Moneris MPI
Verified by Visa / MasterCard SecureCode
Ruby API – v1.2.5

Moneris SOLUTIONS

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 Ruby 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. System and Skill Requirements

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

1. Web server with an SSL Certificate
2. Port 443 open
3. Ruby framework

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

1. Ruby Programming Language
2. Install a dll into the global assembly cache

Note:

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

As a Moneris Solutions client or partner using this method of integration, your solution must demonstrate compliance to the Payment Card Industry Data Security Standard (PCI DSS) and/or the Payment Application Data Security Standard (PA DSS). 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://esplusqa.moneris.com/connect/en/download/index.php to download the PCI-DSS Implementation Guide.

3. Verified by Visa

Verified by Visa (VbV) is a program initiated by Visa. Before approving a transaction eSELECTplus and the Bank that issues the Visa credit cards will attempt to authenticate the cardholder through the use of a password, similar to a debit PIN. When an authentication is attempted the merchant is protected from chargebacks.

If you have enrolled in Verified by Visa (VbV) with Moneris and eSELECTplus, please also refer to the *Java* VbV / SecureCode MPI document found at: https://esplusqa.moneris.com/connect/en/download/index.php

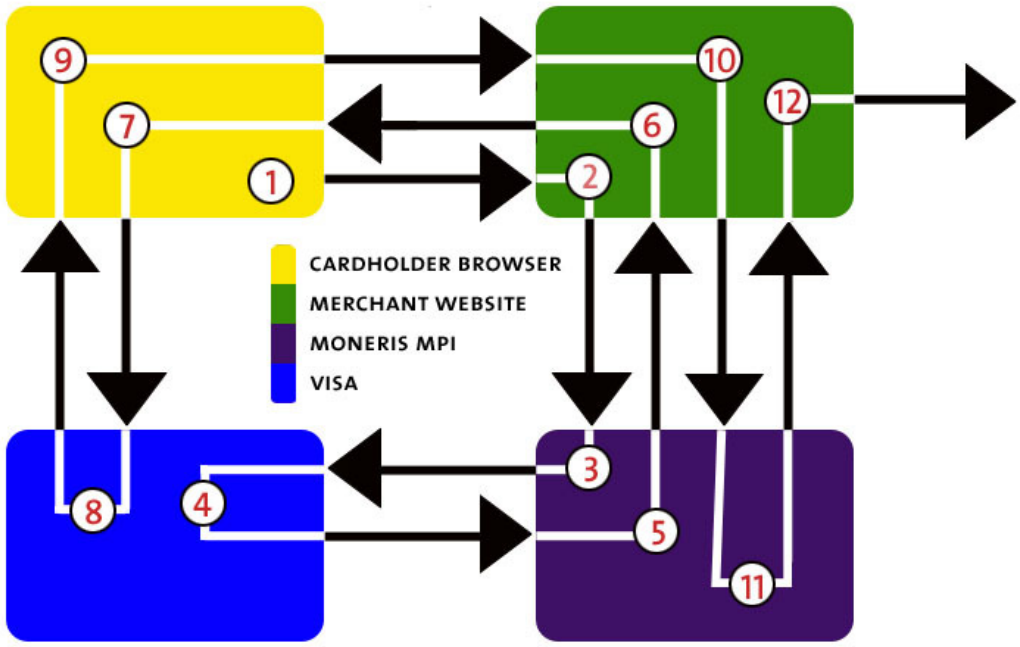
4. MasterCard SecureCode

MasterCard SecureCode (MCSC) is a new feature offered by MasterCard. Merchants who have enrolled in this program with Moneris and eSELECTplus will be able to offer their customers added protection against unauthorized credit card use, as well as protect themselves from fraud-related chargebacks. Cardholders that have applied for SecureCode with their issuing bank will be able to use this password similar to a debit PIN number for online transactions with participating online merchants.

Before approving a transaction, eSELECTplus and the Bank that issued the MasterCard will authenticate the cardholder through the use of this password. For merchants who have enrolled in SecureCode, please also refer to the Java VbV / SecureCode MPI document found at: https://esplusqa.moneris.com/connect/en/download/index.php

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. For sample code please refer to section 6.a.

3. The Moneris MPI receives the request, authenticates the merchant and based on the card type, sends the transaction information to Visa or MasterCard.

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

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 (mpiresponse.message()) 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 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. Please refer to section 6.b. for sample code and please refer to section 7 for possible liability scenarios.

7. The cardholder’s browser uses the URL returned from Visa/MasterCard via the merchant to communicate directly to the bank. The contents of the inline window 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's browser and makes a call to the Moneris MPI API and passes an ACS request type. For sample code please refer to section 6.c.

11. Moneris MPI receives the ACS request and authenticates the information. The Moneris MPI then provides a CAVV value (CavvPurchase) to the merchant. If the acsresponse.success of the response is "true", the merchant may proceed with the cavv purchase or cavv preauth. If the response is "false" and the mpiresponse.message() is "N", the transaction must be cancelled as cardholder failed to authenticate. If the response is "false" and the mpiresponse.message() 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. As part of this transaction method the merchant must pass the CAVV value. For a sample of a VbV/MCSC Purchase transaction, please refer to the complete JAVA API Integration Guide available at: https://esplusqa.moneris.com/connect/en/download/index.php

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 code that is used to make a request to the Moneris MPI API. This code can be found in the mpistore.rb sample included in the download.

```
r.txn.xid = xid.to_s
r.txn.amount = "1.00"
r.txn.pan = "4242424242424242"
r.txn.expdate = "1111"

r.txn.md = xid.to_s + "&" + r.txn.pan + "&" + r.txn.expdate + "&" + r.txn.amount

r.txn.merchanturl = "www.yourURL.com"
r.txn.accept = "null"
r.txn.useragent = "Ruby MPI v1.0"

mpiresponse = MpiHttpsPoster.post(r)
```

B. The TXN Response & Creating the InLine Window (Step 6 of Transaction Flow)

The TXN request will return a response with several values. The “mpiresponse.message()” will contain “Y”, “U”, “N”. If the message is “N” the Purchase or PreAuth can be sent with a crypt_type of 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. “U” is returned for non-participating cards, such as corporate cards. Please refer to section 7 for a breakdown of possible liability scenarios.

```
if mpiresponse.message == "Y"
        print "\nFunction to display MpiInLineform"
else
        if mpiresponse.message = "N"
                #send transaction using the mpg API
                #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 merhcnat #assumes
                liability, use crypt_type = "7"

        end
end
```

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 merchant_url field which was provided in the original TXN request, please refer to section 6.a. 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.

```
acsr = Mpirequest.new
acsr.store_id = "monusqa002"
acsr.api_token = "qatoken"

a = Acs.new
acsr.acs = a
acsr.acs.pares = crypt_type #pares
acsr.acs.md = r.txn.md

acsresponse = MpiHttpsPoster.post(acsr)
```

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.

```
if acsresponse.success == "false"
        require "../mpgapi4r.rb"
        order_id = Integer(rand()*10000000000000000000)
        cavv = acsresponse.cavv

        cp = ReqMod::CavvPurchase.new
        cp.order_id = order_id
        cp.amount = "1.00"
        cp.pan = "4242424242424242"
        cp.expdate = "1111"
        cp.cavv = cavv

        cavvresponse = HttpsPoster.post(cp)

        print "\nThe message is: " + response.receipt.message
else
        if mpiresponse.message() = "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.

        print "\nSuccess = " + acsresponse.success
        print "\nMessage = " + acsresponse.message
end
```

NOTE For a sample of a VbV/MCSC Purchase transaction, please refer to the complete Ruby API Integration Guide found at: https://esplusqa.moneris.com/connect/en/download/index.php

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 - There is a liability shift and the merchant is protected from chargebacks	- Fully authenticated - There is a liability shift and the merchant is protected from chargebacks.
6	- VbV has been attempted - There is a liability shift and the Merchant is protected from chargebacks	- MCSC has been attempted - No liability shift - Merchant is not protected from chargebacks
7	- Non-VbV transaction - No liability shift - Merchant is no longer protected from chargebacks	- Non-MCSC transaction - No liability shift - Merchant is not protected from chargebacks

VERes Response	Response Definition
N	The card/issuer is not enrolled. This should be sent as a normal USPurchase/USPreAuth 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 USCavvPurchase/USCavvPreAuth which will return a crypt type of 6.
Y	Fully authenticated and will receive a CAVV. This should now be sent as a USCavvPurchase/USCavvPreAuth 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 Is the cardholder/issuer enrolled?	Step 2: PARes VbV/MCSC InLine window response	Step 3: Transaction Are you protected?
Y	Y	Send a CAVV transaction
Y	N	Send a regular transaction with a crypt type of 7 (strongly encouraged to cancel)
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 using the Moneris MPI in the test environment you will need to use the test store_id and api_token. These are different than your production IDs. The IDs that you can use in the test environment are in the table below.

Test IDs			
store_id	**api_token**	**Username**	**Password**
monusqa004	qatoken	demouser	abc1234
monusqa005	qatoken	demouser	abc1234
monusqa006	qatoken	demouser	abc1234

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 ACS – Send CAVV to eSELECTplus using "USCavvPurchase" or "USCavvPreAuth"
4012001038488884	U	N/A	Send transaction to eSELECTplus using regular "USPurchase" or "USPreAuth". Set crypt_type = 6
4012001038443335	N	N/A	Send transaction to eSELECTplus using regular "USPurchase" or "USPreAuth". Set crypt_type = 7
4242424242424242	Y	true	TXN – call function to create inLine window ACS – Send CAVV to eSELECTplus using "USCavvPurchase" or "USCavvPreAuth"
4012001037461114	Y	false	Card failed to authenticate, merchant may chose to send transaction or decline transaction. If transaction is sent crypt type = 7

VERes – the result U, Y or N is obtained by using: `mpiresponse.message()`
PARes – the result "true" or "false" is obtained by using: `acsresponse.success()`

NOTE

MasterCard SecureCode may not be tested within our current test environment. Though please note, the process and behaviour tested with the above Visa test cards will be the same for MCSC.

For a sample of a "USPurchase", "USPreAuth", "USCavvPurchase" and "USCavvPreAuth" transaction, please refer to the complete Ruby API Integration Guide found at:
https://esplusqa.moneris.com/connect/en/download/index.php

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

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

The test environment will approve and decline 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 $11.00. This limit does not exist in the production environment. For a list of all current test environment responses for various penny values, please see the Test Environment Penny Response table as well as the Test Environment eFraud Response table, available at https://esplusqa.moneris.com/connect/en/download/index.php

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

Ruby CA Root Certificate File:

The default installation of Ruby does not include the Net::HTTP CA root certificate file. In order for the eSelectPlus Ruby API to connect to the eSelectPlus gateway during transaction processing, the 'mpgapi4r.rb' file that's included with the Ruby 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) You will need to download the 'carcert.pem' file from 'http://curl.haxx.se/docs/caextract.html' and save it to the necessary directory. Once downloaded, rename the file to 'ca-certificate.crt and determine the file path (e.g. 'C:\path\to\ ca-certificate.crt).

2) Insert the code below into the 'mpgapi4r.rb' file as part of the option setting, at approximately line 16 below the line 'http.verify_mode = OpenSSL::SSL::VERIFY_PEER'
http.ca_file = "c:\path\to\ca-certificate.crt"

For more information regarding the OpenSSL::SSL::VERIFY_PEER option, please refer to your Ruby manual.

8. How Do I Configure My Store For Production?

Once you have completed testing using the PIT you can move your store to production. You will need to change the "host" from esplusqa.moneris.com to esplus.moneris.com (Note: you do not need to include https://). As well you will need to change the store_id and api_token to the production values for your store.

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: 1-866-696-0488

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

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

10. Appendix A. Definition of Required Fields

Required Fields		
Variable Name	**Size/Type**	**Description**
store_id	12 / an	A value that identifies your company when you send a transaction. Provided by Moneris Solutions.
api_token	20 / an	A unique key that when matched with your store_id creates a secure method of authenticating your store_id. Generated when you first activate your store.
xid	20 / an	Must be exactly 20 alpha numeric characters. This must be unique for every transaction attempt – this can also be used as your order_id when using the eSELECTplus MPG API
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
cardNum	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. 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
storeUrl		This is the URL to which you would like the MPI response sent to. Please refer to section 6.a. and 6.c.
Accept		MIME types the browser accepts
Agent		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. Please refer to section 6.c. for further details.

11. Appendix B. Definition of Response Fields

<table>
<tr><th colspan="3">Response Fields</th></tr>
<tr><th>Variable Name</th><th>Size/Type</th><th>Description</th></tr>
<tr><td>Type</td><td>99 / an</td><td>VERes or PARes or error defines what type of response you are receiving</td></tr>
<tr><td>Success</td><td>true/false</td><td>Returns whether the attempt was successful or not</td></tr>
<tr><td>message</td><td>alpha</td><td>Will contain:
<table>
<tr><th>Txn</th><th>Action</th></tr>
<tr><td>Y</td><td>Create VBV verification form popup window.</td></tr>
<tr><td>N</td><td>Send purchase or preauth with crypt type 6</td></tr>
<tr><td>U</td><td>Send purchase or preauth as crypt type 7</td></tr>
</table>
<table>
<tr><th>ACS</th><th>Action</th></tr>
<tr><td>Y or A (<code>acsresponse.success()</code>=true)</td><td>Proceed with cavv purchase or cavv preauth</td></tr>
<tr><td>N</td><td>Transaction must be cancelled, authentication failed.</td></tr>
<tr><td>U or time out</td><td>Send purchase or preauth as crypt type 7</td></tr>
</table></td></tr>
<tr><td>PARes</td><td>n/a</td><td>Variable Length – data that Visa/MasterCard passes and needs for authentication</td></tr>
<tr><td>TermUrl</td><td>255 / an</td><td>The URL to which the PARes is returned. Please refer to section 6.c.</td></tr>
<tr><td>MD</td><td>1024 / an</td><td>Merchant defined data that will be echoed back</td></tr>
<tr><td>ACSUrl</td><td>255 / an</td><td>URL that will generate the inLine window</td></tr>
<tr><td>cavv</td><td>28 /an</td><td>Visa/MasterCard authentication field</td></tr>
</table>