Moneris™

BE PAYMENT READY

PHP - Moneris Gateway API - Credential on File

Version: 1.0.1

Applies to Canadian integrations only

Table of Contents

1 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/

2 About Credential on File

When storing customers' credit card credentials for use in future authorizations, or when using these credentials in subsequent transactions, card brands now require merchants to indicate this in the transaction request.

In the Moneris API, this is handled by the Moneris Gateway via the inclusion of the Credential on File object and its variables in the transaction request.

While the requirements for handling Credential on File transactions relate to Visa and Mastercard only, in order to avoid confusion and prevent error, please implement these changes for all card types and the Moneris system will then correctly flow the relevant card data values as appropriate.

NOTE: If either the first transaction or a Card Verification authorization is declined when attempting to store cardholder credentials, those credentials cannot be stored —therefore the merchant must not use the credential for any subsequent transactions.

3 Credential on File Info Object and Variables

The Credential on File Info object is nested within the request for the applicable transaction types.

Object:

cof

Variables in the cof object:

Payment Indicator
Payment Information
Issuer ID

For more information, see Definition of Request Fields – Credential on File.

4 Credential on File Transaction Types

The Credential on File Info object applies to the following transaction types:

- Purchase
- Pre-Authorization
- Purchase with Vault – ResPurchaseCC
- Pre-Authorization with Vault – ResPreauthCC
- Card Verification with AVS and CVD
- Card Verification with Vault – ResCardVerificationCC
- Vault Add Credit Card – ResAddCC
- Vault Update Credit Card – ResUpdateCC
- Vault Add Token – ResAddToken
- Recurring Billing transactions (except when updating)

NOTE: For the following transactions, the Credential on File Info object also applies, but Moneris sends the indicators on your behalf:

- Re-Authorization

4.1 Purchase

Purchase transaction object definition

```
$txnArray = array(‘type’=>’purchase', …);

$mpgTxn = new mpgTransaction($txnArray);
```

HttpsPostRequest object for Purchase transaction

```
$mpgRequest = new mpgRequest($mpgTxn);

$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
```

Purchase transaction values

Table 1: Purchase transaction object mandatory values

Value	Type	Limits	Set method
Order ID	String	50-character alphanumeric	`'order_id'=>$order_id`
Amount	String	9-character decimal	`'amount'=>$amount`
Credit card number	String	20-character alphanumeric	`'pan'=>$pan`
Expiry date	String	4-character alphanumeric (YYMM format)	`'expiry_date'=>$expiry_date`
E-commerce indicator	String	1-character alphanumeric	`'crypt'=>$crypt`

Table 2: Purchase transaction object optional values

Value	Type	Limits	Set method
Status Check	Boolean	true/false	`$mpgHttpPost =new mpgHttpsPostStatus($store_id,$api_token,$status,$mpgRequest);`
Customer information	Object	N/A	`$mpgTxn->setCustInfo($mpgCustInfo);`
AVS	Object	N/A	`$mpgTxn->setAvsInfo($mpgAvsInfo);`
CVD **NOTE:** When storing credentials on the initial transaction, the CVD object must be sent; for subsequent transactions using stored credentials, CVD can be sent with cardholder-initiated transactions only—**merchants must not store CVD information**.	Object	N/A	`$mpgTxn->setCvdInfo($mpgCvdInfo);`

Table 2: Purchase transaction object optional values

Value	Type	Limits	Set method
Convenience fee **NOTE:** This variable does not apply to Credential on File transactions.	Object	N/A	`$mpgConvFee = new mpgConvFeeInfo($convFeeTemplate);`
Recurring billing	Object	N/A	`$mpgTxn->setRecur($mpgRecur);`
Dynamic descriptor	String	20-character alphanumeric	`'dynamic_descriptor'=>$dynamic_descriptor`
Wallet indicator[1]	String	3-character alphanumeric	`'wallet_indicator'=>$wallet_indicator`
Credential on File Info `cof` **NOTE:** This is a nested object within the transaction, and required when storing or using the customer's stored credentials. The Credential on File Info object has its own request variables, listed in blue in the table below, "Credential on File Object Request Variables".	Object	N/A	`$mpgTxn->setCofInfo($cof);`

[1]Available to Canadian integrations only.

Credential on File Transaction Object Request Variables

Value	Type	Limits	Set Method
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	`$cof->setIssuerId("VALUE_FOR_ISSUER_ID");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Indicator	String	1-character alphabetic	`$cof->setPaymentIndicator("PAYMENT_INDICATOR_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Information	String	1-character numeric	`$cof->setPaymentInformation("PAYMENT_INFO_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File

Sample Purchase

```
<?php
##
## Example php -q TestPurchase.php store1
##
require "../../mpgClasses.php";
/**************************** Request Variables *******************************/
$store_id='store5';
$api_token='yesguy';
/************************* Transactional Variables ****************************/
$type='purchase';
$cust_id='cust id';
```

Sample Purchase

```
$order_id='ord-'.date("dmy-G:i:s");
$amount='1.00';
$pan='4242424242424242';
$expiry_date='2011';
$crypt='7';
$dynamic_descriptor='123';
$status_check = 'false';
//Optional - Set for Multi-Currency only
//$amount must be 0.00 when using multi-currency
$mcp_amount = '500'; //penny value amount 1.25 = 125
$mcp_currency_code = '840'; //ISO-4217 country currency number
/*********************** Transactional Associative Array **********************/
$txnArray=array('type'=>$type,
'order_id'=>$order_id,
'cust_id'=>$cust_id,
'amount'=>$amount,
'pan'=>$pan,
'expdate'=>$expiry_date,
'crypt_type'=>$crypt,
'dynamic_descriptor'=>$dynamic_descriptor
//,'wallet_indicator' => '' //Refer to documentation for details
//,'mcp_amount' => $mcp_amount,
//'mcp_currency_code' => $mcp_currency_code
);
/**************************** Transaction Object *****************************/
$mpgTxn = new mpgTransaction($txnArray);
/****************** Credential on File **********************************/
$cof = new CofInfo();
$cof->setPaymentIndicator("U");
$cof->setPaymentInformation("2");
$cof->setIssuerId("12345678901234");
$mpgTxn->setCofInfo($cof);
/****************************** Request Object *******************************/
$mpgRequest = new mpgRequest($mpgTxn);
$mpgRequest->setProcCountryCode("CA"); //"US" for sending transaction to US environment
$mpgRequest->setTestMode(true); //false or comment out this line for production transactions
/***************************** HTTPS Post Object *****************************/
/* Status Check Example
$mpgHttpPost =new mpgHttpsPostStatus($store_id,$api_token,$status_check,$mpgRequest);
*/
$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
/******************************** Response ************************************/
$mpgResponse=$mpgHttpPost->getMpgResponse();
print("\nCardType = " . $mpgResponse->getCardType());
print("\nTransAmount = " . $mpgResponse->getTransAmount());
print("\nTxnNumber = " . $mpgResponse->getTxnNumber());
print("\nReceiptId = " . $mpgResponse->getReceiptId());
print("\nTransType = " . $mpgResponse->getTransType());
print("\nReferenceNum = " . $mpgResponse->getReferenceNum());
print("\nResponseCode = " . $mpgResponse->getResponseCode());
print("\nISO = " . $mpgResponse->getISO());
print("\nMessage = " . $mpgResponse->getMessage());
print("\nIsVisaDebit = " . $mpgResponse->getIsVisaDebit());
print("\nAuthCode = " . $mpgResponse->getAuthCode());
print("\nComplete = " . $mpgResponse->getComplete());
print("\nTransDate = " . $mpgResponse->getTransDate());
print("\nTransTime = " . $mpgResponse->getTransTime());
print("\nTicket = " . $mpgResponse->getTicket());
print("\nTimedOut = " . $mpgResponse->getTimedOut());
```

Sample Purchase

```
    print("\nStatusCode = " . $mpgResponse->getStatusCode());
    print("\nStatusMessage = " . $mpgResponse->getStatusMessage());
    print("\nMCPAmount = " . $mpgResponse->getMCPAmount());
    print("\nMCPCurrenyCode = " . $mpgResponse->getMCPCurrencyCode());
    print("\nHostId = " . $mpgResponse->getHostId());
    print("\nIssuerId = " . $mpgResponse->getIssuerId());
    ?>
```

4.2 Pre-Authorization

Pre-Authorization transaction object definition

```
$txnArray = array('type'=>'preauth', …);
```

```
$mpgTxn = new mpgTransaction($txnArray);
```

HttpsPostRequest object for Pre-Authorization transaction

```
$mpgRequest = new mpgRequest($mpgTxn);
```

```
$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
```

Pre-Authorization transaction values

Table 3: Pre-Authorization object mandatory values

Value	Type	Limits	Set method
Order ID	String	50-character alpha-numeric	`'order_id'=>$order_id`
Amount	String	9-character decimal	`'amount'=>$amount`
Credit card number	String	20-character numeric	`'pan'=>$pan`
Expiry date	String	4-character numeric	`'expiry_date'=>$expiry_date`
E-Commerce indicator	String	1-character alpha-numeric	`'crypt'=>$crypt`

Table 4: Pre-Authorization object optional values

Value	Type	Limits	Set method
Status Check	Boolean	true/false	`$mpgHttpPost =new mpgHttpsPostStatus($store_id,$api_token,$status,$mpgRequest);`
Dynamic descriptor	String	20-character alpha-numeric	`'dynamic_descriptor'=>$dynamic_descriptor`
Customer information	Object	N/A	`$mpgTxn->setCustInfo($mpgCustInfo);`
AVS	Object	N/A	`$mpgTxn->setAvsInfo($mpgAvsInfo);`
CVD **NOTE:** When storing credentials on the initial transaction, the CVD object must be sent; for subsequent transactions using stored credentials, CVD can be sent with cardholder-initiated transactions only—**merchants must not store CVD information**.	Object	N/A	`$mpgTxn->setCvdInfo($mpgCvdInfo);`

Value	Type	Limits	Set method
Customer ID	String	50-character alpha-numeric	`'cust_id'=>$cust_id`
Wallet indicator[1]	String	3-character alpha-numeric	`'wallet_indicator'=>$wallet_indicator`
Credential on File Info `cof` **NOTE:** This is a nested object within the transaction, and required when storing or using the customer's stored credentials. The Credential on File Info object has its own request variables, listed in blue in the table below, "Credential on File Object Request Variables".	Object	N/A	`$mpgTxn->setCofInfo($cof);`

[1]Available to Canadian integrations only.

Credential on File Transaction Object Request Variables

Value	Type	Limits	Set Method
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	`$cof->setIssuerId("VALUE_FOR_ISSUER_ID");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Indicator	String	1-character alphabetic	`$cof->setPaymentIndicator("PAYMENT_INDICATOR_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Information	String	1-character numeric	`$cof->setPaymentInformation("PAYMENT_INFO_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File

Sample Pre-Authorization

```
<?php
## Example php -q TestPurchase-VBV.php "moneris" store
require "../../mpgClasses.php";
/******************************** Request Variables ********************************/
$store_id='store5';
$api_token='yesguy';
/****************************** Transactional Variables ***************************/
$type='cavv_preauth';
$order_id='ord-'.date("dmy-G:i:s");
$cust_id='CUST887763';
$amount='10.00';
$pan="4242424242424242";
$expiry_date="0812";
```

Sample Pre-Authorization

```
$cavv='AAABBJg0VhI0VniQEjRWAAAAAAA=';
$crypt_type = '7';
$wallet_indicator = "APP";
$dynamic_descriptor='123456';
/*************************** Transaction Associative Array ************************/
$txnArray=array(
'type'=>$type,
'order_id'=>$order_id,
'cust_id'=>$cust_id,
'amount'=>$amount,
'pan'=>$pan,
'expdate'=>$expiry_date,
'cavv'=>$cavv,
'crypt_type'=>$crypt_type, //mandatory for AMEX only
//'wallet_indicator'=>$wallet_indicator, //set only for wallet transactions. e.g. APPLE PAY
'dynamic_descriptor'=>$dynamic_descriptor
);
/****************************** Transaction Object *******************************/
$mpgTxn = new mpgTransaction($txnArray);
/******************* Credential on File **********************************/
$cof = new CofInfo();
$cof->setPaymentIndicator("U");
$cof->setPaymentInformation("2");
$cof->setIssuerId("12345678901234");
$mpgTxn->setCofInfo($cof);
/******************************* Request Object **********************************/
$mpgRequest = new mpgRequest($mpgTxn);
$mpgRequest->setProcCountryCode("CA"); //"US" for sending transaction to US environment
$mpgRequest->setTestMode(true); //false or comment out this line for production transactions
/****************************** HTTPS Post Object *******************************/
$mpgHttpPost =new mpgHttpsPost($store_id,$api_token,$mpgRequest);
/************************************ Response **********************************/
$mpgResponse=$mpgHttpPost->getMpgResponse();
print("\nCardType = " . $mpgResponse->getCardType());
print("\nTransAmount = " . $mpgResponse->getTransAmount());
print("\nTxnNumber = " . $mpgResponse->getTxnNumber());
print("\nReceiptId = " . $mpgResponse->getReceiptId());
print("\nTransType = " . $mpgResponse->getTransType());
print("\nReferenceNum = " . $mpgResponse->getReferenceNum());
print("\nResponseCode = " . $mpgResponse->getResponseCode());
print("\nISO = " . $mpgResponse->getISO());
print("\nMessage = " . $mpgResponse->getMessage());
print("\nAuthCode = " . $mpgResponse->getAuthCode());
print("\nComplete = " . $mpgResponse->getComplete());
print("\nTransDate = " . $mpgResponse->getTransDate());
print("\nTransTime = " . $mpgResponse->getTransTime());
print("\nTicket = " . $mpgResponse->getTicket());
print("\nTimedOut = " . $mpgResponse->getTimedOut());
print("\nCavvResultCode = " . $mpgResponse->getCavvResultCode());
print("\nIssuerId = " . $mpgResponse->getIssuerId());
?>
```

4.3 Purchase with Vault – ResPurchaseCC

Purchase with Vault transaction object definition

```
$txnArray = array(‘type’=>’res_purchase_cc', …);

$mpgTxn = new mpgTransaction($txnArray);
```

HttpsPostRequest object for Purchase with Vault transaction

```
$mpgRequest = new mpgRequest($mpgTxn);

$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
```

Purchase with Vault transaction values

Table 5: Purchase with Vault transaction object mandatory values

Value	Type	Limits	Set method
Data key	String	25-character alpha-numeric	`'data_key'=>$data_key`
Order ID	String	50-character alpha-numeric	`'order_id'=>$order_id`
Amount	String	9-character decimal	`'amount'=>$amount`
E-commerce indicator	String	1-character alpha-numeric	`'crypt'=>$crypt`
Credential on File Info `cof` **NOTE:** This is a nested object within the transaction, and required when storing or using the customer's stored credentials. The Credential on File Info object has its own request variables, listed in blue in the table below, "Credential on File Object Request Variables".	Object	N/A	`$mpgTxn->setCofInfo($cof);`

Table 6: Purchase with Vault transaction optional values

Value	Type	Limits	Set method
Status Check	Boolean	true/false	`$mpgHttpPost =new mpgHttpsPostStatus($store_id,$api_token,$status,$mpgRequest);`
Expiry date	String	4-character numeric YYMM format. (Note that this is reversed from the date displayed on the card, which is MMYY)	`'expiry_date'=>$expiry_date`
Customer ID	String	50-character alphanumeric	`'cust_id'=>$cust_id`
Dynamic descriptor	String	20-character alphanumeric	`'dynamic_descriptor'=>$dynamic_descriptor`
Customer information	Object	N/A	`$mpgTxn->setCustInfo($mpgCustInfo);`
AVS information	Object	N/A	`$mpgTxn->setAvsInfo($mpgAvsInfo);`
CVD information **NOTE:** When storing credentials on the initial transaction, the CVD object must be sent; for subsequent transactions using stored credentials, CVD can be sent with cardholder-initiated transactions only—**merchants must not store CVD information**.	Object	N/A	`$mpgTxn->setCvdInfo($mpgCvdInfo);`
Recurring billing	Object	N/A	`$mpgTxn->setRecur($mpgRecur);`

Credential on File Transaction Object Request Variables

Value	Type	Limits	Set Method
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	`$cof->setIssuerId("VALUE_FOR_ISSUER_ID");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Indicator	String	1-character alphabetic	`$cof->setPaymentIndicator("PAYMENT_INDICATOR_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Information	String	1-character numeric	`$cof->setPaymentInformation("PAYMENT_INFO_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File

Sample Purchase with Vault

```
<?php
##
## This program takes 3 arguments from the command line:
## 1. Store id
## 2. api token
## 3. order id
##
## Example php -q TestResPurchaseCC.php store3 yesguy unique_order_id 1.00
##
require "../../mpgClasses.php";
/************************ Request Variables **********************************/
```

Sample Purchase with Vault

```
$store_id='store5';
$api_token='yesguy';
/************************ Transaction Variables ******************************/
$data_key='ot-odvn9lBTZm0lSWyQgansBqQi3';
$orderid='res-purch-'.date("dmy-G:i:s");
$amount='1.00';
$custid='cust';
$crypt_type='1';
$expdate='1911'; //For Temp Tokens only
/************************ Transaction Array **********************************/
$txnArray=array('type'=>'res_purchase_cc',
'data_key'=>$data_key,
'order_id'=>$orderid,
'cust_id'=>$custid,
'amount'=>$amount,
'crypt_type'=>$crypt_type,
//'expdate'=>$expdate,
'dynamic_descriptor'=>'12484'
);
/************************ Transaction Object ******************************/
$mpgTxn = new mpgTransaction($txnArray);
/******************* Credential on File **********************************/
$cof = new CofInfo();
$cof->setPaymentIndicator("U");
$cof->setPaymentInformation("2");
$cof->setIssuerId("12345678901234");
$mpgTxn->setCofInfo($cof);
/************************ Request Object **********************************/
$mpgRequest = new mpgRequest($mpgTxn);
$mpgRequest->setProcCountryCode("CA"); //"US" for sending transaction to US environment
$mpgRequest->setTestMode(true); //false or comment out this line for production transactions
/************************ mpgHttpsPost Object ******************************/
$mpgHttpPost =new mpgHttpsPost($store_id,$api_token,$mpgRequest);
/************************ Response Object **********************************/
$mpgResponse=$mpgHttpPost->getMpgResponse();
print("\nDataKey = " . $mpgResponse->getDataKey());
print("\nReceiptId = " . $mpgResponse->getReceiptId());
print("\nReferenceNum = " . $mpgResponse->getReferenceNum());
print("\nResponseCode = " . $mpgResponse->getResponseCode());
print("\nISO = " . $mpgResponse->getISO());
print("\nAuthCode = " . $mpgResponse->getAuthCode());
print("\nMessage = " . $mpgResponse->getMessage());
print("\nTransDate = " . $mpgResponse->getTransDate());
print("\nTransTime = " . $mpgResponse->getTransTime());
print("\nTransType = " . $mpgResponse->getTransType());
print("\nComplete = " . $mpgResponse->getComplete());
print("\nTransAmount = " . $mpgResponse->getTransAmount());
print("\nCardType = " . $mpgResponse->getCardType());
print("\nTxnNumber = " . $mpgResponse->getTxnNumber());
print("\nTimedOut = " . $mpgResponse->getTimedOut());
print("\nAVSResponse = " . $mpgResponse->getAvsResultCode());
print("\nResSuccess = " . $mpgResponse->getResSuccess());
print("\nPaymentType = " . $mpgResponse->getPaymentType());
print("\nIssuerId = " . $mpgResponse->getIssuerId());
//----------------- ResolveData ------------------------------
print("\n\nCust ID = " . $mpgResponse->getResDataCustId());
print("\nPhone = " . $mpgResponse->getResDataPhone());
print("\nEmail = " . $mpgResponse->getResDataEmail());
print("\nNote = " . $mpgResponse->getResDataNote());
```

Sample Purchase with Vault

```
    print("\nMasked Pan = " . $mpgResponse->getResDataMaskedPan());
    print("\nExp Date = " . $mpgResponse->getResDataExpDate());
    print("\nCrypt Type = " . $mpgResponse->getResDataCryptType());
    print("\nAvs Street Number = " . $mpgResponse->getResDataAvsStreetNumber());
    print("\nAvs Street Name = " . $mpgResponse->getResDataAvsStreetName());
    print("\nAvs Zipcode = " . $mpgResponse->getResDataAvsZipcode());
    ?>
```

4.4 Pre-Authorization with Vault – ResPreauthCC

Pre-Authorization with Vault transaction object definition

```
$txnArray = array('type'=>'res_preauth_cc', …);

$mpgTxn = new mpgTransaction($txnArray);
```

HttpsPostRequest object for Pre-Authorization with Vault transaction

```
$mpgRequest = new mpgRequest($mpgTxn);

$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
```

Pre-Authorization with Vault transaction values

Table 7: Pre-Authorization with Vault transaction object mandatory values

Value	Type	Limits	Set method
Data key	String	25- character alpha-numeric	`'data_key'=>$data_key`
Order ID	String	50-character alpha-numeric	`'order_id'=>$order_id`
Amount	String	9-character decimal	`'amount'=>$amount`

Table 7: Pre-Authorization with Vault transaction object mandatory values (continued)

Value	Type	Limits	Set method
E-commerce indicator	String	1-character alpha-numeric	`'crypt'=>$crypt`
Credential on File Info `cof` **NOTE:** This is a nested object within the transaction, and required when storing or using the customer's stored credentials. The Credential on File Info object has its own request variables, listed in blue in the table below, "Credential on File Object Request Variables".	Object	N/A	`$mpgTxn->setCofInfo($cof);`

Table 8: Pre-Authorization with Vault transaction optional values

Value	Type	Limits	Set method
Status Check	Boolean	true/false	`$mpgHttpPost =new mpgHttpsPostStatus($store_id,$api_token,$status,$mpgRequest);`
Expiry date	String	4-character alpha-numeric (YYMM format)	`'expiry_date'=>$expiry_date`
Customer ID	String	50-character alpha-numeric	`'cust_id'=>$cust_id`

Value	Type	Limits	Set method
Customer information	Object	N/A	`$mpgTxn->setCustInfo ($mpgCustInfo);`
AVS information	Object	N/A	`$mpgTxn->setAvsInfo ($mpgAvsInfo);`
CVD information **NOTE:** When storing credentials on the initial transaction, the CVD object must be sent; for subsequent transactions using stored credentials, CVD can be sent with cardholder-initiated transactions only—**merchants must not store CVD information**.	Object	N/A	`$mpgTxn->setCvdInfo ($mpgCvdInfo);`

Credential on File Transaction Object Request Variables

Value	Type	Limits	Set Method
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	`$cof->setIssuerId("VALUE_FOR_ISSUER_ID");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Indicator	String	1-character alphabetic	`$cof->setPaymentIndicator("PAYMENT_INDICATOR_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Information	String	1-character numeric	`$cof->setPaymentInformation("PAYMENT_INFO_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File

Sample Pre-Authorization with Vault

```
<?php
##
## This program takes 3 arguments from the command line:
## 1. Store id
## 2. api token
## 3. order id
##
## Example php -q TestResPreauthCC.php store3 yesguy unique_order_id cust_id 15.00 1
##
require "../../mpgClasses.php";
/************************ Request Variables **********************************/
```

Sample Pre-Authorization with Vault

```
$store_id='store5';
$api_token='yesguy';
/************************ Transaction Variables ******************************/
$data_key='ot-H0q8anK6eeHm0NDe9cwXkDvUw';
$orderid='res-preauth-'.date("dmy-G:i:s");
$amount='1.00';
$custid='cust'; //if sent will be submitted, otherwise cust_id from profile will be used
$crypt_type='1';
//$expdate='1512';
/************************ Transaction Array **********************************/
$txnArray =array('type'=>'res_preauth_cc',
'data_key'=>$data_key,
'order_id'=>$orderid,
'cust_id'=>$custid,
'amount'=>$amount,
'crypt_type'=>$crypt_type,
//'expdate=>$expdate,
'dynamic_descriptor'=>'12424'
);
/************************ Transaction Object ******************************/
$mpgTxn = new mpgTransaction($txnArray);
/******************* Credential on File **********************************/
$cof = new CofInfo();
$cof->setPaymentIndicator("U");
$cof->setPaymentInformation("2");
$cof->setIssuerId("12345678901234");
$mpgTxn->setCofInfo($cof);
/************************ Request Object **********************************/
$mpgRequest = new mpgRequest($mpgTxn);
$mpgRequest->setProcCountryCode("CA"); //"US" for sending transaction to US environment
$mpgRequest->setTestMode(true); //false or comment out this line for production transactions
/************************ mpgHttpsPost Object ******************************/
$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
/************************ Response Object **********************************/
$mpgResponse=$mpgHttpPost->getMpgResponse();
print("\nDataKey = " . $mpgResponse->getDataKey());
print("\nReceiptId = " . $mpgResponse->getReceiptId());
print("\nReferenceNum = " . $mpgResponse->getReferenceNum());
print("\nResponseCode = " . $mpgResponse->getResponseCode());
print("\nISO = " . $mpgResponse->getISO());
print("\nAuthCode = " . $mpgResponse->getAuthCode());
print("\nMessage = " . $mpgResponse->getMessage());
print("\nTransDate = " . $mpgResponse->getTransDate());
print("\nTransTime = " . $mpgResponse->getTransTime());
print("\nTransType = " . $mpgResponse->getTransType());
print("\nComplete = " . $mpgResponse->getComplete());
print("\nTransAmount = " . $mpgResponse->getTransAmount());
print("\nCardType = " . $mpgResponse->getCardType());
print("\nTxnNumber = " . $mpgResponse->getTxnNumber());
print("\nTimedOut = " . $mpgResponse->getTimedOut());
print("\nAVSResponse = " . $mpgResponse->getAvsResultCode());
print("\nResSuccess = " . $mpgResponse->getResSuccess());
print("\nPaymentType = " . $mpgResponse->getPaymentType());
print("\nIssuerId = " . $mpgResponse->getIssuerId());
//----------------- ResolveData ------------------------------
print("\n\nCust ID = " . $mpgResponse->getResDataCustId());
print("\nPhone = " . $mpgResponse->getResDataPhone());
print("\nEmail = " . $mpgResponse->getResDataEmail());
print("\nNote = " . $mpgResponse->getResDataNote());
```

Sample Pre-Authorization with Vault

```
print("\nMasked Pan = " . $mpgResponse->getResDataMaskedPan());
print("\nExp Date = " . $mpgResponse->getResDataExpDate());
print("\nCrypt Type = " . $mpgResponse->getResDataCryptType());
print("\nAvs Street Number = " . $mpgResponse->getResDataAvsStreetNumber());
print("\nAvs Street Name = " . $mpgResponse->getResDataAvsStreetName());
print("\nAvs Zipcode = " . $mpgResponse->getResDataAvsZipcode());
?>
```

4.5 Card Verification and Credential on File Transactions

NOTE: The following information applies to Visa, Mastercard and Discover transactions only.

In certain cases, some Credential on File transactions require the prior use of a Card Verification transaction.

In the absence of a Purchase or Pre-Authorization, a Card Verification transaction is used to get the unique Issuer ID value that is used in subsequent Credential on File transactions. Issuer ID is a variable included in the nested Credential on File Info object. For a complete list of these variables, see each transaction type or Definition of Request Fields – Credential on File

The Card Verification request, including the Credential on File Info object, must be sent immediately prior to sending the transactions in these scenarios.

4.5.1 When to Use Card Verification With COF

If you are not sending a Purchase or Pre-Authorization transaction (i.e., you are not charging the customer immediately), you must use Card Verification (or in the case of Vault Add Token, Card Verification with Vault) first before running the transaction in order to get the Issuer ID.

Transactions this applies to:

Vault Add Credit Card
Vault Update Credit Card
Vault Add Token
Recurring Billing transaction (first in series), if:

- the first transaction does not begin immediately

4.5.2 Credential on File and Vault Add Token

For Vault Add Token transactions:

1. Send Card Verification with Vault transaction request including the Credential on File object to get the Issuer ID
2. Send the Vault Add Token request including the Credential on File object

4.5.2.1 Vault Add Token – ResAddToken

Things to Consider:

- This transaction is used to convert a temporary token into a permanent token for storage in the Moneris Vault
- If you intend to store the token for use in future transactions (i.e., Credential on File transactions), **first** you must send either a Vault financial transaction (Purchase with Vault or Pre-Authorization with Vault) or a Card Verification with Vault in order to get the Issuer ID

Vault Add Token transaction object definition

```
$txnArray = array('type'=>'res_add_token', …);

$mpgTxn = new mpgTransaction($txnArray);
```

HttpsPostRequest object for Vault Add Token transaction

```
$mpgRequest = new mpgRequest($mpgTxn);

$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
```

Vault Add Token transaction values

Table 9: Vault Add Token transaction object mandatory values

Value	Type	Limits	Set method
Data key	String	28-character alpha-numeric	`'data_key'=>$data_key`
E-commerce indicator	String	1-character alpha-numeric	`'crypt'=>$crypt`
Credential on File Info `cof` **NOTE:** This is a nested object within the transaction, and required when storing or using the customer's stored	Object	N/A	`$mpgTxn->setCofInfo($cof);`

Value	Type	Limits	Set method
credentials. The Credential on File Info object has its own request variables, listed in blue in the table below, "Credential on File Object Request Variables".			

Table 10: **Vault Add Token transaction optional values**

Value	Type	Limits	Set method
Customer ID	String	50-character alpha-numeric	`'cust_id'=>$cust_id`
AVS information	Object	N/A	`$mpgTxn->setAvsInfo($mpgAvsInfo);`
Email address	String	30-character alpha-numeric	`'email'=>$email`
Phone number	String	30-character alpha-numeric	`'phone'=>$phone`
Note	String	30-character alpha-numeric	`'note'=>$note`
Data key format[1]	String	2-character alpha-numeric	`'data_key_format'=>$data_key_format`

[1]Available to Canadian integrations only.

Credential on File Transaction Object Request Variables

Value	Type	Limits	Set Method
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	`$cof->setIssuerId("VALUE_FOR_ISSUER_ID");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Indicator	String	1-character alphabetic	`$cof->setPaymentIndicator("PAYMENT_INDICATOR_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Information	String	1-character numeric	`$cof->setPaymentInformation("PAYMENT_INFO_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File

Sample Vault Add Token

```
<?php
require "../../mpgClasses.php";
/**************************** Request Variables *******************************/
$store_id='store5';
$api_token='yesguy';
/************************* Transactional Variables ****************************/
$type='res_add_token';
$temp_data_key='ot-mtNKdu8NcxDoChqOJKZJZ1BOB';
$cust_id='customer1';
$phone = '5555551234';
$email = 'bob@smith.com';
```

Sample Vault Add Token

```
$note = 'this is my note';
$expiry_date='1811';
$data_key_format = "0";
$crypt_type='1';
$avs_street_number = '123';
$avs_street_name = 'lakeshore blvd';
$avs_zipcode = '90210';
/*********************** Transactional Associative Array **********************/
$txnArray=array('type'=>$type,
'data_key'=>$temp_data_key,
'cust_id'=>$cust_id,
'phone'=>$phone,
'email'=>$email,
'note'=>$note,
'expdate'=>$expiry_date,
//'data_key_format'=>$data_key_format, //optional
'crypt_type'=>$crypt_type
);
/********************** AVS Associative Array *********************************/
$avsTemplate = array(
'avs_street_number' => $avs_street_number,
'avs_street_name' => $avs_street_name,
'avs_zipcode' => $avs_zipcode
);
/************************** AVS Object ***************************************/
$mpgAvsInfo = new mpgAvsInfo ($avsTemplate);
/****************** Credential on File **********************************/
$cof = new CofInfo();
$cof->setPaymentIndicator("U");
$cof->setPaymentInformation("2");
$cof->setIssuerId("12345678901234");
/**************************** Transaction Object *****************************/
$mpgTxn = new mpgTransaction($txnArray);
$mpgTxn->setAvsInfo($mpgAvsInfo);
$mpgTxn->setCofInfo($cof);
/****************************** Request Object ******************************/
$mpgRequest = new mpgRequest($mpgTxn);
$mpgRequest->setProcCountryCode("CA"); //"US" for sending transaction to US environment
$mpgRequest->setTestMode(true); //false or comment out this line for production transactions
/***************************** HTTPS Post Object *****************************/
$mpgHttpPost =new mpgHttpsPost($store_id,$api_token,$mpgRequest);
/****************************** Response ************************************/
$mpgResponse=$mpgHttpPost->getMpgResponse();
print("\nDataKey = " . $mpgResponse->getDataKey());
print("\nResponseCode = " . $mpgResponse->getResponseCode());
print("\nMessage = " . $mpgResponse->getMessage());
print("\nTransDate = " . $mpgResponse->getTransDate());
print("\nTransTime = " . $mpgResponse->getTransTime());
print("\nComplete = " . $mpgResponse->getComplete());
print("\nTimedOut = " . $mpgResponse->getTimedOut());
print("\nResSuccess = " . $mpgResponse->getResSuccess());
print("\nPaymentType = " . $mpgResponse->getPaymentType());
print("\nIssuerId = " . $mpgResponse->getIssuerId());
//----------------- ResolveData ------------------------------
print("\n\nCust ID = " . $mpgResponse->getResDataCustId());
print("\nPhone = " . $mpgResponse->getResDataPhone());
print("\nEmail = " . $mpgResponse->getResDataEmail());
print("\nNote = " . $mpgResponse->getResDataNote());
print("\nMasked Pan = " . $mpgResponse->getResDataMaskedPan());
```

Sample Vault Add Token

```
print("\nExp Date = " . $mpgResponse->getResDataExpDate());
print("\nCrypt Type = " . $mpgResponse->getResDataCryptType());
print("\nAvs Street Number = " . $mpgResponse->getResDataAvsStreetNumber());
print("\nAvs Street Name = " . $mpgResponse->getResDataAvsStreetName());
print("\nAvs Zipcode = " . $mpgResponse->getResDataAvsZipcode());
?>
```

4.5.3 Credential on File and Vault Update Credit Card

For Vault Update Credit Card transactions:

1. Send Card Verification transaction request including the Credential on File object to get the Issuer ID
2. Send the Vault Update Credit Card request including the Credential on File object

4.5.3.1 Vault Update Credit Card – ResUpdateCC

Things to Consider:

- Updates a Vault profile (based on the data key) to contain credit card information. All information contained within a credit card profile is updated as indicated by the submitted fields.
- This will update a profile to contain Credit Card information by referencing the profile's unique data_key. If the profile which is being updated was already a Credit Card profile, all information contained within it will simply be updated as indicated by the submitted fields. This means that all fields are optional, and only those fields that are submitted will be updated.
- To update a specific field on the profile, only set that specific element using the corresponding set method.

Vault Update Credit Card transaction object definition

```
$txnArray = array('type'=>'res_update_cc', …);
```

```
$mpgTxn = new mpgTransaction($txnArray);
```

HttpsPostRequest object for Vault Update Credit Card transaction

```
$mpgRequest = new mpgRequest($mpgTxn);
```

```
$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
```

Vault Update Credit Card transaction values

Table 11: Vault Update Credit Card transaction object mandatory values

Value	Type	Limits	Set method
Data key	String	25-character alpha-numeric	`'data_key'=>$data_key`

Optional values that are submitted to the ResUpdateCC object are updated. Unsubmitted optional values (with one exception) remain unchanged. This allows you to change only the fields you want.

The exception is that if you are making changes to the payment type, **all** of the variables in the optional values table below must be submitted.

If you update a profile to a different payment type, it is automatically deactivated and a new credit card profile is created and assigned to the data key. The only values from the prior profile that will remain unchanged are the customer ID, phone number, email address, and note.

EXAMPLE: If a profile contains AVS information, but a ResUpdateCC transaction is submitted without an AVSInfo object, the existing AVSInfo details are deactivated and the new credit card information is registered without AVS.

Table 12: Vault Update Credit Card transaction optional values

Value	Type	Limits	Set method
Credit card number	String	20-character alpha-numeric	`'pan'=>$pan`
Expiry date	String	4-character alpha-numeric (YYMM format)	`'expiry_date'=>$expiry_date`
E-commerce indicator	String	1-character alpha-numeric	`'crypt'=>$crypt`
Customer ID	String	50-character alpha-numeric	`'cust_id'=>$cust_id`
AVS information	Object	n/a	`$mpgTxn->setAvsInfo($mpgAvsInfo);`
Email address	String	30-character alpha-numeric	`'email'=>$email`
Phone number	String	30-character alpha-	`'phone'=>$phone`

Value	Type	Limits	Set method
		numeric	
Note	String	30-character alpha-numeric	`'note'=>$note`
Credential on File Info `cof` **NOTE:** This is a nested object within the transaction, and required when storing or using the customer's stored credentials. The Credential on File Info object has its own request variables, listed in blue in the table below, "Credential on File Object Request Variables".	Object	N/A	`$mpgTxn->setCofInfo($cof);`

Credential on File Transaction Object Request Variables

Value	Type	Limits	Set Method
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	`$cof->setIssuerId("VALUE_FOR_ISSUER_ID");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Indicator	String	1-character alphabetic	`$cof->setPaymentIndicator("PAYMENT_INDICATOR_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Information	String	1-character numeric	`$cof->setPaymentInformation("PAYMENT_INFO_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File

Sample Vault Update Credit Card

```
<?php
##
## Example php -q TestResUpdateCC.php store3 yesguy
##
require "../../mpgClasses.php";
/**************************** Request Variables *******************************/
$store_id='store5';
$api_token='yesguy';
/************************* Transactional Variables ****************************/
$type='res_update_cc';
$data_key='D8cpd4r7REXoN8NIJPi512xPh';
```

Sample Vault Update Credit Card

```
$cust_id='customer1';
$phone = '5555555555';
$email = 'bob@smith.com';
$note = 'stuff';
$pan='5454545454545454';
$expiry_date='0909';
$crypt_type='7';
$avs_street_number = '123';
$avs_street_name = 'stuff dr';
$avs_zipcode = '90215';
/********************** Transactional Associative Array **********************/
$txnArray=array('type'=>$type,
'data_key'=>$data_key,
'cust_id'=>$cust_id,
'phone'=>$phone,
'email'=>$email,
'note'=>$note,
'pan'=>$pan,
'expdate'=>$expiry_date,
'crypt_type'=>$crypt_type
);
/********************** AVS Associative Array *********************************/
$avsTemplate = array(
'avs_street_number' => $avs_street_number,
'avs_street_name' => $avs_street_name,
'avs_zipcode' => $avs_zipcode
);
/************************* AVS Object **************************************/
$mpgAvsInfo = new mpgAvsInfo ($avsTemplate);
/****************** Credential on File *********************************/
$cof = new CofInfo();
$cof->setPaymentIndicator("U");
$cof->setPaymentInformation("2");
$cof->setIssuerId("12345678901234");
/**************************** Transaction Object ****************************/
$mpgTxn = new mpgTransaction($txnArray);
$mpgTxn->setAvsInfo($mpgAvsInfo);
$mpgTxn->setCofInfo($cof);
/****************************** Request Object ******************************/
$mpgRequest = new mpgRequest($mpgTxn);
$mpgRequest->setProcCountryCode("CA"); //"US" for sending transaction to US environment
$mpgRequest->setTestMode(true); //false or comment out this line for production transactions
/***************************** HTTPS Post Object ****************************/
$mpgHttpPost =new mpgHttpsPost($store_id,$api_token,$mpgRequest);
/******************************** Response ***********************************/
$mpgResponse=$mpgHttpPost->getMpgResponse();
print("\nDataKey = " . $mpgResponse->getDataKey());
print("\nResponseCode = " . $mpgResponse->getResponseCode());
print("\nMessage = " . $mpgResponse->getMessage());
print("\nTransDate = " . $mpgResponse->getTransDate());
print("\nTransTime = " . $mpgResponse->getTransTime());
print("\nComplete = " . $mpgResponse->getComplete());
print("\nTimedOut = " . $mpgResponse->getTimedOut());
print("\nResSuccess = " . $mpgResponse->getResSuccess());
print("\nPaymentType = " . $mpgResponse->getPaymentType());
//----------------- ResolveData ------------------------------
print("\n\nCust ID = " . $mpgResponse->getResDataCustId());
print("\nPhone = " . $mpgResponse->getResDataPhone());
print("\nEmail = " . $mpgResponse->getResDataEmail());
```

Sample Vault Update Credit Card

```
print("\nNote = " . $mpgResponse->getResDataNote());
print("\nMasked Pan = " . $mpgResponse->getResDataMaskedPan());
print("\nExp Date = " . $mpgResponse->getResDataExpDate());
print("\nCrypt Type = " . $mpgResponse->getResDataCryptType());
print("\nAvs Street Number = " . $mpgResponse->getResDataAvsStreetNumber());
print("\nAvs Street Name = " . $mpgResponse->getResDataAvsStreetName());
print("\nAvs Zipcode = " . $mpgResponse->getResDataAvsZipcode());
?>
```

4.5.4 Credential on File and Vault Add Credit Card

For Vault Add Credit Card transactions:

1. Send Card Verification transaction request including the Credential on File object to get the Issuer ID
2. Send the Vault Add Credit Card request including the Credential on File object

4.5.4.1 Vault Add Credit Card – ResAddCC

ResAddCC transaction object definition

```
$txnArray = array('type'=>'resaddcc', …);
$mpgTxn = new mpgTransaction($txnArray);
```

HttpsPostRequest object for ResAddCC transaction

```
$mpgRequest = new mpgRequest($mpgTxn);
$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
```

ResAddCC transaction values

Table 13: Vault Add Credit Card transaction object mandatory values

Value	Type	Limits	Set method
Credit card number	String	20-character alpha-numeric	`'pan'=>$pan`
Expiry date	String	4-character alpha-numeric (YYMM format)	`'expiry_date'=>$expiry_date`
E-commerce indicator	String	1-character alpha-numeric	`'crypt'=>$crypt`
Credential on File Info	Object	N/A	

Value	Type	Limits	Set method
`cof` **NOTE:** This is a nested object within the transaction, and required when storing or using the customer's stored credentials. The Credential on File Info object has its own request variables, listed in blue in the table below, "Credential on File Object Request Variables".			`$mpgTxn->setCofInfo($cof);`

Table 14: Vault Add Credit Card transaction optional values

Value	Type	Limits	Set method
Customer ID	String	50-character alphanumeric	`'cust_id'=>$cust_id`
AVS information	Object	N/A	`$mpgTxn->setAvsInfo($mpgAvsInfo);`
Email address	String	30-character alphanumeric	`'email'=>$email`
Phone number	String	30-character alphanumeric	`'phone'=>$phone`
Note	String	30-character alphanumeric	`'note'=>$note`
Data key format[1]	String	2-character alphanumeric	`'data_key_format'=>$data_key_format`

[1]Available to Canadian integrations only.

Credential on File Transaction Object Request Variables

Value	Type	Limits	Set Method
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	`$cof->setIssuerId("VALUE_FOR_ISSUER_ID");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Indicator	String	1-character alphabetic	`$cof->setPaymentIndicator("PAYMENT_INDICATOR_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Information	String	1-character numeric	`$cof->setPaymentInformation("PAYMENT_INFO_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File

Sample Vault Add Credit Card

```
<?php
##
## Example php -q TestResAddCC.php store3 yesguy
##
require "../../mpgClasses.php";
/**************************** Request Variables *******************************/
$store_id='store5';
$api_token='yesguy';
/************************* Transactional Variables ****************************/
$type='res_add_cc';
$cust_id='customer1';
```

Sample Vault Add Credit Card

```
$phone = '5555551234';
$email = 'bob@smith.com';
$note = 'this is my note';
$pan='5454545454545454';
$expiry_date='1412';
$crypt_type='1';
$data_key_format = "0";
$avs_street_number = '123';
$avs_street_name = 'lakeshore blvd';
$avs_zipcode = '90210';
/*********************** Transactional Associative Array **********************/
$txnArray=array('type'=>$type,
'cust_id'=>$cust_id,
'phone'=>$phone,
'email'=>$email,
'note'=>$note,
'pan'=>$pan,
'expdate'=>$expiry_date,
//'data_key_format'=>$data_key_format, //optional
'crypt_type'=>$crypt_type
);
/********************** AVS Associative Array *********************************/
$avsTemplate = array(
'avs_street_number' => $avs_street_number,
'avs_street_name' => $avs_street_name,
'avs_zipcode' => $avs_zipcode
);
/************************** AVS Object ***************************************/
$mpgAvsInfo = new mpgAvsInfo ($avsTemplate);
/**************************** Transaction Object ****************************/
$mpgTxn = new mpgTransaction($txnArray);
$mpgTxn->setAvsInfo($mpgAvsInfo);
/****************** Credential on File **********************************/
$cof = new CofInfo();
$cof->setPaymentIndicator("U");
$cof->setPaymentInformation("2");
$cof->setIssuerId("12345678901234");
$mpgTxn->setCofInfo($cof);
/****************************** Request Object ******************************/
$mpgRequest = new mpgRequest($mpgTxn);
$mpgRequest->setProcCountryCode("CA"); //"US" for sending transaction to US environment
$mpgRequest->setTestMode(true); //false or comment out this line for production transactions
/****************************** HTTPS Post Object ****************************/
$mpgHttpPost =new mpgHttpsPost($store_id,$api_token,$mpgRequest);
/******************************* Response ************************************/
$mpgResponse=$mpgHttpPost->getMpgResponse();
print("\nDataKey = " . $mpgResponse->getDataKey());
print("\nResponseCode = " . $mpgResponse->getResponseCode());
print("\nMessage = " . $mpgResponse->getMessage());
print("\nTransDate = " . $mpgResponse->getTransDate());
print("\nTransTime = " . $mpgResponse->getTransTime());
print("\nComplete = " . $mpgResponse->getComplete());
print("\nTimedOut = " . $mpgResponse->getTimedOut());
print("\nResSuccess = " . $mpgResponse->getResSuccess());
print("\nPaymentType = " . $mpgResponse->getPaymentType());
print("\nIssuerId = " . $mpgResponse->getIssuerId());
//----------------- ResolveData ------------------------------
print("\n\nCust ID = " . $mpgResponse->getResDataCustId());
print("\nPhone = " . $mpgResponse->getResDataPhone());
```

Sample Vault Add Credit Card

```
    print("\nEmail = " . $mpgResponse->getResDataEmail());
    print("\nNote = " . $mpgResponse->getResDataNote());
    print("\nMasked Pan = " . $mpgResponse->getResDataMaskedPan());
    print("\nExp Date = " . $mpgResponse->getResDataExpDate());
    print("\nCrypt Type = " . $mpgResponse->getResDataCryptType());
    print("\nAvs Street Number = " . $mpgResponse->getResDataAvsStreetNumber());
    print("\nAvs Street Name = " . $mpgResponse->getResDataAvsStreetName());
    print("\nAvs Zipcode = " . $mpgResponse->getResDataAvsZipcode());
    ?>
```

4.5.5 Credential on File and Recurring Billing

NOTE: Updating Recurring Billing transactions (using the UpdateRecur object) is not currently permitted with Credential on File.

For Recurring Billing transactions which are set to start **immediately**:

- Send a Purchase transaction request with both the Recur and Credential on File objects.

For Recurring Billing transactions which are set to start on a **future** date:

1. Send Card Verification transaction request including the Credential on File object to get the Issuer ID
2. Send Purchase transaction request with the Recur and Credential on File objects included

For more information about the Recur object, see Definition of Request Fields – Recurring.

4.5.6 Card Verification with AVS and CVD

Things to Consider:

- The Card Verification transaction is only supported by Visa, MasterCard and Discover
- For some Credential on File transactions, Card Verification is used as a prior step to get the Issuer ID used in the subsequent transaction
- This transaction is also known as an "account status inquiry"

Card Verification object definition

```
$txnArray = array('type'=>'cardVerification', …);

$mpgTxn = new mpgTransaction($txnArray);
```

HttpsPostRequest object for Card Verification transaction

```
$mpgRequest = new mpgRequest($mpgTxn);

$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
```

Card Verification transaction values

Table 15: Card Verification transaction object mandatory values

Value	Type	Limits	Set method
Order ID	String	50-character alpha-numeric	`'order_id'=>$order_id`
Credit card number	String	20-character alpha-numeric	`'pan'=>$pan`
Expiry date	String	4-character alpha-numeric (YYMM format)	`'expiry_date'=>$expiry_date`
E-commerce indicator	String	1-character alpha-numeric	`'crypt'=>$crypt`
AVS	Object	N/A	`$mpgTxn->setAvsInfo($mpgAvsInfo);`
CVD **NOTE:** When storing credentials on the initial transaction, the CVD object must be sent; for subsequent transactions using stored credentials, CVD can be sent with cardholder-initiated transactions only—**merchants must not store CVD information**.	Object	N/A	`$mpgTxn->setCvdInfo($mpgCvdInfo);`

Table 16: Basic Card Verification transaction object optional values

Value	Type	Limits	Set Method
Credential on File Info `cof` **NOTE:** This is a nested object within the transaction, and required when storing or using the customer's stored credentials. The Credential on File Info object has its own request variables, listed in blue in the table below, "Credential on File Object Request Variables".	Object	N/A	`$mpgTxn->setCofInfo($cof);`

Credential on File Transaction Object Request Variables

Value	Type	Limits	Set Method
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	`$cof->setIssuerId("VALUE_FOR_ISSUER_ID");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Indicator	String	1-character alphabetic	`$cof->setPaymentIndicator("PAYMENT_INDICATOR_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Information	String	1-character numeric	`$cof->setPaymentInformation("PAYMENT_INFO_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File

Sample Card Verification

```
<?php
require "../../mpgClasses.php";
$store_id='store5';
$api_token="yesguy";
$txnArray=array('type'=>'card_verification',
'order_id'=>'ord-'.date("dmy-G:i:s"),
'cust_id'=>'my cust id',
'pan'=>'4242424242424242',
'expdate'=>'1512',
'crypt_type'=>'7'
);
$mpgTxn = new mpgTransaction($txnArray);
/************************** AVS Variables *****************************/
```

Sample Card Verification

```
$avs_street_number = '201';
$avs_street_name = 'Michigan Ave';
$avs_zipcode = 'M1M1M1';
/************************** CVD Variables *****************************/
$cvd_indicator = '1';
$cvd_value = '198';
/********************** AVS Associative Array *************************/
$avsTemplate = array(
'avs_street_number'=>$avs_street_number,
'avs_street_name' =>$avs_street_name,
'avs_zipcode' => $avs_zipcode
);
/********************** CVD Associative Array *************************/
$cvdTemplate = array(
'cvd_indicator' => $cvd_indicator,
'cvd_value' => $cvd_value
);
/************************** AVS Object ********************************/
$mpgAvsInfo = new mpgAvsInfo ($avsTemplate);
/************************** CVD Object ********************************/
$mpgCvdInfo = new mpgCvdInfo ($cvdTemplate);
/*********************** Credential on File ************************/
$cof = new CofInfo();
$cof->setPaymentIndicator("U");
$cof->setPaymentInformation("2");
$cof->setIssuerId("12345678901234");
$mpgTxn->setAvsInfo($mpgAvsInfo);
$mpgTxn->setCvdInfo($mpgCvdInfo);
$mpgTxn->setCofInfo($cof);
$mpgRequest = new mpgRequest($mpgTxn);
$mpgRequest->setProcCountryCode("CA"); //"US" for sending transaction to US environment
$mpgRequest->setTestMode(true); //false or comment out this line for production transactions
$mpgHttpPost =new mpgHttpsPost($store_id,$api_token,$mpgRequest);
$mpgResponse=$mpgHttpPost->getMpgResponse();
print("\nCardType = " . $mpgResponse->getCardType());
print("\nTransAmount = " . $mpgResponse->getTransAmount());
print("\nTxnNumber = " . $mpgResponse->getTxnNumber());
print("\nReceiptId = " . $mpgResponse->getReceiptId());
print("\nTransType = " . $mpgResponse->getTransType());
print("\nReferenceNum = " . $mpgResponse->getReferenceNum());
print("\nResponseCode = " . $mpgResponse->getResponseCode());
print("\nISO = " . $mpgResponse->getISO());
print("\nMessage = " . $mpgResponse->getMessage());
print("\nIsVisaDebit = " . $mpgResponse->getIsVisaDebit());
print("\nAuthCode = " . $mpgResponse->getAuthCode());
print("\nComplete = " . $mpgResponse->getComplete());
print("\nTransDate = " . $mpgResponse->getTransDate());
print("\nTransTime = " . $mpgResponse->getTransTime());
print("\nTicket = " . $mpgResponse->getTicket());
print("\nTimedOut = " . $mpgResponse->getTimedOut());
print("\nIssuerId = " . $mpgResponse->getIssuerId());
?>
```

4.5.7 Card Verification with Vault – ResCardVerificationCC

Things to Consider:

- This transaction type only applies to Visa, Mastercard and Discover transactions
- This transaction is also known as an "account status inquiry"
- The card number and expiry date for this transaction are passed using a token, as represented by the data key value
- When using a temporary token (e.g., such as with Hosted Tokenization) **and** you intend to store the cardholder credentials, this transaction must be run prior to running the Vault Add Token transaction

Card Verification object definition

```
$txnArray = array(‘type’=>’resCardVerificationCC', …);

$mpgTxn = new mpgTransaction($txnArray);
```

HttpsPostRequest object for Card Verification transaction

```
$mpgRequest = new mpgRequest($mpgTxn);

$mpgHttpPost = new mpgHttpsPost($store_id,$api_token,$mpgRequest);
```

Card Verification transaction values

Table 17: Card Verification with Vault transaction object mandatory values

Value	Type	Limits	Set method
Order ID	String	50-character alpha-numeric	`'order_id'=>$order_id`
Data key	String	25-character alpha-numeric	`'data_key'=>$data_key`
E-commerce indicator	String	1-character alpha-numeric	`'crypt'=>$crypt`

Table 17: Card Verification with Vault transaction object mandatory values

Value	Type	Limits	Set method
AVS	Object	N/A	`$mpgTxn->setAvsInfo($mp-gAvsInfo);`
CVD	Object	N/A	`$mpgTxn->setCvdInfo($mp-gCvdInfo);`
Credential on File Info `cof` **NOTE:** This is a nested object within the trans-action, and required when storing or using the customer's stored credentials. The Cre-dential on File Info object has its own request variables, lis-ted in blue in the table below, "Credential on File Object Request Variables".	Object	N/A	`$mpgTxn->setCofInfo($cof);`

Credential on File Transaction Object Request Variables

Value	Type	Limits	Set Method
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	`$cof->setIssuerId("VALUE_FOR_ISSUER_ID");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Indicator	String	1-character alphabetic	`$cof->setPaymentIndicator("PAYMENT_INDICATOR_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File
Payment Information	String	1-character numeric	`$cof->setPaymentInformation("PAYMENT_INFO_VALUE");` **NOTE:** For a list and explanation of the possible values to send for this variable, see Definition of Request Fields – Credential on File

Sample Card Verification with Vault

```
<?php
##
## This program takes 3 arguments from the command line:
## 1. Store id
## 2. api token
## 3. order id
##
## Example php -q TestResPurchaseCC.php store3 yesguy unique_order_id 1.00
##
require "../../mpgClasses.php";
/************************ Request Variables **********************************/
```

Sample Card Verification with Vault

```
$store_id='store5';
$api_token='yesguy';
/************************ Transaction Variables ******************************/
$data_key='t8RCndWBNFNt4Dx32CCnl2tlz';
$orderid='res-purch-'.date("dmy-G:i:s");
$crypt_type='1';
$expdate='1911'; //for temp token
/************************ Transaction Array **********************************/
$txnArray=array('type'=>'res_card_verification_cc',
'data_key'=>$data_key,
'order_id'=>$orderid,
'crypt_type'=>$crypt_type,
'expdate'=>$expdate
);
/************************** CVD Variables *****************************/
$cvd_indicator = '1';
$cvd_value = '198';
/********************** CVD Associative Array ************************/
$cvdTemplate = array(
'cvd_indicator' => $cvd_indicator,
'cvd_value' => $cvd_value
);
$mpgCvdInfo = new mpgCvdInfo ($cvdTemplate);
/************************** AVS Variables *****************************/
//The AVS portion is optional if AVS details are already stored in this profile
//If AVS details are resent in Purchase transaction, they will replace stored details
$avs_street_number = '';
$avs_street_name = 'bloor st';
$avs_zipcode = '111111';
/********************** AVS Associative Array ************************/
$avsTemplate = array(
'avs_street_number' => $avs_street_number,
'avs_street_name' => $avs_street_name,
'avs_zipcode' => $avs_zipcode
);
$mpgAvsInfo = new mpgAvsInfo ($avsTemplate);
/************************ Transaction Object *******************************/
$mpgTxn = new mpgTransaction($txnArray);
$mpgTxn->setCvdInfo($mpgCvdInfo);
$mpgTxn->setAvsInfo($mpgAvsInfo);
/****************** Credential on File **********************************/
$cof = new CofInfo();
$cof->setPaymentIndicator("U");
$cof->setPaymentInformation("2");
$cof->setIssuerId("12345678901234");
$mpgTxn->setCofInfo($cof);
/************************ Request Object **********************************/
$mpgRequest = new mpgRequest($mpgTxn);
$mpgRequest->setProcCountryCode("CA"); //"US" for sending transaction to US environment
$mpgRequest->setTestMode(true); //false or comment out this line for production transactions
/************************ mpgHttpsPost Object ****************************/
$mpgHttpPost =new mpgHttpsPost($store_id,$api_token,$mpgRequest);
/************************ Response Object **********************************/
$mpgResponse=$mpgHttpPost->getMpgResponse();
print("\nDataKey = " . $mpgResponse->getDataKey());
print("\nReceiptId = " . $mpgResponse->getReceiptId());
print("\nReferenceNum = " . $mpgResponse->getReferenceNum());
print("\nResponseCode = " . $mpgResponse->getResponseCode());
print("\nISO = " . $mpgResponse->getISO());
```

Sample Card Verification with Vault

```
print("\nAuthCode = " . $mpgResponse->getAuthCode());
print("\nMessage = " . $mpgResponse->getMessage());
print("\nTransDate = " . $mpgResponse->getTransDate());
print("\nTransTime = " . $mpgResponse->getTransTime());
print("\nTransType = " . $mpgResponse->getTransType());
print("\nComplete = " . $mpgResponse->getComplete());
print("\nTransAmount = " . $mpgResponse->getTransAmount());
print("\nCardType = " . $mpgResponse->getCardType());
print("\nTxnNumber = " . $mpgResponse->getTxnNumber());
print("\nTimedOut = " . $mpgResponse->getTimedOut());
print("\nCVDResponse = " . $mpgResponse->getCvdResultCode());
print("\nAVSResponse = " . $mpgResponse->getAvsResultCode());
print("\nResSuccess = " . $mpgResponse->getResSuccess());
print("\nPaymentType = " . $mpgResponse->getPaymentType());
print("\nIssuerId = " . $mpgResponse->getIssuerId());
//----------------- ResolveData ------------------------------
print("\n\nCust ID = " . $mpgResponse->getResDataCustId());
print("\nPhone = " . $mpgResponse->getResDataPhone());
print("\nEmail = " . $mpgResponse->getResDataEmail());
print("\nNote = " . $mpgResponse->getResDataNote());
print("\nMasked Pan = " . $mpgResponse->getResDataMaskedPan());
print("\nExp Date = " . $mpgResponse->getResDataExpDate());
print("\nCrypt Type = " . $mpgResponse->getResDataCryptType());
print("\nAvs Street Number = " . $mpgResponse->getResDataAvsStreetNumber());
print("\nAvs Street Name = " . $mpgResponse->getResDataAvsStreetName());
print("\nAvs Zipcode = " . $mpgResponse->getResDataAvsZipcode());
?>
```

Appendix A Definition of Request Fields – Credential on File

Value	Type	Limits	Description
Issuer ID **NOTE:** This variable is required for all merchant-intiated transactions following the first one; upon sending the first transaction, the Issuer ID value is received in the transaction response and then used in subsequent transaction requests (Issuer ID does not apply for Discover or Union Pay).	String	15-character numeric variable length	Unique identifier for the cardholder's stored credentials Sent back in the response from the card brand when processing a transaction If the cardholder's credentials are being stored for the first time , you must save the Issuer ID on your system to use in subsequent Credential on File transactions
Payment Indicator	String	1-character alphabetic	Indicates the intended or current use of the credentials Possible values for first transactions: C - unscheduled credential on file (first transaction only) R - recurring Possible values for subsequent transactions: R - recurring U - unscheduled merchant-initiated transaction Z - unscheduled cardholder-initiated transaction
Payment Information	String	1-character numeric	Describes whether the transaction is the first or subsequent in the series Possible values are: 0 - first transaction in a series (storing payment details provided by the cardholder)

Value	Type	Limits	Description
			2 - subsequent transactions (using previously stored payment details)

Appendix B Definition of Request Fields – Recurring

Value	Type	Size	Description
Number of Recurs `num_recurs`	String	numeric	The number of times that the trans-action must recur
Period `period`	String	numeric	Number of recur units that must pass between recurring billings
Start Date `start_date`	String	YYYY/MM/DD	Date of the first future recurring billing transaction This value **must** be a date in the future If an additional charge is to be made immediately, the value of Start Now must be set to true

Value	Type	Size	Description
Start Now `start_now`	String	true/false	If a single charge is to be made against the card immediately, set this value to true; the amount to be billed immediately may differ from the amount billed on a regular basis thereafter If the billing is to start in the future, set this value to false When set to false, use Card Verification prior to sending the Purchase with Recur and Credential on File objects
Recurring Amount `recur_amount`	String	9-character decimal Minimum three digits, two of which are penny values	Amount of the recurring transaction This is the amount that will be billed on the Start Date and then billed repeatedly based on the interval defined by Period and Recur Unit
Recur Unit `recur_unit`	String	alphabetic	Unit to be used as a basis for the interval Works in conjunction with Period to define the billing frequency Possible values are: day week month eom (end of month)

Appendix C Definition of Response Fields – Credential on File

Value	Type	Size	Get Method / Description
Issuer ID	String	15-character alpha-numeric	`$mpgResponse->getIssuerId ());` Returned when processing a transaction where the cardholder's credentials are being stored for the first time , and is used as the value for Issuer ID in the requests for subsequent transactions **NOTE:** For Discover and Union Pay transactions, Issuer ID is not returned in the response