First Data Pay With Google Integration Guide

Version 1.0a / December 2017

First Data.

Contents

Revision History

Version	Date	Description
1.0	19 Dec 2017	Initial version of document
1.0a	20 Dec 2017	Added links to Google repository and Quick Start page

Overview

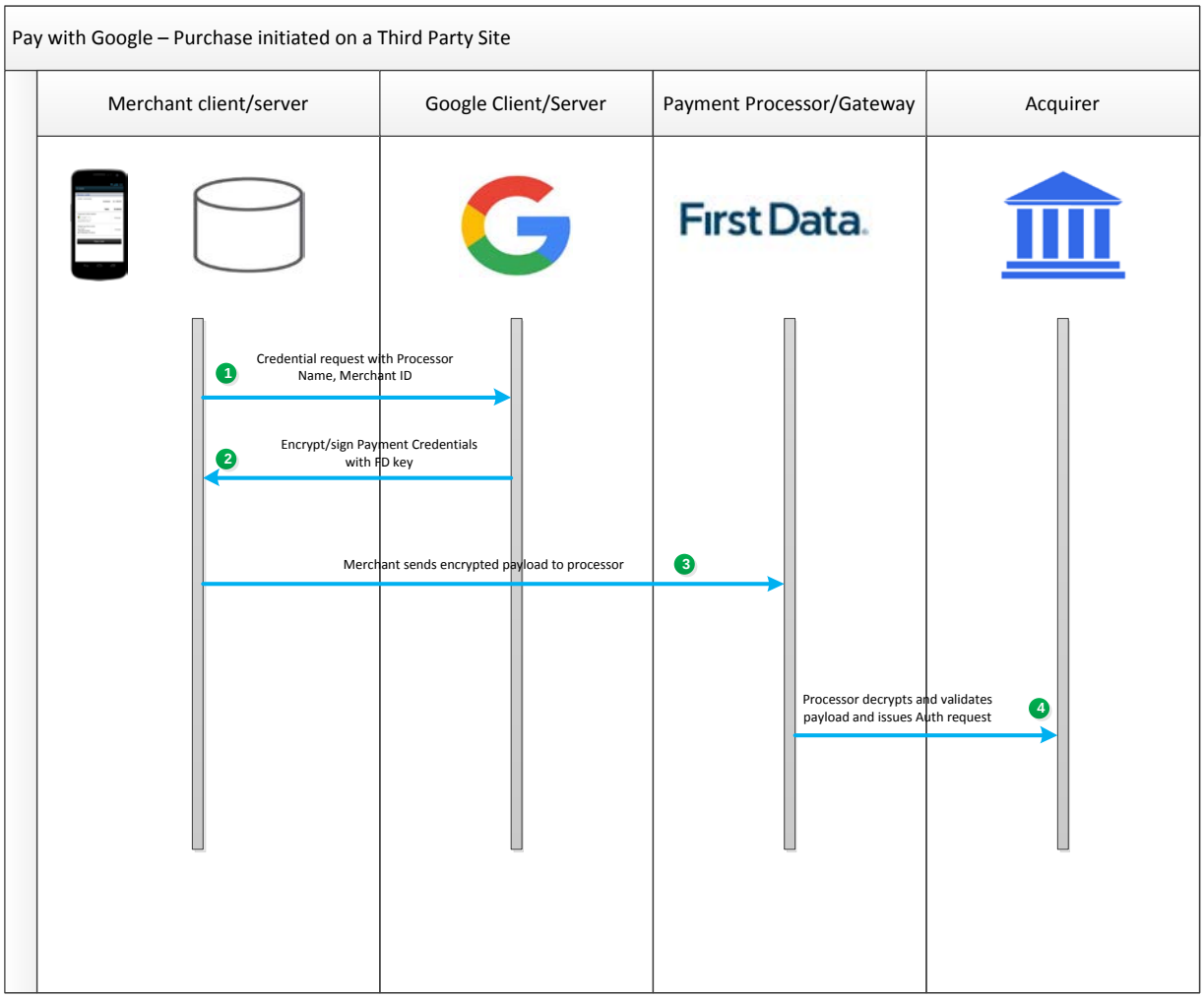
First Data's RESTful API allows developers to quickly enable secure and convenient payments in their payment applications. The API handles all of the tokenization needed to protect customers' transactions.

This documentation refers to the Pay with Google integration method.

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 diagram show the payment flow with a purchase initiated from a third party site.



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:

- Developer Portal (https://developer.payeezy.com/) - This portal is where developers register and are boarded to First Data's RESTful API. 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
- First Data Sample Pay with Google Application – This sample application shows how to integrate the First Data APIs to support the Pay with Google feature. This sample application runs in the First Data CERT environment and the Google test environment.

Third Party Resources

In addition to the First Data resources, the following third party documents and resources provide supporting information for working with the First Data APIs 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 Quick Start – Tutorial for working with the Google Pay with Google sample application
- Google Payments API Quick Start Repository – This github repository holds the Google sample application and other helpful resources
- 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 First Data sample application 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, the standard Gradle build can be used.

Minimum Technical Requirements

Developers wishing to use the First Data Pay with Google sample application 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 First Data's APIs, developers should first:

- Review the prerequisites described in the First Data Getting Started Guide.
- Complete registration and certification to the APIs at the Developer Portal, if necessary.
- Build your app to make purchases directly in the app.

To integrate code with the First Data sample application:

1. Acquire the First Data Pay with Google sample application 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 First Data Pay with Google Sample Application

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

```
git clone https://github.com/payeezy/pay_with_google/tree/master/sample_app
```

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

```
https://github.com/payeezy/pay_with_google/tree/master/sample_app
```

Once you have the sample application, import the downloaded GooglePay project into Android Studio. First Data supports integration with Android, Java, PHP, Python, Ruby, NodeJS, and Curl.

Define the First Data 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 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 Developer Portal account:

- APIKey – Shown in the example below set to `y6pWAJNyJyjGv66IsVuWnklkKUPFbb0a`
- Token – Shown in the example below set to `fdoa-a480ce8951daa73262734cf102641994c1e55e7cdf4c02b6`
- APISecret – Shown in the example below set to `86fbae7030253af3cd15faef2a1f4b67353e41fb6799f576b5093ae52901e6f7`

```
envMap.put("CERT", new EnvPropertiesImpl(
             "CERT",
             "https://api-cert.payeezy.com/v1/transactions",

             "y6pWAJNyJyjGv66IsVuWnklkKUPFbb0a",
             "fdoa-a480ce8951daa73262734cf102641994c1e55e7cdf4c02b6",
             "86fbae7030253af3cd15faef2a1f4b67353e41fb6799f576b5093ae52901e6f7"));
```

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 sections shows sample code snippets.

First Data Header Authorization Parameter

The Authorization parameter, required as part of the Header for a First Data API 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 Developer Portal account
    b. nonce – A secure random number
    c. timestamp – Epoch timestamp in milliseconds
    d. token – the Merchant Token from the developer's 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 Developer Portal account, as shown below.

```
/**
 * Compute HMAC signature for the payload. The signature is based on the APIKey and the
 * APISecret provided by First Data. If the APISecret is not specified, the HMAC is
 * not computed.
 *
 * @param payload The payload as a String
 * @return Map of HTTP headers to be added to the request
 */
private Map<String, String> computeHMAC(String payload) {

    EnvProperties ep = EnvData.getProperties(mEnv);
    String apiSecret = ep.getApiSecret();
    String apiKey = ep.getApiKey();
    String token = ep.getToken();

    Map<String, String> headerMap = new HashMap<>();
    if (apiSecret != null) {
        try {
            String authorizeString;
            String nonce =
Long.toString(Math.abs(SecureRandom.getInstance("SHA1PRNG").nextLong()));
            String timestamp = Long.toString(System.currentTimeMillis());

            Mac mac = Mac.getInstance("HmacSHA256");
            SecretKeySpec secretKey = new SecretKeySpec(apiSecret.getBytes(),
"HmacSHA256");
            mac.init(secretKey);

            StringBuilder buffer = new StringBuilder()
                    .append(apiKey)
                    .append(nonce)
                    .append(timestamp)
                    .append(token)
                    .append(payload);

            byte[] macHash = mac.doFinal(buffer.toString().getBytes("UTF-8"));
            authorizeString = Base64.encodeToString(bytesToHex(macHash).getBytes(),
Base64.NO_WRAP);

            headerMap.put("nonce", nonce);
            headerMap.put("timestamp", timestamp);
            headerMap.put("Authorization", authorizeString);
        } catch (Exception e) {
            //  Nothing to do
        }
    }
    return headerMap;
}
```

The following code shows a sample request sent to First Data using the Volley Restful library.

```
/*******
 *Added for FD processing
 */
/**
 * Send a request to the First Data server to process the payment. The REST request
 * includes HTTP headers that identify the developer and the merchant issuing the request:
 * <ul>
 * <li>{@code apikey} - identifies the developer</li>
 * <li>{@code token} - identifies the merchant</li>
 * </ul>
 * The values for the two headers are provided by First Data.
 * <p>
 * The token created by Android Pay is extracted from the FullWallet object. The token
 * is in JSON format and consists of the following fields:
 * <ul>
 * <li>{@code encryptedMessage} - the encrypted details of the transaction</li>
 * <li>{@code ephemeralPublicKey} - the key used, together with the key pair issued
 * by First Data, to decrypt of the transaction detail</li>
 * <li>{@code tag} - a signature field</li>
 * </ul>
 * These items, along with a {@code PublicKeyHash} that is used to identify the
 * key pair provided by First data, are used
 * to create the transaction payload. The payload is sent to the First Data servers
 * for execution.
 * </p>
 *
 * @param paymentData PaymentData object
 * //@param env        First Data environment to be used
 */
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);

                //  First data issued APIKey identifies the developer
                headerMap.put("apikey", EnvData.getProperties(mEnv).getApiKey());

                //  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));
        RequestQueue queue = Volley.newRequestQueue(CheckoutActivity.this);

        queue.add(request);

    } catch (JSONException e) {
        Toast.makeText(CheckoutActivity.this, "Error parsing JSON payload",
Toast.LENGTH_LONG).show();
    }
}
```

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 for the sample app (available on the First Data github, here) 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 First Data CERT environment/Google test environment.

   Note that the Payment Details page cannot be captured for security reasons.