1 Purchase

Verifies funds on the customer's card, removes the funds and prepares them for deposit into the merchant's account.

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 **order_id**	String	50-character alpha-numeric	`purchase.setOrderId(order_id);`
Amount	String	10-character decimal Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	`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);`
Card Match ID **NOTE:** Applies to Off-linx™ only; must be unique value for each transaction	*String*	50-character alpha-numeric	`purchase.setCmId(transaction_id);`
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 store CVD information**.	Object	N/A	`purchase.setCvdInfo(cvdCheck);`
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);`

Table 2: Purchase transaction object optional values

Value	Type	Limits	Set method
Dynamic descriptor	String	20-character alpha-numeric	`purchase.setDynamicDescriptor (dynamic_descriptor);`
Wallet indicator[1] **NOTE:** For basic Purchase and Preauthorization, the wallet indicator applies to Visa Checkout and MasterCard MasterPass only. For more, see Appendix A Definition of Request Fields.	String	3-character alpha-numeric	`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 Fields

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.	*String*	15-character alphanumeric 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

```
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
String crypt = "7";
String processing_country_code = "CA";
boolean status_check = false;
Purchase purchase = new Purchase();
```

Sample 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
purchase.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max 50
alphanumeric characters transaction id generated by merchant
//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());
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)
{
```

Sample Purchase

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