Moneris™

BE PAYMENT READY

.NET - 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

```
Purchase purchase = new Purchase();
```

HttpsPostRequest object for Purchase transaction

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(purchase);
```

Purchase transaction values

Table 1: Purchase transaction object mandatory values

Value	Type	Limits	Set method
Order ID	String	50-character alpha-numeric	`purchase.setOrderId(order_id);`
Amount	String	9-character decimal	`purchase.setAmount(amount);`
Credit card number	String	20-character alpha-numeric	`purchase.setPan(pan);`
Expiry date	String	4-character alpha-numeric (YYMM format)	`purchase.setExpDate(expiry_date);`
E-commerce indicator	String	1-character alpha-numeric	`purchase.setCryptType(crypt);`

Table 2: Purchase transaction object optional values

Value	Type	Limits	Set method
Status Check	Boolean	true/false	`mpgReq.setStatusCheck(status_check);`
Customer information	Object	N/A	`purchase.setCustInfo(customer);`
AVS	Object	N/A	`purchase.setAvsInfo(avsCheck);`
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**	Object	N/A	`purchase.setCvdInfo(cvdCheck);`

Table 2: Purchase transaction object optional values

Value	Type	Limits	Set method
store CVD information.			
Convenience fee **NOTE:** This variable does not apply to Credential on File transactions.	Object	N/A	`purchase.setConvenienceFee (convFeeInfo);`
Recurring billing	Object	N/A	`purchase.setRecurInfo (recurInfo);`
Dynamic descriptor	String	20-character alphanumeric	`purchase.setDynamicDescriptor(dynamic_descriptor);`
Wallet indicator[1]	String	3-character alphanumeric	`purchase.setWalletIndicator (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	`purchase.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

```
package Canada;
import JavaAPI.*;
public class TestCanadaPurchase
{
public static void main(String[] args)
{
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String store_id = "store5";
String api_token = "yesguy";
String amount = "5.00";
String pan = "4242424242424242";
String expdate = "1901"; //YYMM format
```

Sample Purchase

```
String crypt = "7";
String processing_country_code = "CA";
boolean status_check = false;
Purchase purchase = new Purchase();
purchase.setOrderId(order_id);
purchase.setAmount(amount);
purchase.setPan(pan);
purchase.setExpdate(expdate);
purchase.setCryptType(crypt);
purchase.setDynamicDescriptor("123456");
//purchase.setWalletIndicator(""); //Refer documentation for possible values

//Optional - Set for Multi-Currency only
//setAmount must be 0.00 when using multi-currency
//purchase.setMCPAmount("500"); //penny value amount 1.25 = 125
//purchase.setMCPCurrencyCode("840"); //ISO-4217 country currency number
//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

purchase.setCofInfo(cof);


HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(purchase);
mpgReq.setStatusCheck(status_check);

//Optional - Proxy
mpgReq.setProxy(false); //true to use proxy
mpgReq.setProxyHost("proxyURL");
mpgReq.setProxyPort("proxyPort");
mpgReq.setProxyUser("proxyUser"); //optional - domainName\User
mpgReq.setProxyPassword("proxyPassword"); //optional
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
```

Sample Purchase

```
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("HostId = " + receipt.getHostId());
System.out.println("MCPAmount = " + receipt.getMCPAmount());
System.out.println("MCPCurrencyCode = " + receipt.getMCPCurrencyCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
```

4.2 Pre-Authorization

Pre-Authorization transaction object definition

```
PreAuth preauth = new PreAuth();
```

HttpsPostRequest object for Pre-Authorization transaction

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(preauth);
```

Pre-Authorization transaction values

Table 3: Pre-Authorization object mandatory values

Value	Type	Limits	Set method
Order ID	String	50-character alpha-numeric	`preauth.setOrderId(order_id);`
Amount	String	9-character decimal	`preauth.setAmount(amount);`
Credit card number	String	20-character numeric	`preauth.setPan(pan);`
Expiry date	String	4-character numeric	`preauth.setExpDate(expiry_date);`
E-Commerce indicator	String	1-character alpha-numeric	`preauth.setCryptType(crypt);`

Table 4: Pre-Authorization object optional values

Value	Type	Limits	Set method
Status Check	Boolean	true/false	`mpgReq.setStatusCheck (status_check);`
Dynamic descriptor	String	20-character alpha-numeric	`preauth.setDynamicDescriptor (dynamic_descriptor);`
Customer information	Object	N/A	`preauth.setCustInfo(customer);`
AVS	Object	N/A	`preauth.setAvsInfo (avsCheck);`
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	`preauth.setCvdInfo (cvdCheck);`

Value	Type	Limits	Set method
Customer ID	String	50-character alpha-numeric	`preauth.setCustId(cust_id);`
Wallet indicator[1]	String	3-character alpha-numeric	`preauth.setWalletIndicator (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	`cof.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

```
package Canada;
import JavaAPI.*;
public class TestCanadaPreauth
{
public static void main(String[] args)
{
String store_id = "store5";
String api_token = "yesguy";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String amount = "5.00";
String pan = "4242424242424242";
String expdate = "1902";
```

Sample Pre-Authorization

```
String crypt = "7";
String processing_country_code = "CA";
boolean status_check = false;
PreAuth preauth = new PreAuth();
preauth.setOrderId(order_id);
preauth.setAmount(amount);
preauth.setPan(pan);
preauth.setExpdate(expdate);
preauth.setCryptType(crypt);
//preauth.setWalletIndicator(""); //Refer documentation for possible values

//Optional - Set for Multi-Currency only
//setAmount must be 0.00 when using multi-currency
//preauth.setMCPAmount("500"); //penny value amount 1.25 = 125
//preauth.setMCPCurrencyCode("840"); //ISO-4217 country currency number
//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

preauth.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(preauth);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
//System.out.println("StatusCode = " + receipt.getStatusCode());
//System.out.println("StatusMessage = " + receipt.getStatusMessage());
System.out.println("MCPAmount = " + receipt.getMCPAmount());
System.out.println("MCPCurrencyCode = " + receipt.getMCPCurrencyCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
}
catch (Exception e)
{
```

Sample Pre-Authorization

```
    e.printStackTrace();
    }
    }
    }
```

4.3 Purchase with Vault – ResPurchaseCC

Purchase with Vault transaction object definition

```
ResPurchaseCC resPurchaseCC = new ResPurchaseCC();
```

HttpsPostRequest object for Purchase with Vault transaction

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(resPurchaseCC);
```

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	`resPurchaseCC.setData(data_key);`
Order ID	String	50-character alpha-numeric	`resPurchaseCC.setOrderId(order_id);`
Amount	String	9-character decimal	`resPurchaseCC.setAmount(amount);`

Value	Type	Limits	Set method
E-commerce indicator	String	1-character alpha-numeric	`resPurchaseCC.setCryptType (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	`cof.setCofInfo(cof);`

Table 6: Purchase with Vault transaction optional values

Value	Type	Limits	Set method
Status Check	Boolean	true/false	`mpgReq.setStatusCheck (status_check);`
Expiry date	String	4-character numeric YYMM format. (Note that this is reversed from the date displayed on the card, which is MMYY)	`resPurchaseCC.setExpDate (expiry_date);`
Customer ID	String	50-character alpha-numeric	`resPurchaseCC.setCustId (cust_id);`
Dynamic descriptor	String	20-character alpha-numeric	`resPurchaseCC.setDy-namicDescriptor(dynamic_descriptor);`
Customer information	Object	N/A	`resPurchaseCC.setCustInfo (customer);`

Value	Type	Limits	Set method
AVS information	Object	N/A	`resPurchaseCC.setAvsInfo (avsCheck);`
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	`resPurchaseCC.setCvdInfo (cvdCheck);`
Recurring billing	Object	N/A	`resPurchaseCC.setRecurInfo (recurInfo);`

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

```
package Canada;
import JavaAPI.*;
public class TestCanadaResPurchaseCC
{
public static void main(String[] args)
{
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String store_id = "store5";
String api_token = "yesguy";
String data_key = "8OOXGiwxgvfbZngigVFeld9d2";
String amount = "1.00";
```

Sample Purchase with Vault

```
String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile will be
    used
String crypt_type = "1";
String descriptor = "my descriptor";
String processing_country_code = "CA";
String expdate = "1512"; //For Temp Token
boolean status_check = false;
ResPurchaseCC resPurchaseCC = new ResPurchaseCC();
resPurchaseCC.setData(data_key);
resPurchaseCC.setOrderId(order_id);
resPurchaseCC.setCustId(cust_id);
resPurchaseCC.setAmount(amount);
resPurchaseCC.setCryptType(crypt_type);
//resPurchaseCC.setDynamicDescriptor(descriptor);
//resPurchaseCC.setExpDate(expdate); //Temp Tokens only
//Mandatory - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

resPurchaseCC.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resPurchaseCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
```

Sample Purchase with Vault

```
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("IssuerId = " + receipt.getIssuerId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
```

4.4 Pre-Authorization with Vault – ResPreauthCC

Pre-Authorization with Vault transaction object definition

```
ResPreauthCC resPreauthCC = new ResPreauthCC();
```

HttpsPostRequest object for Pre-Authorization with Vault transaction

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(resPreauthCC);
```

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	`resPreauthCC.setData(data_key);`
Order ID	String	50-character alpha-numeric	`resPreauthCC.setOrderId(order_id);`
Amount	String	9-character decimal	`resPreauthCC.setAmount(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	`resPreauthCC.setCryptType (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	`resPreauthCC.setCofInfo (cof);`

Table 8: Pre-Authorization with Vault transaction optional values

Value	Type	Limits	Set method
Status Check	Boolean	true/false	`mpgReq.setStatusCheck(status_check);`
Expiry date	String	4-character alpha-numeric (YYMM format)	`resPreauthCC.setExpDate (expiry_date);`
Customer ID	String	50-character alpha-numeric	`resPreauthCC.setCustId(cust_id);`

Value	Type	Limits	Set method
Customer inform-ation	Object	N/A	`resPreauthCC.setCustInfo (customer);`
AVS information	Object	N/A	`resPreauthCC.setAvsInfo (avsCheck);`
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	`resPreauthCC.setCvdInfo (cvdCheck);`

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

```
package Canada;
import JavaAPI.*;
public class TestCanadaResPreauthCC
{
public static void main(String[] args)
{
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String store_id = "store5";
String api_token = "yesguy";
String data_key = "rS7DbroQHJmJxdBfXFXiauQc4";
String amount = "1.00";
```

Sample Pre-Authorization with Vault

```
String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile will be
    used
String crypt_type = "1";
String dynamic_descriptor = "my descriptor";
String processing_country_code = "CA";
String expdate = "1712"; //For Temp Token
boolean status_check = false;
ResPreauthCC resPreauthCC = new ResPreauthCC();
resPreauthCC.setData(data_key);
resPreauthCC.setOrderId(order_id);
resPreauthCC.setCustId(cust_id);
resPreauthCC.setAmount(amount);
resPreauthCC.setCryptType(crypt_type);
resPreauthCC.setDynamicDescriptor(dynamic_descriptor);
//resPreauthCC.setExpDate(expdate); //Temp Tokens only

//Mandatory - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

resPreauthCC.setCofInfo(cof);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resPreauthCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("IsCorporate = " + receipt.getCorporateCard());
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
```

Sample Pre-Authorization with Vault

```
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("IssuerId = " + receipt.getIssuerId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
```

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

```
ResAddToken resAddToken = new ResAddToken();
```

HttpsPostRequest object for Vault Add Token transaction

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(resAddToken);
```

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	`resAddToken.setData(data_key);`
E-commerce indicator	String	1-character alpha-numeric	`resAddToken.setCryptType(crypt);`
Credential on File Info `cof` **NOTE:** This is a nested object within the trans-	Object	N/A	`resaddcc.setCofInfo(cof);`

Value	Type	Limits	Set method
action, 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".			

Table 10: Vault Add Token transaction optional values

Value	Type	Limits	Set method
Customer ID	String	50-character alpha-numeric	`resAddToken.setCustId(cust_id);`
AVS information	Object	N/A	`resAddToken.setAvsInfo(avsCheck);`
Email address	String	30-character alpha-numeric	`resAddToken.setEmail(email);`
Phone number	String	30-character alpha-numeric	`resAddToken.setPhone(phone);`
Note	String	30-character alpha-numeric	`resAddToken.setNote(note);`
Data key format[1]	String	2-character alpha-numeric	`resAddToken.setDataKeyFormat(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

```
package Canada;
import JavaAPI.*;
public class TestCanadaResAddToken
{
public static void main(String[] args)
{
String store_id = "store1";
String api_token = "yesguy";
String data_key = "ot-545454ucx87A5454";
String expdate = "2001";
String phone = "0000000000";
String email = "bob@smith.com";
String note = "my note";
```

Sample Vault Add Token

```
String cust_id = "customer1";
String crypt_type = "7";
String data_key_format = "0";
String processing_country_code = "CA";
boolean status_check = false;
AvsInfo avsCheck = new AvsInfo();
avsCheck.setAvsStreetNumber("212");
avsCheck.setAvsStreetName("Payton Street");
avsCheck.setAvsZipCode("M1M1M1");

//Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");
ResAddToken resAddToken = new ResAddToken();
resAddToken.setDataKey(data_key);
resAddToken.setCryptType(crypt_type);
resAddToken.setExpdate(expdate);
resAddToken.setCustId(cust_id);
resAddToken.setPhone(phone);
resAddToken.setEmail(email);
resAddToken.setNote(note);
resAddToken.setAvsInfo(avsCheck);
resAddToken.setCofInfo(cof);
//resAddToken.setDataKeyFormat(data_key_format); //optional
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resAddToken);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("MaskedPan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
}
catch (Exception e)
{
```

Sample Vault Add Token

```
    e.printStackTrace();
    }
    }
    }
```

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

```
ResUpdateCC resUpdateCC = new ResUpdateCC();
```

HttpsPostRequest object for Vault Update Credit Card transaction

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(resUpdateCC);
```

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	`resUpdateCC.setData(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	`resUpdateCC.setPan(pan);`
Expiry date	String	4-character alpha-numeric (YYMM format)	`resUpdateCC.setExpDate(expiry_date);`
E-commerce indicator	String	1-character alpha-numeric	`resUpdateCC.setCryptType(crypt);`
Customer ID	String	50-character alpha-numeric	`resUpdateCC.setCustId(cust_id);`
AVS information	Object	n/a	`resUpdateCC.setAvsInfo(avsCheck);`
Email address	String	30-character alpha-numeric	`resUpdateCC.setEmail(email);`
Phone number	String	30-character alpha-	`resUpdateCC.setPhone(phone);`

Value	Type	Limits	Set method
		numeric	
Note	String	30-character alpha-numeric	`resUpdateCC.setNote(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	`resUpdateCC.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

```
package Canada;
import JavaAPI.*;
public class TestCanadaResUpdateCC
{
public static void main(String[] args)
{
String store_id = "moneris";
String api_token = "hurgle";
String data_key = "vthBJyN1BicbRkdWFZ9flyDP2";
String pan = "4242424242424242";
String expdate = "1901";
String phone = "0000000000";
String email = "bob@smith.com";
```

Sample Vault Update Credit Card

```
String note = "my note";
String cust_id = "customer1";
String crypt_type = "7";
String processing_country_code = "CA";
boolean status_check = false;
AvsInfo avsCheck = new AvsInfo();
avsCheck.setAvsStreetNumber("212");
avsCheck.setAvsStreetName("Payton Street");
avsCheck.setAvsZipCode("M1M1M1");
//Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

ResUpdateCC resUpdateCC = new ResUpdateCC();
resUpdateCC.setData(data_key);
resUpdateCC.setAvsInfo(avsCheck);
resUpdateCC.setCustId(cust_id);
resUpdateCC.setPan(pan);
resUpdateCC.setExpdate(expdate);
resUpdateCC.setPhone(phone);
resUpdateCC.setEmail(email);
resUpdateCC.setNote(note);
resUpdateCC.setCryptType(crypt_type);
resUpdateCC.setCofInfo(cof);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resUpdateCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("MaskedPan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
}
catch (Exception e)
{
```

Sample Vault Update Credit Card

e.printStackTrace();
}
}
}

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

```
ResAddCC resaddcc = new ResAddCC();
```

HttpsPostRequest object for ResAddCC transaction

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(resaddcc);
```

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	`resaddcc.setPan(pan);`
Expiry date	String	4-character alpha-numeric (YYMM format)	`resaddcc.setExpDate(expiry_date);`
E-commerce indicator	String	1-character alpha-numeric	`resaddcc.setCryptType(crypt);`
Credential on File Info `cof` **NOTE:** This is a nested	Object	N/A	`resaddcc.setCofInfo(cof);`

Value	Type	Limits	Set method
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".			

Table 14: Vault Add Credit Card transaction optional values

Value	Type	Limits	Set method
Customer ID	String	50-character alphanumeric	`resaddcc.setCustId(cust_id);`
AVS information	Object	N/A	`resaddcc.setAvsInfo (avsCheck);`
Email address	String	30-character alphanumeric	`resaddcc.setEmail(email);`
Phone number	String	30-character alphanumeric	`resaddcc.setPhone(phone);`
Note	String	30-character alphanumeric	`resaddcc.setNote(note);`
Data key format[1]	String	2-character alphanumeric	`resaddcc.setDataKeyFormat (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

```
package Canada;
import JavaAPI.*;
public class TestCanadaResAddCC
{
public static void main(String[] args)
{
String store_id = "store5";
String api_token = "yesguy";
String pan = "4242424242424242";
String expdate = "1912";
String phone = "0000000000";
String email = "bob@smith.com";
String note = "my note";
```

Sample Vault Add Credit Card

```
String cust_id = "customer1";
String crypt_type = "7";
String data_key_format = "0";
String processing_country_code = "CA";
boolean status_check = false;
AvsInfo avsCheck = new AvsInfo();
avsCheck.setAvsStreetNumber("212");
avsCheck.setAvsStreetName("Payton Street");
avsCheck.setAvsZipCode("M1M1M1");
ResAddCC resaddcc = new ResAddCC();
resaddcc.setPan(pan);
resaddcc.setExpdate(expdate);
resaddcc.setCryptType(crypt_type);
resaddcc.setCustId(cust_id);
resaddcc.setPhone(phone);
resaddcc.setEmail(email);
resaddcc.setNote(note);
resaddcc.setAvsInfo(avsCheck);
//resaddcc.setDataKeyFormat(data_key_format); //optional
//Mandatory - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9"); //can be obtained by performing card verification

resaddcc.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resaddcc);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("MaskedPan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("IssuerId = " + receipt.getIssuerId());
}
```

Sample Vault Add Credit Card

```
catch (Exception e)
{
e.printStackTrace();
}
}
}
```

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

```
CardVerification cardVerification = new CardVerification();
```

HttpsPostRequest object for Card Verification transaction

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(cardVerification );
```

Card Verification transaction values

Table 15: Card Verification transaction object mandatory values

Value	Type	Limits	Set method
Order ID	String	50-character alpha-numeric	`cardVerification.setOrderId (order_id);`
Credit card number	String	20-character alpha-numeric	`cardVerification.setPan (pan);`
Expiry date	String	4-character alpha-numeric (YYMM format)	`cardVerification.setExpDate (expiry_date);`
E-commerce indicator	String	1-character alpha-numeric	`cardVerification .setCryptType(crypt);`
AVS	Object	N/A	`cardVerification.setAvsInfo (avsCheck);`
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	`cardVerification.setCvdInfo (cvdCheck);`

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	`cardVerification.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

```
package Canada;
import JavaAPI.*;
public class TestCanadaCardVerification
{
public static void main(String[] args)
{
String store_id = "store5";
String api_token = "yesguy";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String pan = "4242424242424242";
String expdate = "1901"; //YYMM format
String crypt = "7";
```

Sample Card Verification

```
String processing_country_code = "CA";
boolean status_check = false;
AvsInfo avsCheck = new AvsInfo();
avsCheck.setAvsStreetNumber("212");
avsCheck.setAvsStreetName("Payton Street");
avsCheck.setAvsZipCode("M1M1M1");
CvdInfo cvdCheck = new CvdInfo();
cvdCheck.setCvdIndicator("1");
cvdCheck.setCvdValue("099");
CardVerification cardVerification = new CardVerification();
cardVerification.setOrderId(order_id);
cardVerification.setPan(pan);
cardVerification.setExpdate(expdate);
cardVerification.setCryptType(crypt);
cardVerification.setAvsInfo(avsCheck);
cardVerification.setCvdInfo(cvdCheck);

//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

cardVerification.setCofInfo(cof);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(cardVerification);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("IssuerId = " + receipt.getIssuerId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
```

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

```
ResCardVerificationCC resCardVerificationCC = new ResCardVerificationCC();
```

HttpsPostRequest object for Card Verification transaction

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(resCardVerificationCC);
```

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	`resCardVerificationCC .setOrderId(order_id);`
Data key	String	25-character alpha-numeric	`resCardVerificationCC .setDataKeyFormat(data_key_ format);`
E-commerce indicator	String	1-character alpha-numeric	`resCardVerificationCC .setCryptType(crypt);`

Table 17: Card Verification with Vault transaction object mandatory values

Value	Type	Limits	Set method
AVS	Object	N/A	`resCardVerificationCC .setAvsInfo(avsCheck);`
CVD	Object	N/A	`resCardVerificationCC .setCvdInfo(cvdCheck);`
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	`resCardVerificationCC .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

```
package Canada;
import java.io.*;
import JavaAPI.*;
public class TestCanadaResCardVerificationCC
{
public static void main(String args[]) throws IOException
{
String store_id = "store5";
String api_token = "yesguy";
String data_key = "AoG4zAFzlFFfxcVmzWAZVQuhj";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String crypt_type = "7";
```

Sample Card Verification with Vault

```
String processing_country_code = "CA";
boolean status_check = false;

/********************** Efraud Variables ************************/
AvsInfo avs = new AvsInfo ();
avs.setAvsStreetName("test ave");
avs.setAvsStreetNumber("123");
avs.setAvsZipcode("123456");
CvdInfo cvd = new CvdInfo ("1", "099");
/*********************** Transaction Object *******************************/
ResCardVerificationCC resCardVerificationCC = new ResCardVerificationCC();
resCardVerificationCC.setDataKey(data_key);
resCardVerificationCC.setOrderId(order_id);
resCardVerificationCC.setCryptType(crypt_type);
resCardVerificationCC.setAvsInfo(avs);
resCardVerificationCC.setCvdInfo(cvd);
//resCardVerificationCC.setExpdate("1412"); //For Temp Tokens only

//Mandatory - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

resCardVerificationCC.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resCardVerificationCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
/************************ Receipt Object ******************************/
try
{
Receipt resreceipt = mpgReq.getReceipt();
System.out.println("DataKey = " + resreceipt.getDataKey());
System.out.println("ReceiptId = " + resreceipt.getReceiptId());
System.out.println("ReferenceNum = " + resreceipt.getReferenceNum());
System.out.println("ResponseCode = " + resreceipt.getResponseCode());
System.out.println("AuthCode = " + resreceipt.getAuthCode());
System.out.println("ISO = " + resreceipt.getISO());
System.out.println("Message = " + resreceipt.getMessage());
System.out.println("TransDate = " + resreceipt.getTransDate());
System.out.println("TransTime = " + resreceipt.getTransTime());
System.out.println("TransType = " + resreceipt.getTransType());
System.out.println("Complete = " + resreceipt.getComplete());
System.out.println("TransAmount = " + resreceipt.getTransAmount());
System.out.println("CardType = " + resreceipt.getCardType());
System.out.println("TxnNumber = " + resreceipt.getTxnNumber());
System.out.println("TimedOut = " + resreceipt.getTimedOut());
System.out.println("ResSuccess = " + resreceipt.getResSuccess());
System.out.println("PaymentType = " + resreceipt.getPaymentType() + "\n");
System.out.println("IssuerId = " + resreceipt.getIssuerId());
//Contents of ResolveData
System.out.println("Cust ID = " + resreceipt.getResCustId());
System.out.println("Phone = " + resreceipt.getResPhone());
```

Sample Card Verification with Vault

```
System.out.println("Email = " + resreceipt.getResEmail());
System.out.println("Note = " + resreceipt.getResNote());
System.out.println("Masked Pan = " + resreceipt.getResMaskedPan());
System.out.println("Exp Date = " + resreceipt.getResExpdate());
System.out.println("Crypt Type = " + resreceipt.getResCryptType());
System.out.println("Avs Street Number = " + resreceipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + resreceipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + resreceipt.getResAvsZipcode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResCardVerificationCC
```

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	`receipt.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