the resultant variables into this scope
if( strcmp($mpiResponse->getMpiSuccess(),"true") == 0 )
{
      //send transaction to host using CAVV purchase or CAVV preauth, refer to sample code
      //for eSELECTplus.  Call getMpiCavv() to obtain the CAVV value
      //If you are using preauth/capture model, be sure to call getMpiMessage() so that the
      //value can be stored and used in the capture transaction after on to protect your
      //chargeback liability.  (e.g. getMpiMessage()= A = crypt type of 6 for follow on
      //transaction and getMpiMessage() = Y = crypt type of 5 for follow on transaction.

      require("mpgClasses.php");

      $orderid =sprintf("%'920d", rand());

      $cavv = $mpiResponse->getMpiCavv();

      $txnArray=array(type=>'cavv_purchase',
                                 order_id=> $orderid,
                                 amount=>$amount,
                                 pan=>$pan,
                                 expdate=>$expiry,
                                 cavv=>$cavv,
                                 );

      $mpgTxn = new mpgTransaction($txnArray);

      $mpgRequest = new mpgRequest($mpgTxn);

      $mpgHttpPost  = new mpgHttpsPost($storeid,$apiToken,$mpgRequest);

      $mpgResponse =$mpgHttpPost->getMpgResponse();    }
      print "<br>Message = ".$mpiResponse->getMpiMessage();
}
```

```
        else {
                if($mpiResponse->getMpiMessage() == 'N')
                {
                        //Do not send transaction as the cardholder failed authentication.
                }
                else
                {
                        //Optional to send transaction using the mpg API in this case merchant assumes
                        //liability.
                }
```

## 6. What does My VbV/MCSC Response Mean?

For each transaction, a crypt type will be sent to identify whether it is a VbV or MCSC authenticated transaction. Below are the tables defining the different possible crypt types as well as the possible VARes and PARes responses.

| Crypt Type | Visa Definition | MasterCard Definition |
|---|---|---|
| 5 | - Fully authenticated<br>- There is a liability shift and the merchant is protected from chargebacks | - Fully authenticated<br>- There is a liability shift and the merchant is protected from chargebacks. |
| 6 | - VbV has been attempted<br>- There is a liability shift and the Merchant is protected from certain chargebacks on fraudulent transactions | - MCSC has been attempted<br>- There is a liability shift and the Merchant is protected from certain chargebacks on fraudulent transactions |
| 7 | - Non-VbV transaction<br>- No liability shift<br>- Merchant is not protected from chargebacks | - Non-MCSC transaction<br>- No liability shift<br>- Merchant is not protected from chargebacks |

| VERes Response | Response Definition |
|---|---|
| N | The card/issuer is not enrolled. This should be sent as a normal Purchase/PreAuth transaction with a crypt type of 6. |
| U | The card type is not participating in VbV or MCSC: this could be corporate or other card plans that Visa or MasterCard excludes. Can proceed with a regular transaction with a crypt type of 7 or cancel the transaction. |
| Y | The card is enrolled. Proceed to create the VbV/MCSC inline window for cardholder authentication. Proceed to PARes for crypt type. |

| PARes Response | Response Definition |
|---|---|
| A | Attempted to verify PIN and will receive a CAVV. This should now be sent as a cavv_purchase/cavv_preAuth which will return a crypt type of 6. |
| Y | Fully authenticated and will receive a CAVV. This should now be sent as a cavv_purchase/cavv_preAuth which will return a crypt type of 5. |
| N | Failed to authenticate, no CAVV will be returned. Transaction should be cancelled; the merchant may proceed with a crypt type of 7 though strongly discouraged. |

| Step 1: VERes<br>Is the cardholder/issuer enrolled? | Step 2: PARes<br>VbV/MCSC InLine window response | Step 3: Transaction<br>Are you protected? |
|---|---|---|
| Y | Y | Send a CAVV transaction |
| Y | N | Cancel transaction. Authentication failed or high-risk transaction. |
| Y | A | Send a CAVV transaction |
| U | n/a | Send a regular transaction with a crypt type of 7 |
| N | n/a | Send a regular transaction with a crypt type of 6 |

## 7. How Do I Test My Solution?

When testing your implementation of the Moneris MPI you can use the VISA / MasterCard PIT (production integration testing) environment to test. When testing, the process is a little different in that when the inLine window is generated it will not contain any input boxes but rather a window of data and a "Submit" button. When you hit "Submit" it will load the response in the window and not in the main as it will in production.

The test environment is generally available 7x24, however since it is a test environment we cannot guarantee 100% availability. Also, please be aware that other merchants are using the test environment so in the Merchant Resource Centre you may see transactions and user IDs that you did not create. As a courtesy to others that are testing we ask that when you are processing Refunds, changing passwords and/or trying other functions that you use only the transactions/users that you created.

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

**Test Card Numbers**

| Card Number | VERes | PARes | Action |
|---|---|---|---|
| 4012001037141112 | Y | true | TXN – call function to create inLine window<br>ACS – Send CAVV to eSELECTplus using "cavv_purchase" or "cavv_preauth" |
| 4012001038488884 | U | N/A | Send transaction to eSELECTplus using regular "purchase" or "preauth". Set crypt_type = 7 |
| 4012001038443335 | N | N/A | Send transaction to eSELECTplus using regular "purchase" or "preauth". Set crypt_type = 6 |
| 4242424242424242 | Y | true | TXN – call function to create inLine window<br>ACS – Send CAVV to eSELECTplus using "cavv_purchase" or "cavv_preauth" |
| 4012001037461114 | Y | false | Card failed to authenticate, merchant may chose to send transaction or decline transaction. If transaction is sent, use crypt type = 7 |

VERes – the result U, Y or N is obtained by using: `getMpiMessage()`
PARes – the result "true" or "false" is obtained by using: `getMpiSuccess()`

---

**NOTE** For a sample of a "purchase", "preauth", "cavv_purchase" and "cavv_preauth" transaction, please refer to the complete PHP API Integration Guide found at: https://developer.moneris.com

---

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

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

*The test environment will approve and decline transactions based on the penny value of the amount field.*
For 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 $1000.00. This limit does not exist in the production environment. For a list of all current test environment responses for various penny values, please see the Test Environment Penny Response table as well as the Test Environment eFraud Response table, available at https://developer.moneris.com.

---

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

---

**cURL CA Root Certificate File:**

The default installation of PHP/cURL does not include the cURL CA root certificate file. In order for the eSelectPlus PHP API to connect to the eSelectPlus gateway during transaction processing, the 'mpgclasses.php' file that's included with the PHP API package needs to be modified to include a path to the CA root certificate file. Follow the instructions below to set this up.

1) If cURL was not installed separately from your PHP installation, libcurl is included in your PHP installation. You will need to download the 'cacert.pem' file from 'http://curl.haxx.se/docs/caextract.html' and save it to the necessary directory. Once downloaded, rename the file to 'curl-ca-bundle.crt' (e.g. 'C:\path\to\curl-ca-bundle.crt'). If cURL was installed separately from PHP, you may need to determine the path to the cURL CA root certificate bundle on your system (e.g. 'C:\path\to\curl-ca-bundle.crt').

2) Insert the code below into the 'mpgclasses.php' file as part of the cURL option setting, at approximately line 73 below the line 'curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, TRUE);'

```
curl_setopt($ch, CURLOPT_CAINFO, 'C:\path\to\curl-ca-bundle.crt');
```

For more information regarding the *CURLOPT_SSL_VERIFYPEER* option, please refer to your PHP manual.

## 8. How Do I Configure My Store For Production?

Once you have completed you testing you are ready to point your store to the production host. You will need to edit the mpgClasses.php file and change the $Globals array as highlighted below in red. You will also need to change the store_id to reflect your production store ID as well the api_token must be changed to your production token to reflect the token that you received during activation.

| PRODUCTION | DEVELOPMENT |
|---|---|
| `var $Globals=array(`<br>`    ‘MONERIS_PROTOCOL’ => 'https',`<br>`    ‘MONERIS_HOST’ => ‘www3.moneris.com’,`<br>`    ‘MONERIS_PORT’ =>'443',`<br>`    ‘MONERIS_FILE’ => ‘/gateway2/servlet/MpgRequest',`<br>`    ‘API_VERSION’  =>'MPG Version 2.01',`<br>`    ‘CLIENT_TIMEOUT’ => '60'`<br>` );` | `var $Globals=array(`<br>`    ‘MONERIS_PROTOCOL’ => 'https',`<br>`    ‘MONERIS_HOST’ => ‘esqa.moneris.com’,`<br>`    ‘MONERIS_PORT’ =>'443',`<br>`    ‘MONERIS_FILE’ => ‘/gateway2/servlet/MpgRequest',`<br>`    ‘API_VERSION’  =>'MPG Version 2.01',`<br>`    ‘CLIENT_TIMEOUT’ => '60'`<br>` );` |

Once you are in production, you will access the Merchant Resource Centre 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 further information on how to use the Merchant Resource Centre please see the eSELECTplus Merchant Resource Centre User's Guide which is available for download at http://developer.moneris.com.

## 9. How Do I Get Help?

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

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

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

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

## 10. Appendix A: Definitions of Required Fields

| Required Fields | | |
|---|---|---|
| **Variable Name** | **Size/Type** | **Description** |
| store_id | 12 / an | A value that identifies your company when your send a transaction. |
| api_token | 20 / an | A unique key that when matched with your store_id creates a secure method of authenticating your store_id |
| xid | 20 / an | Must be 20 alpha numeric characters. This must be unique for every transaction attempt – this can also be used as your order_id when using eSELECTplus |
| amount | 9 / decimal | Amount of the transaction. This must contain at least 3 digits including two penny values. The minimum value passed can be 0.01 and the maximum 9999999.99 |
| pan | 20 / num | Credit Card Number - no spaces or dashes. Most credit card numbers today are 16 digits in length but some 13 digits are still accepted by some issuers. This field has been intentionally expanded to 20 digits in consideration for future expansion and/or potential support of private label card ranges. |
| expdate | 4 / num | Expiry Date - format **YYMM** no spaces or slashes.<br>PLEASE NOTE THAT THIS IS REVERSED FROM THE DATE DISPLAYED ON THE PHYSICAL CARD WHICH IS MMYY |
| MD | 1024 / an | This is information that you would like echoed back in the response |
| merchantUrl | | This is the URL to which you would like the MPI response sent to. |
| accept | | MIME types the browser accepts |
| userAgent | | The browser details |
| PaRes | | This is a value that is passed back to the API during the TXN and returned to the MPI when an ACS request is made. |

## 11. Appendix B: Definitions of Response Fields

| Response Fields | | |
|---|---|---|
| **Variable Name** | **Size/Type** | **Description** |
| Type | 99 / an | VERes or PARes or error defines what type of response you are receiving |
| success | true/false | Returns whether the attempt was successful or not |
| message | alpha | Will contain: |

| **Txn** | **Action** |
|---|---|
| Y | Create VBV verification form popup window. |
| N | Send purchase or preauth with crypt type 6 |
| U | Send purchase or preauth with crypt type 7 |

| **ACS** | **Action** |
|---|---|
| Y or A (getMpiSuccess()=true) | Proceed with cavv purchase or cavv preauth |
| N | Authentication failed or high-risk transaction. Recommended not to proceed with transaction. Depending on merchant's risk tolerant level and results from other methods of fraud detection system, transaction may proceed with crypt type of 7. |
| U or time out | Send purchase or preauth as crypt type 7 |

| | | |
|---|---|---|
| PARes | n/a | Variable Length – data that Visa/MasterCard passes and needs for authentication |
| TermUrl | 255 / an | The URL to which the PARes is returned |
| MD | 1024 / an | Merchant defined data that will be echoed back |
| ACSUrl | 255 / an | URL that will for the generated popup |
| cavv | 28 /an | Visa/MasterCard authentication data |