FirstAPI Pay With Google Integration Guide

November 2017

Contents

Overview

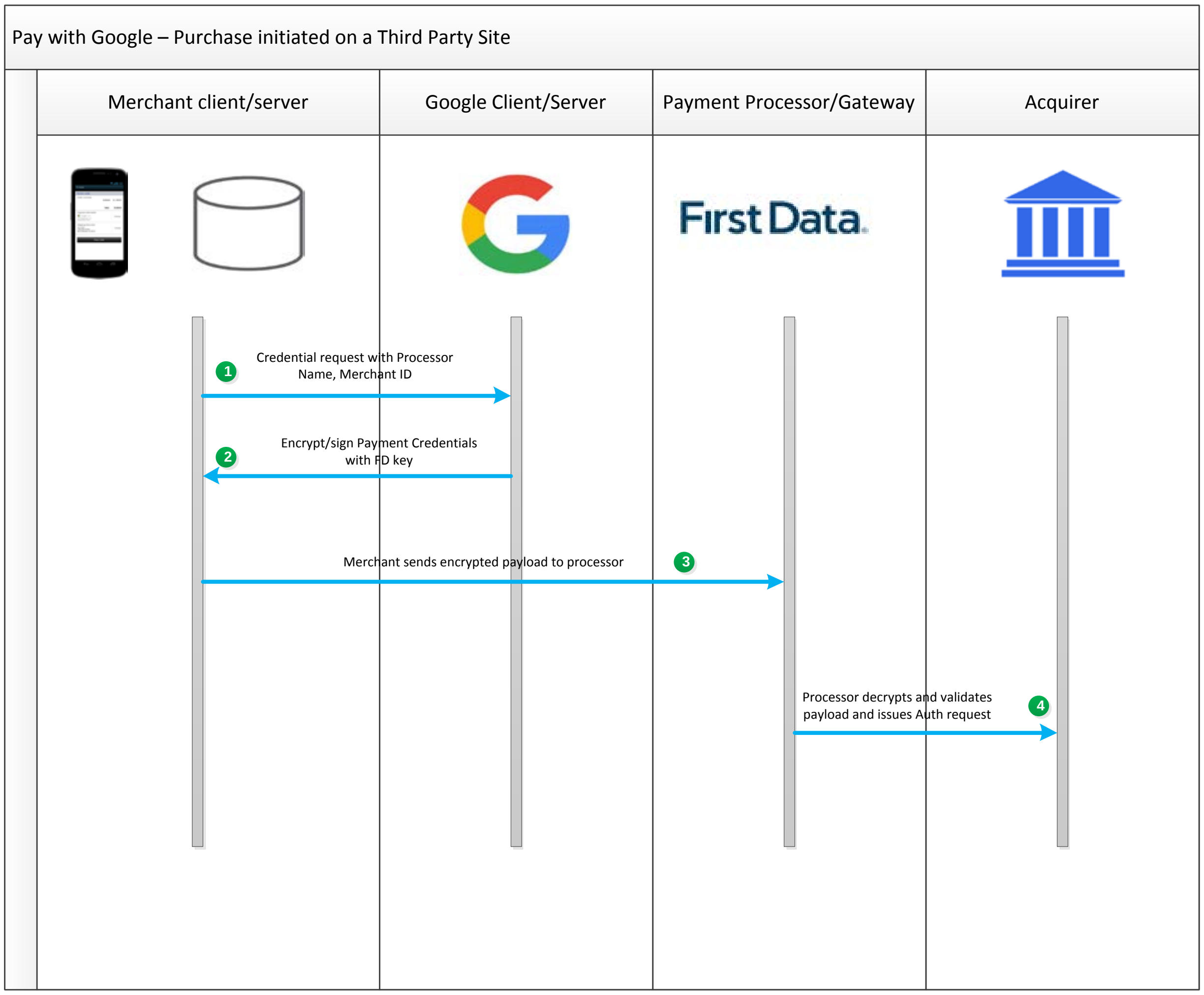
FirstAPI allows developers to quickly enable secure and convenient payments in their payment applications. FirstAPI handles all of the tokenization needed to protect customers' transactions.

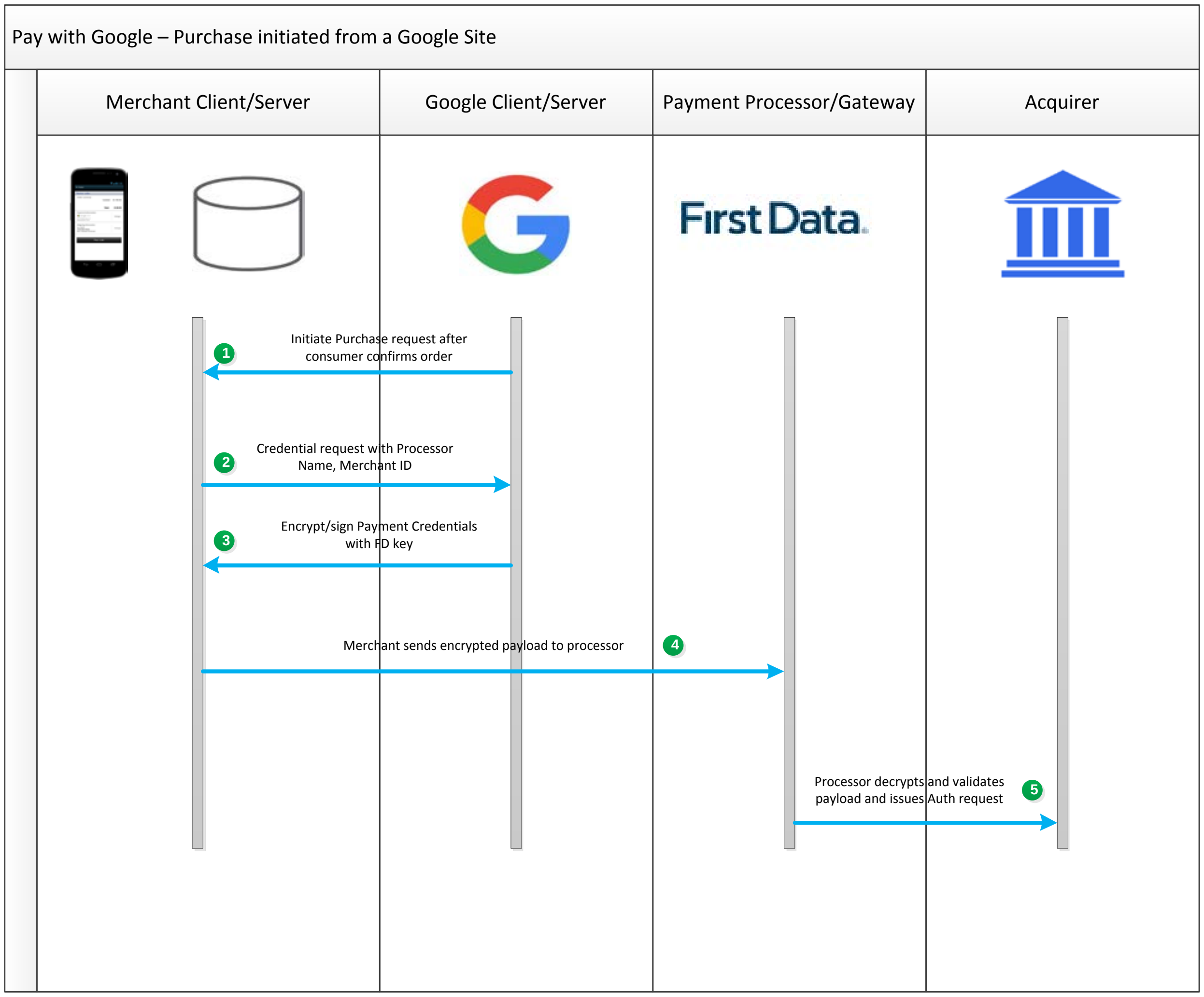
This documentation refers to the Pay with Google integration method included within the overall FirstAPI eCommerce Solution.

The new Google Payment API enables developers to add payment processing to merchants' Android-compatible apps and on Chrome on Android. These APIs allow consumers to pay with any crdit card they have stored in their Google account, including any cards they may have previously set up on their Android Pay digital wallet. Consumers may also add a new payment account without leaving the app.

Tokenized payment credentials from the Android Pay app will continue to be available for purchases.

The following diagrams show the payment flow with a purchase initiated from a third party site, and from a Google site.


Pay with Google – Purchase initiated on a Third Party Site
Merchant client/server
Google Client/Server
Payment Processor/Gateway
Acquirer
First Data
1 Credential request with Processor Name, Merchant ID
2 Encrypt/sign Payment Credentials with FD key
3 Merchant sends encrypted payload to processor
4 Processor decrypts and validates payload and issues Auth request


Pay with Google – Purchase initiated from a Google Site
Merchant Client/Server
Google Client/Server
Payment Processor/Gateway
Acquirer
First Data
1 Initiate Purchase request after consumer confirms order
2 Credential request with Processor Name, Merchant ID
3 Encrypt/sign Payment Credentials with FD key
4 Merchant sends encrypted payload to processor
5 Processor decrypts and validates payload and issues Auth request

Target Audience

The target audience for this document is a developer who wants to use Pay With Google in their payment application.

Related Documents and Resources

The following First Data documents and resources provide supporting information for this document:

- FirstAPI Developer Portal (https://developer.payeezy.com/) - This portal is where developers register and are boarded to FirstAPI. It also contains specifications for all supported APIs, sample requests and responses, and development guides. The following items are especially pertinent to readers of this guide:
  - Pay With Google API
  - Getting Started
  - FAQs

Third Party Resources

In addition to the FirstAPI resources, the following third party documents and resources provide supporting information for working with the FirstAPI SDK and sample Pay with Google application.

Pay with Google Resources

Developers may wish to refer to the Google Payment API documentation, available as follows:

- Google Payment API (http://developers.google.com/payments/)
- Brand Guidelines (https://developers.google.com/payments/brand-guidelines)

Download and Build Resources

- GitHub (https://github.com/) – Recommended for FirstAPI source code downloads
- Android Studio (http://developer.android.com/tools/studio/index.html)– This has the Android SDK tools and plugins and the Android Studio for development. It also contains the most recent version of OpenJDK, which is recommended.
- Gradle (http://gradle.org/) – For builds, thestandard Gradle build can be used.

Minimum Technical Requirements

Developers wishing to use the FirstAPI Pay with Google SDK will need the following software and hardware:

- Google Play Services version 11.4.x or newer installed. (For Mobile web, you must use Chrome M61 on Android for devices running Google Play Services version 11.4.x or later.)

- A physical device or an emulator to use for developing and testing. Google Play services can only be installed on an emulator with an AVD that runs Google APIs platform based on Android 4.4 or higher.

- The latest version of Android Studio (see Third Party Resources, above.) This includes:
  - The latest version of the Android SDK, including the SDK Tools component. The SDK is available from the Android SDK Manager
  - Java JRE (JDK for development) as per Android SDK requirements.

Your project should be able to compile against Android 4.4 (KITKAT) or higher.

Integration

Before integrating with FirstAPI, developers should first:

- Review the prerequisites described in FirstAPI's Getting Started Guide.
- Complete registration and certification to FirstAPI, if necessary.
- Build your app to make purchases directly in the app.

To integrate code with the FirstAPI SDK:

1. Acquire the FirstAPI Pay with Google SDK from GitHub.
2. Define the following parameters:
    a. Merchant ID
    b. Merchant Token
    c. APIKey
    d. APISecret (.m file)
3. Collect credit card information.
4. Use the APIKey, APISecret, and Merchant token from the developer's account on the Developer Portal to execute Authorize and Purchase.
5. Send token to complete Authorize and Purchase.

The following sections describe these steps in more detail.

Acquire the FirstAPI SDK

You can clone the SDK using HTTPS or Subversion. The GitHub clone command is as follows:

```
git clone https://github.com/payeezy/pay_with_google/sdk
```

Or you can download the .zip file at the following location:

```
https://github.com/payeezy/pay_with_google/sdk
```

Once you have the SDK, import the downloaded GooglePay project into Android Studio. FirstAPI supports integration with Android, Java, PHP, Python, Ruby, NodeJS, and Curl.

Define the FirstAPI Object Parameters

Parameters must be updated in the following files:

- Constants.java
- EnvData.java

Update the **`Constants.java`** file with the Merchant ID and Gateway Tokenization parameters. Note that:

- The Merchant ID is found in the developer's FirstAPI Developer Portal account; and
- The Gateway Tokenization parameter defaults to `'firstdata'`. This does not need to be changed unless the merchant works with a different gateway.

In the **`EnvData.java`** file, set the following environment variables, which can all be found in the developer's FirstAPI Developer Portal account:

- APIKey – Shown in the example below set to `fP0iYUx4oJ8LolKl2LiOT1Zo94mL0IDQ`
- Token – Shown in the example below set to `fdoa-a480ce8951daa73262734cf102641994c1e55e7cdf4c02b6`
- APISecret – Shown in the example below set to `rLBOY4B5BYE4mVDg`

```
envMap.put("INT", new EnvPropertiesImpl(
        "INT",
        "https://api-int.payeezy.com/v1/transactions",
        "fP0iYUx4oJ8LolKl2LiOT1Zo94mL0IDQ",
        "fdoa-a480ce8951daa73262734cf102641994c1e55e7cdf4c02b6",Token
        "rLBOY4B5BYE4mVDg" API Secret
      )
 );
```

Collect Credit Card Information

Note that even for testing purposes, the credit card information used in the app must be attached to an active account. The standard test cards archived in the Sandbox region of the Developer Portal will not be validated by Google and will fail in processing.

Execute Authorize and Purchase Request

The following sample shows the request code.

Note: See FirstAPI Header Authorization Parameter, below, for details on how to create the Authorization parameter that is required as part of the Header.

```
public void sendRequestToFirstData(final PaymentData paymentData) {

      try {
          //  Parse the Json token retrieved from the Full Wallet.
          String tokenJSON = paymentData.getPaymentMethodToken().getToken();
          final JSONObject jsonObject = new JSONObject(tokenJSON);

          String signedMessage=jsonObject.getString("signedMessage");//contains
encryptedMessage, publickey and tag
          String protocolVersion=jsonObject.getString("protocolVersion");
          String signature = jsonObject.getString("signature");


          //  Create a First Data Json request
          JSONObject requestPayload = getRequestPayload(signedMessage, protocolVersion,
signature);
          final String payloadString = requestPayload.toString();
          final Map<String, String> HMACMap = computeHMAC(payloadString);


          StringRequest request = new StringRequest(
                  Request.Method.POST,
                  getUrl(mEnv),
                  new Response.Listener<String>() {
                      @Override
                      public void onResponse(String response) {
                          //  request completed - launch the response activity

                          startResponseActivity("SUCCESS", response);
                      }
                  },
                  new Response.ErrorListener() {
                      @Override
                      public void onErrorResponse(VolleyError error) {

                          startResponseActivity("ERROR", formatErrorResponse(error));
                      }
                  }) {

              @Override
              public String getBodyContentType() {
                  return "application/json";
              }

              @Override
              public byte[] getBody() {
                  try {
                      return payloadString.getBytes("UTF-8");
                  } catch (UnsupportedEncodingException e) {
                      return null;
                  }
              }

              @Override
              public Map<String, String> getHeaders() throws AuthFailureError {
                  Map<String, String> headerMap = new HashMap<>(HMACMap);
                  //Map<String, String> headerMap = new HashMap<>();
                  //  First data issued APIKey identifies the developer
                  headerMap.put("apikey", EnvData.getProperties(mEnv).getApiKey());
                  //headerMap.put("developer_id",
EnvData.getProperties("QA").getDeveloperId());
                  //  First data issued token identifies the merchant
                  headerMap.put("token", EnvData.getProperties(mEnv).getToken());

                  return headerMap;
              }
          };

          request.setRetryPolicy(new DefaultRetryPolicy(0, -1,
DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
          //Toast.makeText(CheckoutActivity.this, "before forming queue request to volley!!",
Toast.LENGTH_LONG).show();
          RequestQueue queue = Volley.newRequestQueue(CheckoutActivity.this);

          queue.add(request);
          //Toast.makeText(CheckoutActivity.this, "after forming queue request to volley!!",
Toast.LENGTH_LONG).show();
      } catch (JSONException e) {
          Toast.makeText(CheckoutActivity.this, "Error parsing JSON payload",
Toast.LENGTH_LONG).show();
      }
```

FirstAPI Header Authorization Parameter

The Authorization parameter, required as part of the Header for a FirstAPI transaction, is created as follows:

1. Construct the data param by appending the following parameters in the order shown:
   a. apikey – the developer's API key from their FirstAPI Developer Portal account
   b. nonce – A secure random number
   c. timestamp – Epoch timestamp in milliseconds
   d. token – the Merchant Token from the developer's FirstAPI Developer Portal account
   e. payload – The actual body content passed as the POST request
2. Compute the HMAC SHA256 hash on the param using the `apiSecret` token (associated with the apikey parameter) from the developer's FirstAPI Developer Portal account.
3. Calculate the base64 of the hash, which will be the required Authorization header parameter.

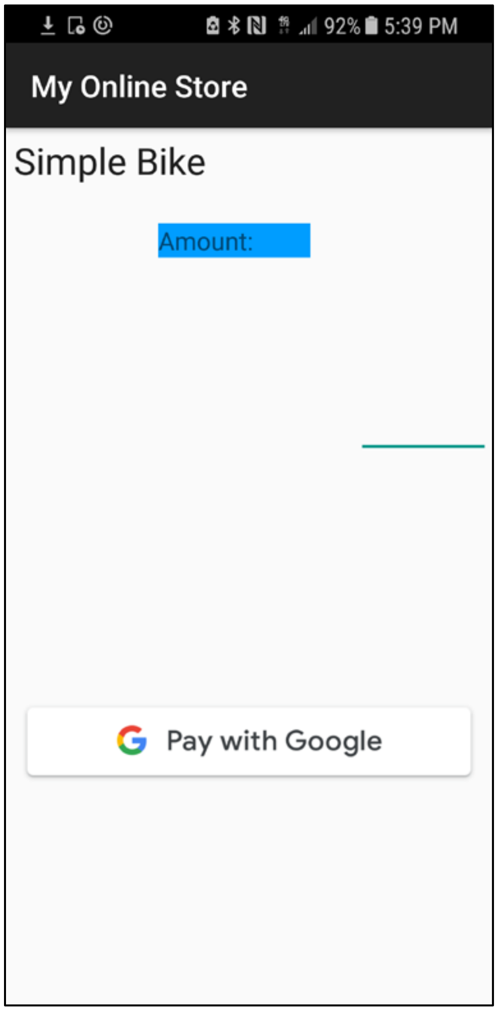
See the example below.

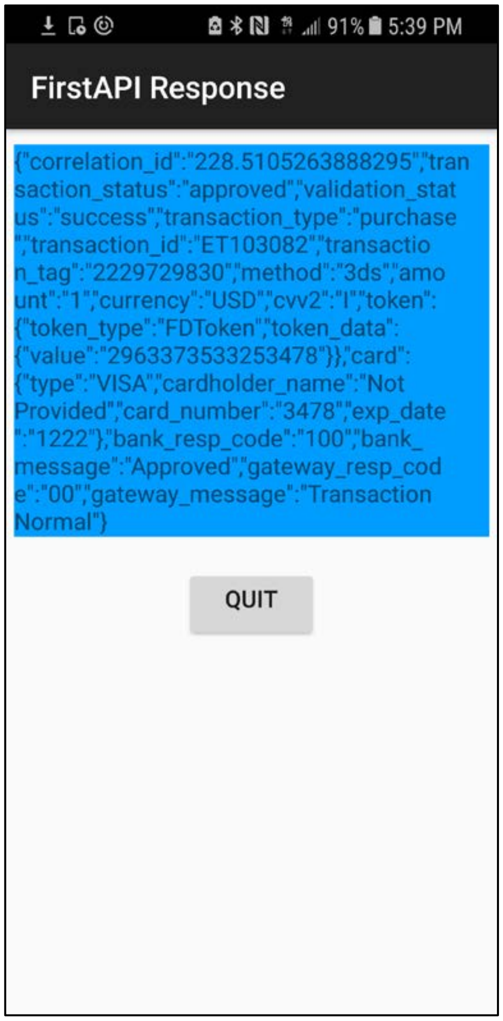
```
public String getMacValue(Map<String,String> data) throws Exception{
Mac mac=Mac.getInstance("HmacSHA256");
String apiSecret= data.get(APISECRET);
MessageLogger.logMessage(String.format("API_SECRET:{}",apiSecret));
SecretKeySpec secret_key = new SecretKeySpec(apiSecret.getBytes(), "HmacSHA256");
mac.init(secret_key);
StringBuilder buff=new StringBuilder();
buff.append(data.get(APIKEY))
.append(data.get(NONCE))
.append(data.get(TIMESTAMP));
if(data.get(TOKEN)!=null)
buff.append(data.get(TOKEN));
if(data.get(PAYLOAD)!=null)
buff.append(data.get(PAYLOAD));
String bufferData = buff.toString();
MessageLogger.logMessage(String.format(bufferData));
byte[] macHash=mac.doFinal(bufferData.getBytes("UTF-8"));
MessageLogger.logMessage(Integer.toString(macHash.length));
MessageLogger.logMessage(String.format("MacHAsh:{}",Arrays.toString( macHash)));
String authorizeString=android.util.Base64.encodeToString(toHex(macHash),
android.util.Base64.NO_WRAP);
MessageLogger.logMessage(String.format("Authorize: {}",authorizeString));
return authorizeString;
}
```

Pay With Google APK Installation To Device

To install the GooglePay Android Application Package (APK) file directly on an Android device:

1. Once the downloaded code is built successfully in Android Studio, build the APK and install it on your device.
2. Once the APK is installed, select the Open option to access the application. The screenshots below show the sample application installed on the test device, and the response from a payment processed in the FirstAPI CERT environment. Note that the Payment Details page cannot be captured for security reasons.





```
{"correlation_id":"228.5105263888295","tran
saction_status":"approved","validation_stat
us":"success","transaction_type":"purchase
","transaction_id":"ET103082","transactio
n_tag":"2229729830","method":"3ds","amo
unt":"1","currency":"USD","cvv2":"I","token":
{"token_type":"FDToken","token_data":
{"value":"2963373533253478"}},"card":
{"type":"VISA","cardholder_name":"Not
Provided","card_number":"3478","exp_date
":"1222"},"bank_resp_code":"100","bank_
message":"Approved","gateway_resp_cod
e":"00","gateway_message":"Transaction
Normal"}
```