BE PAYMENT READY

Google Pay™ - Merchant Integration Guide

Version: 1.2.0

Table of Contents

System and Skills Requirements

- Android Studio Version 2.2 or higher
- Knowledge of Java
- Android OS Kit Kat (4.4) or higher
- Refer to Google Pay documentation for information on supported browsers
- Your Store ID and API token (testing and production) from Moneris

Changes in v1.2.0

- Added Preload transaction type for Google Pay™ Web
- Added additional information to indicate the amount request field represents transaction amount after shipping is calculated
- Added change log to document
- Added information about follow-on transaction types

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/

1 Getting Started With Google Pay™

In order to integrate your Google Pay™ payment solution with Moneris, there are a few basic tasks you need to do:

For Google Pay™ In-App, follow these steps:

1. Build a demo application for Google Pay™ In-App SDK
2. Customize your demo app's code to work with the Moneris Gateway
3. Compile and run your application
4. Verify the transactions using the Merchant Resource Center.

For Google Pay™ Web, follow these steps:

1. Build a demo web checkout page for Google Pay™Web
2. Customize your demo site's code to work with the Moneris Gateway
3. Compile and run your application
4. Verify the transactions using the Merchant Resource Center.

For additional information on Google Pay™ development, consult Google's developer site:

In-App (Android): https://developers.google.com/pay/api/android/

Web: https://developers.google.com/pay/api/web/guides/tutorial

1.1 Getting Started – Google Pay™ In-App

1.1.1 Building a Google Pay™ Demo App

A demo checkout application is required as part of the integration process for Google Pay™. Google provides code for a Google Pay™ demo application to make it easy.

To build a Google Pay demo application:

1. Go to Google's Google Pay quick start page on Github at https://github.com/google-pay/android-quickstart
2. Git clone the Android-Quickstart library to your computer
3. Unzip the project library
4. Open Android Studio and import this project as a gradle project
5. In the Project tab, in the project's build.gradle, add the following in the repositories block after jcenter():

```
allprojects {
    repositories {
        google()
        jcenter()
        maven {
            url "https://github.com/Moneris/eCommerce-GooglePay-SDK/raw/-
master"
        }
    }
}
```

6. In the build.gradle for the app module, add the following line:

```
dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.google.android.gms:play-services-wallet:16.0.0'
    implementation 'com.moneris.googlepay:googlepayapi:1.2.0'
}
```

7. Sync the gradle

1.1.2 Integrating Your Demo App with Moneris Gateway

To begin testing your integration, you need to modify code in your Google Pay™ demo application project to enable it to communicate with the Moneris Gateway.

To integrate your demo app with the Moneris Gateway, do the following:

1. In the file **Constants.java**,
   a. set the PAYMENT_GATEWAY_TOKENIZATION_NAME to "moneris"
   b. change **exampleGatewayMerchantId** to your testing Store ID
2. In the file **CheckoutActivity.java**
   a. in the section LOAD_PAYMENT_DATA_REQUEST_CODE of onActivityResult, replace this code

```
PaymentData paymentData = PaymentData.getFromIntent(data);
handlePaymentSuccess(paymentData);
```

   with the following field after the PaymentData is received:

```
PaymentData paymentData = PaymentData.getFromIntent(data);
try {
       com.moneris.googlepay.api.Recur recurInfo = new Recur.Builder()
       // Amount to be sent when managed recur transaction is triggered
       .setRecurAmount("1.00")
       // First day the recurring starts
       .setStartDate("2018/09/01")
       // true to charge the txn amount now or false to register the sequence
       .setStartNow("false")
       // Number of recurs
       .setNumRecurs("5")
       // The period for the recur, can be "day", "week", "month", or "eom"
       .setPeriod("day")
       // Generate the recur object
       .build();
```

```
        com.moneris.googlepay.api.Purchase purchase =
                 (com.moneris.googlepay.api.Purchase)
                 new com.moneris.googlepay.api.Request.Builder()
        // Mandatory
        .setTransactionType(TransactionCode.PURCHASE)
        // Mandatory, must be unique for all transactions
        .setOrderId("GooglePayTest-"+new Date().getTime())
        // Mandatory, data returned from Google
        .setPaymentData(paymentData)
        // Transaction amount processed
        .setAmount(
         PaymentsUtil.microsToString(mBikeItem.getPriceMicros()+mShippingCost)
         )
        // Optional, only apply to recurring registration
        .setRecur(recurInfo)
        // Optional, to display Customer Id for transaction
        .setCustId("Customer 1")
        // Generate the transaction object
        .build();

        // Transaction transport structure
        com.moneris.googlepay.api.MonerisHttpsPost monerisHttpsPost =
                 new com.moneris.googlepay.api.MonerisHttpsPost.Builder()
        // Mandatory, should be securely stored.
        .setStoreId(
                 PAYMENT_GATEWAY_TOKENIZATION_PARAMETERS.get("gatewayMerchantId")
                 )
        // Mandatory, should be securely stored.
        .setApiToken("spedguy")
        // Mandatory from line above
        .setRequest(purchase)
        // Optional, default is testMode false;
        .setTestMode(true)
        // Generate transport object
        .build();

  // Send transaction
  com.moneris.googlepay.api.Response resp = monerisHttpsPost.send();

  // Simple test to check if transaction is approved.
  int numRespCode = Integer.parseInt(resp.getResponseCode());
  if ( numRespCode < 50 )
  {
        // Approved complete the transaction.
        handlePaymentSuccess(paymentData);
  }
  else
  {
        // Declined transaction
        handleError(numRespCode);
  }
}
catch (Exception ex)

{
// send -1 for a catch all error
  Log.e("ERROR", "Failed to send transaction with error: "+ex.getMessage());
  handleError(-1);
}
```

b. in the code snippet above, change the **exampleApiToken** value to your testing API token

1.2 Getting Started – Google Pay™ Web

1.2.1 Preload Request

Requests a unique ticket number from the Moneris Gateway server, which responds with the ticket number to your back-end server at the URL provided.

The Preload request is sent prior to sending a financial transaction in Apple Pay on the Web and Google Pay™ Web.

If no response is received within 15 seconds, the request will time out.

Preload transaction request URL for Moneris Gateway

Testing: https://esqa.moneris.com/gateway2/servlet/MpgRequest

Production: https://www3.moneris.com/gateway2/servlet/MpgRequest

Preload transaction request object

<googlepay_preload>

Connection fields – Required

Variable Name	Type and Limits	Description
store ID <store_id>	*String* N/A	Unique identifier provided by Moneris upon merchant account setup
API token <api_token>	*String* N/A	Unique alphanumeric string assigned by Moneris upon merchant account activation To find your API token, refer to your test or production store's Admin settings in the Merchant Resource Center, at the following URLs: Testing: https://esqa.moneris.com/mpg/ Production: https://www3.moneris.com/mpg/

Preload transaction request fields – Required

Variable Name	Type and Limits	Description
order ID <order_id>	*String* 50-character alphanumeric a-Z A-Z 0-9 _ - : . @ spaces	Merchant-defined transaction identifier that must be unique for every Purchase, Pre-Authorization and Independent Refund transaction. No two transactions of these types may have the same order ID. For Refund, Completion and Purchase Correction transactions, the order ID must be the same as that of the original transaction.
amount <amount>	*String* 10-character decimal Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	Transaction dollar amount This must contain at least 3 digits, two of which are penny values Minimum allowable value = $0.01, maximum allowable value = $9999999.99 **NOTE:** This amount is the total transaction amount after shipping is calculated
receipt URL <receipt_url>	*String*	URL that the Moneris Gateway will send your receipt to

1.2.2 Building a Google Pay™ Web Demo Checkout Page

A demo website checkout page is required as part of the integration process for Google Pay™ Web. Google provides code for a Google Pay™-enabled demo checkout page to make it easy.

To build a Google Pay™ Web demo checkout page:

1. Go to Google's developer tutorial page for Google Pay Web at https://developers.google.com/pay/api/web/guides/tutorial
2. Copy the code shown under the subheading "Put it all together" and paste it into your code editor

1.2.3 Integrating Your Demo Checkout Page With Moneris Gateway

To begin testing your integration, you need to modify code in your Google Pay™ Web demo checkout page project to enable it to communicate with the Moneris Gateway.

To integrate your demo checkout page with the Moneris Gateway, do the following:

1. In the code for your demo checkout page, in the object **tokenizationSpecification**:
   a. Set the parameter `gateway` to the value to 'moneris'
   b. Set `gatewayMerchantId` to your test store ID value
2. After the initial `div`, add the following Monerisscript tag:
   Testing/QA

   ```
   <script async src="https://esqa.moneris.com/googlepay/googlepay-api.js"
   onload="MonerisGooglePay.onReady()"></script>
   ```

   Production

   ```
   <script async src="https://www3.moneris.com/googlepay/googlepay-api.js"
   onload="MonerisGooglePay.onReady()"></script>
   ```

3. Below that script, add a `div` with the id "`moneris-google-pay`", substituting your test values for store ID and web merchant key in place of the values for `store-id` and `web-merchant-key` as shown below:
   ```
   <div
   id="moneris-google-pay"
   store-id="MONERIS-GATEWAY-STORE-ID"
   web-merchant-key="WEB-MERCHANT-KEY"></div>
   ```

4. Insert Google's script in your web page code:
   ```
   <script async="" src="https://pay.google.com/gp0/p/js/pay.js"
   onload="onGooglePayLoaded()"></script>
   ```

5. Change the code for the function **processPayment**to the following:
   ```
   function processPayment(paymentData) {
          // show returned data in developer console for debugging

          console.log(paymentData);

          const respDiv = document.getElementById("google-pay-resp");
          const uiDiv = document.getElementById("google-pay-ui");

          uiDiv.classList.add("none");

          respDiv.innerText = "Processing Payment...Please wait";

          // @todo pass payment data response to your gateway to process payment
          /* Moneris-specifics */

          var d = Date.now();
          paymentData["ticket"] = "gp1578937300vPiHwPJxQsDtH0bSz6";

          /* Moneris-specifics */
          MonerisGooglePay.purchase(paymentData, function(response) {

                  var ctr = "AuthCode: " + response.receipt.AuthCode + "\n" +
                  "BankTotals: " + response.receipt.BankTotals + "\n" +
                  "CardType: " + response.receipt.CardType + "\n" +
                  "Complete: " + response.receipt.Complete + "\n" +
                  "CorporateCard: " + response.receipt.CorporateCard + "\n" +
                  "ISO: " + response.receipt.ISO + "\n" +
                  "IsVisaDebit: " + response.receipt.IsVisaDebit + "\n" +
   ```

```
                "Message: " + response.receipt.Message + "\n" +
                "ReceiptId: " + response.receipt.ReceiptId + "\n" +
                "ReferenceNum: " + response.receipt.ReferenceNum + "\n" +
                "ResponseCode: " + response.receipt.ResponseCode + "\n" +
                "Ticket: " + response.receipt.Ticket + "\n" +
                "TimedOut: " + response.receipt.TimedOut + "\n" +
                "TransAmount: " + response.receipt.TransAmount + "\n" +
                "TransDate: " + response.receipt.TransDate + "\n" +
                "TransID: " + response.receipt.TransID + "\n" +
                "TransTime: " + response.receipt.TransTime + "\n" +
                "TransType: " + response.receipt.TransType + "\n";

                if ( response && response.receipt && response.receipt.ResponseCode && !isNaN
(response.receipt.ResponseCode) )
                {
                        if ( parseInt(response.receipt.ResponseCode) < 50 ) {

                        //alert("Looks like the transaction is approved!!! \n\n" + ctr);
                }
                else {
                        //alert("Looks like the transaction is declined!!! \n\n" + ctr);
                }
                }
                else {
                        //alert("Looks like an error!!! \n\n" + ctr);
                        //throw ("Error processing receipt!!! \n\n" + ctr);
                }

                const respDiv = document.getElementById("google-pay-resp");
                respDiv.innerText = "" + JSON.stringify(response);
});
```

2 Building Transactions in Google Pay™

2.1 Google Pay™ Transaction Types

The Moneris Gateway can process the following transaction types via integration with the Google Pay™ In-App and Google Pay™ Web SDK:

- Google Pay Purchase
- Google Pay Pre-Authorization

Once you have processed the initial transaction using Google Pay™ Purchase or Google Pay™ Pre-Authorization, you can use the Moneris Gateway Unified API or Batch Upload solution to process one of the following supplemental transactions:

- Refund
- Purchase Correction (not applicable to INTERAC® e-Commerce transactions)
- Pre-Authorization Completion

Information about the Unified API and Batch Upload are available on the Moneris Developer Portal at

developer.moneris.com

Transactions are carried out by automated builders according to Google's current development standards for applications.

- The transaction request builder will build out Google Pay Purchase and Google Pay Pre-Authorization transactions
- The Moneris httpspost builder performs the necessary communications between the Moneris Gateway and Google Pay™.

2.2 Transaction Request Builder

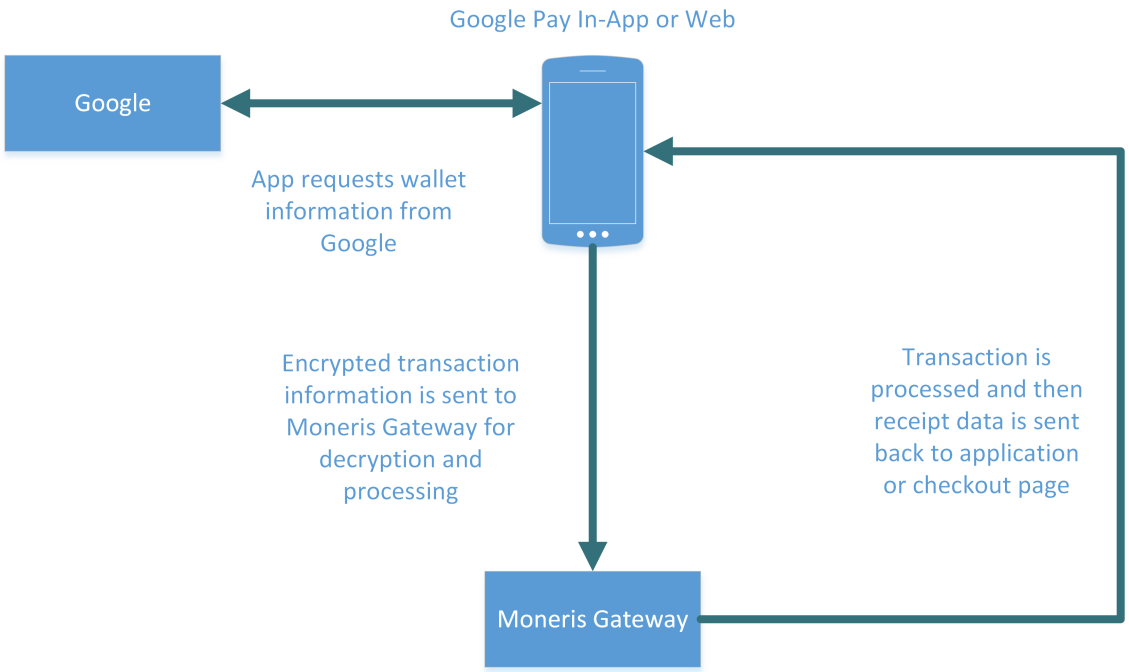
Transaction requests in Google Pay are built by Google's transaction request builder. Below are the request variables which get built into the transaction request.

Table 1 Transaction Request Variables – Required Fields

Variable	Set Method	Description
Transaction Type	`.setTransactionType (Constants.TransactionCode.PURCHASE)`	The type of transaction being sent Possible values: PURCHASE PREAUTH

Variable	Set Method	Description
Order ID	`.setOrderId("GooglePayTest-"+new Date().getTime()`	Merchant-defined transaction identifier that must be unique for every Google Pay Purchase and Google Pay Pre-Authorization transaction No two transactions of these types may have the same order ID
Payment Data	`.setPaymentData(data)`	Object representing the payment data response from Google, which contains the necessary payment result to complete the payment For more on this, see Google's APIs for Android documentation
Amount	`.setAmount(PaymentsUtil.microsToString(mBikeItem.getPriceMicros()+mShippingCost))`	Transaction amount

2.3 Google Pay™ Transaction Process Flow



3 Testing Your Solution - Google Pay™

3.1 Testing Your Google Pay™ In-App Integration

Once you have built your Google Pay™ demo app, you can test your Google Pay™ In-App integration with the Moneris Gateway.

For testing, you will need to have:

- Google Wallet with a valid payment card added to the wallet
- Test API token and store ID from Moneris

3.2 Testing Your Google Pay™ Web Integration

Once you have built your Google Pay™ demo checkout page, you can test your Google Pay™ Web integration with the Moneris Gateway.

For testing, you will need to have:

- Google Wallet with a valid payment card added to the wallet
- Test API token and store ID from Moneris
- Google Pay™ web merchant key from Moneris
- Test version of Google Merchant ID from Google

3.3 Getting a Unique Test Store ID and API Token

Transactions requests via the Moneris Gateway API will require you to have a Store ID and a corresponding API token.

For testing purposes, you can either use the pre-existing test stores with the corresponding test API tokens, or you can create your own unique test API token and a unique test store where you will only see your own transactions.

To get your unique Store ID and API token for testing:

1. Log in to the Developer Portal at https://developer.moneris.com
2. In the My Profile dialog, click the **Full Profile** button
3. Under My Testing Credentials, select **Request Testing Credentials**
4. Enter your Developer Portal password and select your country
5. Record the Store ID and API token that are given, as you will need them for logging in to the Merchant Resource Center (Store ID) and for API requests (API token).

Alternatively, you can use the pre-existing test stores already set up in the Merchant Resource Center as described in 3.4 Test Store Credentials.

For production, you will use the Store ID given to you in your Moneris activation letter and an API token retrieved from the production Merchant Resource Center. For more on this, see 4.4 Getting a Production Store ID and API Token.

3.4 Test Store Credentials

For testing purposes, you can either use the pre-existing test stores with the corresponding test API tokens, or you can create your own unique test API token and a unique test store where you will only see your own transactions. If you want to use pre-existing stores, use the test credentials provided in the following tables.

Table 1: Test Server Credentials - Canada

Store ID	API Token	MRC Username	MRC Password
store1	yesguy	demouser	password
store2	yesguy	demouser	password
store3	yesguy	demouser	password
store4	yesguy	demouser	password
store5	yesguy	demouser	password

Alternatively, you can create and use a unique test store where you will only see your own transactions. For more on this, see 3.3 Getting a Unique Test Store ID and API Token

3.5 Getting the Web Merchant Key for Google Pay™ Web

You can find your Google Pay Web Merchant Key in the Moneris Merchant Resource Center under Admin > Google Pay at:

Testing: https://esqa.moneris.com/mpg
Production: https://www3.moneris.com/mpg

Use the Web Merchant Key that corresponds to whichever stage of development you are in (testing vs. production).

3.6 Getting a Test Google Pay Web Merchant ID

Google provides a placeholder merchant ID for Google Pay™ Web for use in testing. For testing purposes, send the merchant ID as follows:

```
'merchantId' : '01234567890123456789'
```

For production, you receive your own unique Google Pay™ merchant ID after Google approves your production access. For more on this subject, see 4.3 Requesting Production Access from Google.

4 Moving to Production for Google Pay™

4.1 Moving to Production – Google Pay™ In-App

In order to move your Google Pay™ In-App SDK solution into production, gather the following credentials:

- Your **production Store ID**, given in the activation letter sent to you by Moneris
- Your **production API token**, obtained in the production Merchant Resource Center. To learn how, see 4.4 Getting a Production Store ID and API Token

Once you have these credentials, the next step is to configure your production store to work with the Moneris Gateway. To learn how, see 4.5 Configuring Your Store for Production – In-App.

4.2 Moving to Production - Google Pay™ Web

In order to move your Google Pay™ Web solution into production, gather the following credentials:

- Your **production Store ID**, given in the activation letter sent to you by Moneris
- Your **production API token**, obtained in the production Merchant Resource Center. To learn how, see 4.4 Getting a Production Store ID and API Token
- **Production access from Google**. To learn how, see 4.3 Requesting Production Access from Google

4.3 Requesting Production Access from Google

Once you have completed testing your Google Pay integration, you must request production access from Google. Google will evaluate your integration against their integration checklist.

Both the integration checklist and the production access request are found on Google's developer site at:

In-App

https://developers.google.com/pay/api/android/guides/test-and-deploy/integration-checklist

Web

https://developers.google.com/pay/api/web/guides/test-and-deploy/integration-checklist

4.4 Getting a Production Store ID and API Token

In production, you use the Store ID that was given in your activation letter from Moneris. You obtain the production API token from the production Merchant Resource Center.

To get your production API token:

1. If you have not already done so, activate your production account at

   https://www.moneris.com/activate

   The activation process provides you with your first administrator user for the Merchant Resource Center.

2. Once activated, log in to the production Merchant Resource Center at

   https://www3.moneris.com/mpg

3. Select the **Admin** menu and choose **Settings**. Your production API token is located under the API token heading on the page.

4.5 Configuring Your Store for Production – In-App

Once you have completed your testing you are ready to point your store to the production host. You do this by changing the values of your Store ID and API token from their testing values to the production values.

To configure your store for production:

1. In the file **Constants.java**,
   a. change **exampleGatewayMerchantId** to your production Store ID
2. In the file **CheckoutActivity.java**,
   a. change the **exampleApiToken** value to your production API token
   b. change **setTestMode** to `(false)`

4.6 Configuring Your Store for Production – Web

In the code for your demo Google Pay™ checkout page, change testing values to production values:

1. In the object **tokenizationSpecification**, set the parameter `gatewayMerchantId` to your production store ID
2. In the `div` with the id "`moneris-google-pay`", change the values for `store-id` and `web-merchant-key` to your production store ID and production web merchant key

5 Verifying Your Transactions

When you send a Preload transaction, you must include a transaction amount and a receipt URL, hosted on your back-end server. The Moneris Gateway sends the transaction response to your receipt URL so that you can verify the amount you sent in the transaction request matches the one coming back in the response from the Moneris Gateway.

As a secondary check, you can also manually look at your transactions in the Moneris Merchant Resource Center.

To use the Merchant Resource Center to verify your transactions have processed:

1. Log in to the Merchant Resource Center at https://esqa.moneris.com/mpg (testing) or https://www3.moneris.com/mpg (production)
2. Find your transactions under **Reports > Transactions**

Appendix A Definition of Response Fields

Variable Name	Type and Limits	Description
AuthCode	*String* 8-character alphanumeric	Authorization code returned from the issuing institution
Bank Totals	*Object* N/A	Response data returned in a Batch Close and Open Totals request.
CardType	*String* 2-character alphanumeric	Credit Card Type M = Mastercard V = Visa AX = American Express P = INTERAC®
CAVV	*String* 40-character alphanumeric	Decrypted CAVV value for the transaction Returned for Google Pay™ Purchase/Pre-Authorization transaction if payload is successfully decrypted
Complete	*String* true/false	Transaction was sent to authorization host and a response was received
CorporateCard	*String* true/false	Indicates whether the card is a corporate card or not
ISO	*String* 2-character numeric	ISO response code
IsVisaDebit	*String* true/false/null	Indicates whether the card that the transaction was performed on is Visa debit true = Card is Visa Debit false = Card is not Visa Debit null = there was an error in identifying the card

Variable Name	Type and Limits	Description
Message	*String* 100-character alpha-numeric	Response description returned from issuing institution
ReceiptId	*String* 50-character alphanumeric	The order id specified in the request will be echoed back in the response. This field is recommended to be displayed on the receipt for tracking and troubleshooting purposes.
RecurSuccess	*String* true/false	Indicates whether the transaction successfully registered
ReferenceNum	*String* 18-character numeric	This is a bank transaction reference number. The entire reference number must be displayed on the receipt. This information should also be stored by the merchant. The following illustrates the breakdown of this field where "660123450010690030" is the reference number returned in the message. **EXAMPLE** 660123450010690030 • 66012345: Terminal ID • 001: Shift number • 069: Batch number • 003: Transaction number within the batch
ReponseCode	*String* 3-character numeric	Transaction Response Code < 50: Transaction approved >= 50: Transaction declined NULL: Transaction was not sent for authorization Custom Google Pay™ Response Code 900 : Global Error means that Moneris Gatewaywas unable to decrypt payload.

Variable Name	Type and Limits	Description
Ticket	N/A	Reserved field
TimedOut	*String* true/false	Transaction failed due to a process timing out
TransAmount	*String* nnnnnnN.NN 9-character decimal (variable length)	Returns the amount sent in request for processing. The amount represents the amount that the cardholder was charged/refunded. The amount must be displayed on the receipt. **NOTE:** The amount will always contain one (1) dollar value and two (2) cent values separated by a period ".". N = always returned n = returned when required
TransDate	*String* YYYY-MM-DD	Processing host date stamp. Date of the transaction. Must be displayed on the transaction receipt. YYYY = 4-digit year MM = 2-digit month ("0" left padded | Jan = 01) DD = 2 digit day of month ("0" left padded)
TransID	*String* 20-character alphanumeric	Gateway Transaction identifier
TransTime	*String* HH:II:SS	Processing host time stamp. Time of the transaction. Must be displayed on the transaction receipt. HH = 2-digit hour, 24 hour clock ("0" left padded | 02 = 2am, 14 = 2pm) II = 2-digit minute ("0" left padded) SS = 2-digit seconds ("0" left padded)
TransType	*String* numeric	Type of transaction that was performed 00 = Purchase 01 = Pre-authorization

6 Response Codes

Approved Response Codes

Response Code	Messages
000	Approved, Account Balances Included (Balance Inquiry), No Reason to Decline Approved (Balances) File Processed/Successful transaction with fault
001	Approved, Account Balances Not Included Approved – No Balances/Approved or completed successfully VIP Approved (No Balances)/Advice Acknowledged – Financial Liability Accepted
002	Approved, country club
003	Approved, maybe more ID
004	Approved, pending ID (sign paper draft)
005	Approved, blind
006	Approved, VIP
007	Approved, administrative transaction
008	Approved, national NEG file hit OK
009	Approved, commercial
010	Approved for partial amount
023	Amex - credit approval
024	Amex 77 - credit approval
027	Transaction already reversed

Response Code	Messages
028	VIP Credit Approved
029	Credit Response Acknowledgement
900	Global Error
901	Invalid URL
902	Malformed XML

Declined Response Codes

Response Code	Messages
050	Do Not Honor Decline Refer to card issuer ID certification fails Deny – Do not Honour Card not initialized Declined: Deny – Unacceptable Fee Unable to locate original transaction Suspected Fraud Deny – Card Acceptor Call Acquirer’s Security Dep Amount Not Reconciled – Totals Provided ATM/POS terminal number cannot be located MAC failed Declined: MAC failed Reserved Security processing failure No arrears (transaction receipt not printed)

Response Code	Messages
	Invalid File Type No such File File Locked Unsuccessful Incorrect File Length File Decompression Error File Name Error File cannot be received Deny – Do Not Honour
051	Expired Card
052	PIN retries exceeded PIN try limit exceeded Allowable number of PIN tries exceeded
053	No sharing
054	No security module
055	Invalid transaction
056	No Support/Transaction Not Permitted to Acquirer Tran Not Supported by FI/Not Supported by Receiver
057	Lost or stolen card
058	Invalid status
059	Deny (Keep Card) – Restricted Card Restricted Card
060	No Chequing account No Savings Account
061	No PBF

Response Code	Messages
062	PBF update error
063	Invalid authorization type
064	Bad Track 2
065	Adjustment not allowed
066	Invalid credit card advance increment
067	Invalid transaction date
068	PTLF error
069	Bad Message Error/No CVM Results Bad message – edit error/Format error
070	No IDF Invalid Issuer Invalid Issuer/Deny – Issuer/Bank Not Found
071	Invalid route authorization Unable to route/Financial institution or intermediate network facility cannot be found for routing Invalid Rout to Auth /Incorrect IIN
072	Card on National NEG file
073	Invalid route service (destination)
074	Unable to authorize Re-enter Transaction Transaction Cannot be Completed Deny – Security Violation Deny – Violation of Law System problem - ask cardholder to insert card in chip card reader Merchant Link not logged on (Network Management Logon required)
075	Invalid PAN length

Response Code	Messages
076	Low funds
077	Pre-auth full
078	Duplicate transaction Duplicate transaction/Request in progress
079	Maximum online refund reached
080	Maximum offline refund reached
081	Maximum credit per refund reached
082	Number of times used exceeded
083	Maximum refund credit reached
084	Duplicate transaction - authorization number has already been corrected by host
085	Inquiry not allowed
086	Over floor limit
087	Maximum number of refund credit by retailer
088	Place call
089	CAF status inactive or closed
090	Referral file full
091	NEG file problem
092	Advance less than minimum
093	Delinquent
094	Over table limit
095	Amount over maximum Amt Over Max/Transaction amount limit exceeded
096	PIN required

Response Code	Messages
097	Mod 10 check failure
098	Force Post
099	Bad PBF

Referral Response Codes

Response Code	Messages
100	Unable to process transaction Invalid Request. Contact Moneris Client POS Certification for repeat declines. Network Unavailable System Malfunction
101	Place call
102	Refer – Call Expired Card Card Acceptor Contact Call Card Accpt Acq Secur
103	NEG file problem
104	CAF problem
105	Card not supported
106	Amount over maximum
107	Over daily limit
108	CAF Problem
109	Advance less than minimum
110	Number of times used exceeded
111	Delinquent

Response Code	Messages
112	Over table limit
113	Timeout
115	PTLF error
121	Administration file problem
122	Unable to validate PIN: security module down

System Error Response Codes

Response Code	Messages
150	Invalid Service Code/Merchant Merchant Not On File Merchant Not on File/Invalid Merchant
200	Invalid account Invalid Card Number Invalid Account/Deny – No Account Type Requested
201	Incorrect PIN Invalid PIN/Incorrect personal identification number PIN Block Error
202	Advance less than minimum
203	Administrative card needed
204	Amount over maximum
205	Invalid Advance Amount Original Amnt Incorrect Bad message/Invalid Amount Original transaction amount error
206	CAF not found

Response Code	Messages
	Invalid “to” account Invalid “from” account Invalid account
207	Invalid transaction date
208	Invalid expiration date
209	Invalid transaction code
210	PIN key sync error
212	Destination not available
251	Error on cash amount
252	Debit not supported

American Express Response Codes (Declines)

Response Code	Messages
426	AMEX - Denial 12
427	AMEX - Invalid merchant
429	AMEX - Account error
430	AMEX - Expired card
431	AMEX - Call Amex
434	AMEX - Call 03 Note: Invalid CVD (CID)
435	AMEX - System down
436	AMEX - Call 05
437	AMEX - Declined
438	AMEX - Declined

Response Code	Messages
439	AMEX - Service error
440	AMEX - Call Amex
441	AMEX - Amount error

Credit Card Response Codes (Declines)

Response Code	Messages
408	CREDIT CARD - Card use limited - Refer to branch
475	CREDIT CARD - Invalid expiration date
476	CREDIT CARD - Invalid transaction, rejected No Credit Account Invalid transaction/Invalid related transactions Unable to process/Suspected malfunction; related transaction error Unable to Authorize: Cut off is in process Issuer not capable to process Switch system malfunction Issuer response not received by CUPS Unable to Authorize/Illegal Status of Acquirer
477	CREDIT CARD - Refer Call/Invalid Card Number Invalid card number (no such account) Deny – Card Not Found Items not on Bankbook beyond limit, declined/Invalid card number
478	CREDIT CARD - Decline, Pick up card, Call
479	CREDIT CARD - Decline, Pick up card
480	CREDIT CARD - Decline, Pick up card
481	CREDIT CARD - Decline

Response Code	Messages
	Transaction not allowed to be processed by cardholder Low funds/Insufficient Balance Invalid Transaction Transaction not allowed to be processed by merchant
482	CREDIT CARD - Expired Card
483	CREDIT CARD – Refer/Refer to Issuer Deny – Card Acceptor Contact Acquirer
484	CREDIT CARD - Expired card - refer
485	CREDIT CARD - Not authorized
486	CREDIT CARD - CVV Cryptographic error
487	CREDIT CARD - Invalid CVV
489	CREDIT CARD - Invalid CVV
490	CREDIT CARD - Invalid CVV
492	System problem - ask cardholder to insert card in chip card reader Withdrawal count exceeded

System Decline Response Codes

Response Code	Messages
800	Bad format
801	Bad data
802	Invalid Clerk ID
809	Bad close
810	System timeout
811	System error
821	Bad response length

Response Code	Messages
877	Invalid PIN block
878	PIN length error
880	Final packet of a multi-packet transaction
881	Intermediate packet of a multi-packet transaction
889	MAC key sync error
898	Bad MAC value
899	Bad sequence number - resend transaction
900	Capture - PIN Tries Exceeded
901	Capture - Expired Card
902	Capture - NEG Capture
903	Capture - CAF Status 3
904	Capture - Advance < Minimum
905	Capture - Num Times Used
906	Capture - Delinquent
907	Capture - Over Limit Table
908	Capture - Amount Over Maximum Capture - Capture Pick up Card Suspected Fraud Hard Capture Deny – Keep Card: Special Conditions Expired Card Fraud Card Acceptor Call Acquirer’s

Response Code	Messages
	Do Not Honour
950	Admin card is not enabled on Merchant profile

Other Response Codes

Response Code	Message
599	Decline

Admin Response Codes

Response Code	Messages
960	Initialization Failure - No Match on Merchant ID
961	Initialization Failure - No Match on PED ID
962	Initialization Failure - No match on Printer ID
963	No match on Poll code
964	Initialization Failure - No match on Concentrator ID
965	Invalid software version number
966	Duplicate terminal name
970	Terminal/Clerk table full
983	Clerk Totals Unavailable: selected Clerk IDs do not exist or have zero totals
989	MAC Error on Transaction 95 (Initialization and Handshake), most often, this indicates that the wrong keys have been injected into a device/KMAC Sync Error

EMV Reversal Request Codes

Response Code	Messages
990	Chip card declines a host approved transaction

Response Code	Messages
991	Chip card removed before ICC communications are completed

Appendix B Security Requirements

All Merchants and Service Providers that store, process, or transmit cardholder data must comply with PCI DSS and the Card Association Compliance Programs. However, validation requirements vary by business and are contingent upon your "Merchant Level" or "Service Provider Level". Failure to comply with PCI DSS and the Card Association Compliance Programs 3.2 may result in a Merchant being subject to fines, fees or assessments and/or termination of processing services. Non-compliant solutions may prevent merchants boarding with Moneris Solutions.

As a Moneris Solutions client or partner using this method of integration, your solution must demonstrate compliance to the Payment Card Industry Data Security Standard (PCI DSS) and/or the Payment Application Data Security Standard (PA DSS) 3.2. These standards are designed to help the cardholders and merchants in such ways as they ensure credit card numbers are encrypted when transmitted/stored in a database and that merchants have strong access control measures, logging, secure software updates, secure remote access and support.

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

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