BE PAYMENT READY

Google Pay™ - Merchant Integration Guide

Version: 1.1.1

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

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/

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. 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 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()
                                                .setRecurAmount("1.00") // Amount to be
sent when managed recur transaction is triggered
                                                .setStartDate("2018/09/01") // First day
the recurring starts
                                                .setStartNow("false") // true to charge
the transaction amount now or false to register the sequence
                                                .setNumRecurs("5") // Number of recurs
                                                .setPeriod("day") // The period for the
recur, can be "day", "week", "month", or "eom"
                                                .build();


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

    // Transaction transport structure
    com.moneris.googlepay.api.MonerisHttpsPost monerisHttpsPost = new com.-
moneris.googlepay.api.MonerisHttpsPost.Builder()
                                                                    .setStoreId(com.-
google.android.gms.samples.wallet.Constants.GATEWAY_TOKENIZATION_PARAMETERS.get(0).second)
// Mandatory, should be securely stored.
                                                                    .setApiToken
("spedguy")  // Mandatory, should be securely stored.
                                                                    .setRequest(purchase)
```

```
// Mandatory from line above
                                                                .setTestMode(true)  //
Optional, default is testMode false;
                                                                .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)
{
    Log.e("ERROR", "Failed to send transaction with error: "+ex.getMessage());
    // send -1 for a catch all error
    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 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/paymentrequest/tutorial
2. Copy the code shown under the subheading "Putting it all together" and paste it into your code editor
3. After the initial `div`, add the following script tag:
   `<script async src="https:/ <script async src="https://esqa.moneris.com/googlepay/googlepay-api.js" onload="MonerisGooglePay.onReady()"></script>`

4. 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 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>
```

5. In function onBuyClicked, under createPaymentRequest(), insert these lines:
```
paymentData["orderId"] = "Unique Order Id”;

paymentData["amount"] = “Same as GoogleTransactionInfo()['TotalPrice']”
```

1.2.2 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 script tag:
```
<script async src="https:/ <script async
src="https://esqa.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. In function **onBuyClicked**, under `createPaymentRequest()`, insert these lines:
```
paymentData["orderId"] = "Unique Order Id”;

paymentData["amount"] = “Same as GoogleTransactionInfo()['TotalPrice']”
```

5. In the same function **onBuyClicked**, underneath the line `.then(function(response) {`, add the code below to call the transaction builder, as shown for a Purchase transaction:

```
    MonerisGooglePay.purchase(paymentData, function(response) {
      if ( response && response.receipt && response.receipt.ResponseCode
&& !isNaN(response.receipt.ResponseCode) )
      {
        if ( parseInt(response.receipt.ResponseCode) < 50 ) {
          alert("Looks like the transaction is approved.");
        }
        else {
          alert("Looks like the transaction is declined.");
        }
      }
      else {
```

```
            throw ("Error processing receipt.");
        }
    });
```

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
- Recurring Billing transactions

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 Recur builder creates the Recur object, which contains the details of a recurring transaction
- 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

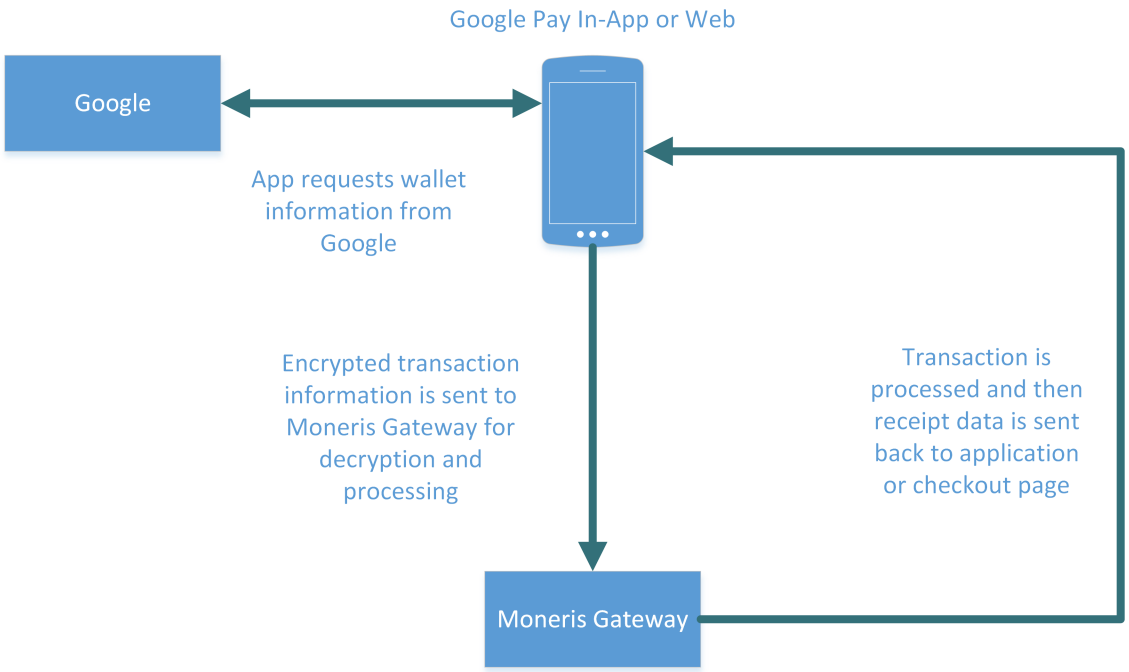
Variable	Set Method	Description
Transaction Type	`.setTransactionType (Constants.TransactionCode.PURCHASE)`	The type of transaction being sent Possible values: PURCHASE PREAUTH
Order ID	`.setOrderId("GooglePayTest-"+new Date().getTime()`	Merchant-defined transaction identifier that must be unique for every Google Pay Purchase and Google Pay Pre-Authorization

Variable	Set Method	Description
		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 Recurring Billing Transactions – Recur Builder

For Recurring Billing transactions, you should send the optional Recur value in the Purchase request.

2.4 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

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).

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 Getting a Production Store ID and API Token (page 1)

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

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

Table 1: Definition of Response Variables - Google Pay™

Variable	Size/Type	Description
ReceiptId	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.
ReferenceNum	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. 660123450010690030 • 66012345: Terminal ID • 001: Shift number • 069: Batch number • 003: Transaction number within the batch.
ReponseCode	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	Size/Type	Description
ISO	2-character numeric	ISO response code
AuthCode	8-character alphanumeric	Authorization code returned from the issuing institution
TransTime	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
TransDate	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)
TransType	alphanumeric	Type of transaction that was performed 00 = Purchase 01 = Pre-authorization
Complete	true/false	Transaction was sent to authorization host and a response was received
Message	100-character alphanumeric	Response description returned from issuing institution
TransAmount	nnnnnnN.NN	Returns the amount sent in request for processing. The

Variable	Size/Type	Description
	9-character decimal (variable length)	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
TransID	20-character alphanumeric	Gateway Transaction identifier
CardType	2-character alphanumeric	Credit Card Type M = MasterCard V = Visa AX = American Express P = INTERAC®
TimedOut	true/false	Transaction failed due to a process timing out
Bank Totals	Object	Response data returned in a Batch Close and Open Totals request.
Ticket	n/a	Reserved field
CorporateCard	true/false	Indicates whether the card is a corporate card or not
CAVV	40-character alphanumeric	Decrypted CAVV value for the transaction Returned for Google Pay™ Purchase/Pre-Authorization transaction if payload is successfully decrypted
IsVisaDebit	true/false/null	Indicates whether the card that the transaction was performed on is Visa debit

Variable	Size/Type	Description
		true = Card is Visa Debit false = Card is not Visa Debit null = there was an error in identifying the card
RecurSuccess	true/false	Indicates whether the transaction successfully registered

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.