Visa Checkout

SDK for Android

Effective: September 27, 2017

Version 5.5

Contents

What’s New in This Release

Global Market Expansion Enhancements

The following enhancements have been added:

- Autofill is now available for the `Municipio`, `Ciudad`, and `Estado` fields in Mexico by using pre-populated data from Servicio Postal Mexicano (Sepomex), the official provider for address-related information in Mexico. For any consumers who create or edit address-related information for the country of Mexico in Visa Checkout, the address-related fields are automatically pre-populated based on the ZIP code (`Codigo postal`) and `Colonia` values. Consumers can also edit any pre-populated fields.
- Ukrainian `Address Line 2` is now optional

  For any consumers who create or edit address-related information in Visa Checkout, `Address Line 2` is now optional for Ukrainian addresses.
- Additional bins have been provided for ELO cards in Brazil:
  - Crédito (Credit only)
  - Múltiplo (Combo) (Credit only)
  - Cultura (Pre-Paid)
  - Pré-Pago (Pre-Paid)

Additional Features

The following features have been added:

- New screen designs

  Any previously redesigned screens have been updated and now include additional screens in the new user flow, as well as a One-Time Password (OTP) screen and a forgot password screen.
- Support for Google Floodlight tags

  Support for Google Floodlight tags has been enabled to track users who click any paid search ads.

Visa Checkout Quick Start 1

Visa Checkout is a digital payment service in which consumers can store card information for Visa, MasterCard, Discover, American Express, and ELO cards. Visa Checkout SDK for Android provides merchants with a quick integration to accept payments from these consumers within their native Android applications.

Important:

Download Visa Checkout SDK for Android from `developer.visa.com`.

To proceed, your environment requires:

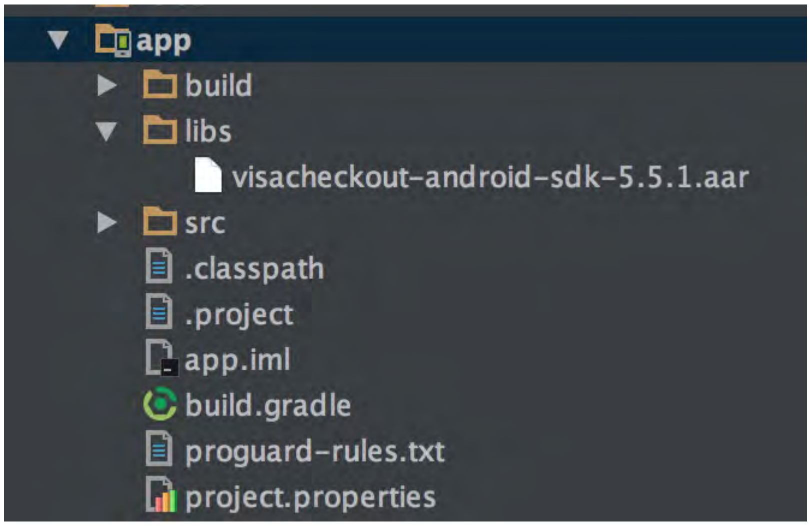
- Android 4.1.x Jelly Bean (API level 16) or later
- Android Studio 2.3 or later
- Gradle Plugin 2.2.3 or later

Basic Configuration

The following steps configure and set up the Visa Checkout mobile SDK for Android in your Android Studio environment and enable you to interact with a Visa Checkout button inside a simulated app:

Note: *You can import the `VisaCheckoutSampleApp` sample app, which is included in the SDK, to quickly look at the parameters and code associated with Visa Checkout. In Android Studio:*

1. Add `visacheckout-android-sdk-5.5.1.aar` to your `libs` folder:



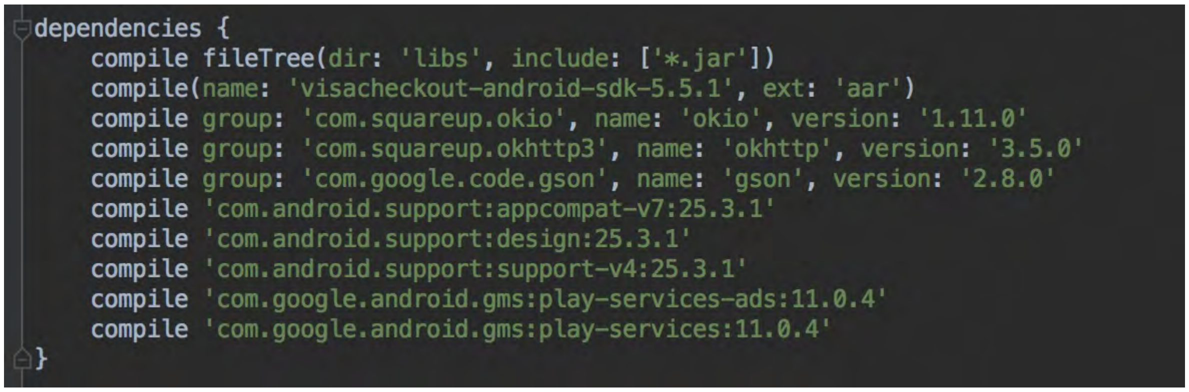
2. In your app module's `build.gradle`:
   - Set the repositories location to include the location of the SDK:

     ```
     repositories {
       flatDir { dirs 'libs' }
     }
     ```

   - Add the Visa Checkout SDK as a dependency:

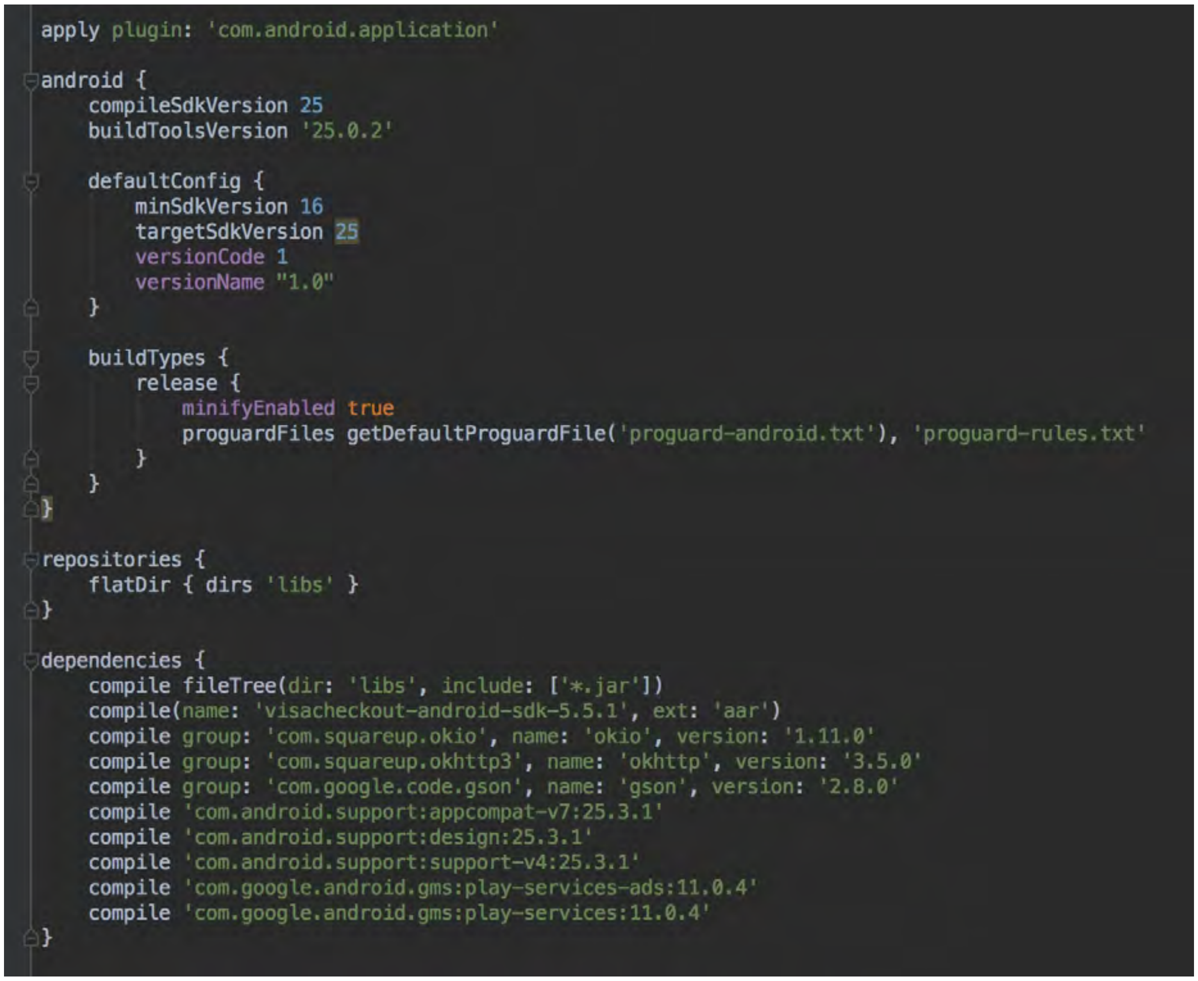
     `compile(name:'visacheckout-android-sdk-5.5.1', ext:'aar')`

   - Add other dependencies required by Visa Checkout:



```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile(name: 'visacheckout-android-sdk-5.5.1', ext: 'aar')
    compile group: 'com.squareup.okio', name: 'okio', version: '1.11.0'
    compile group: 'com.squareup.okhttp3', name: 'okhttp', version: '3.5.0'
    compile group: 'com.google.code.gson', name: 'gson', version: '2.8.0'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:design:25.3.1'
    compile 'com.android.support:support-v4:25.3.1'
    compile 'com.google.android.gms:play-services-ads:11.0.4'
    compile 'com.google.android.gms:play-services:11.0.4'
}
```

Your app module's `build.gradle` file looks similar to the following:

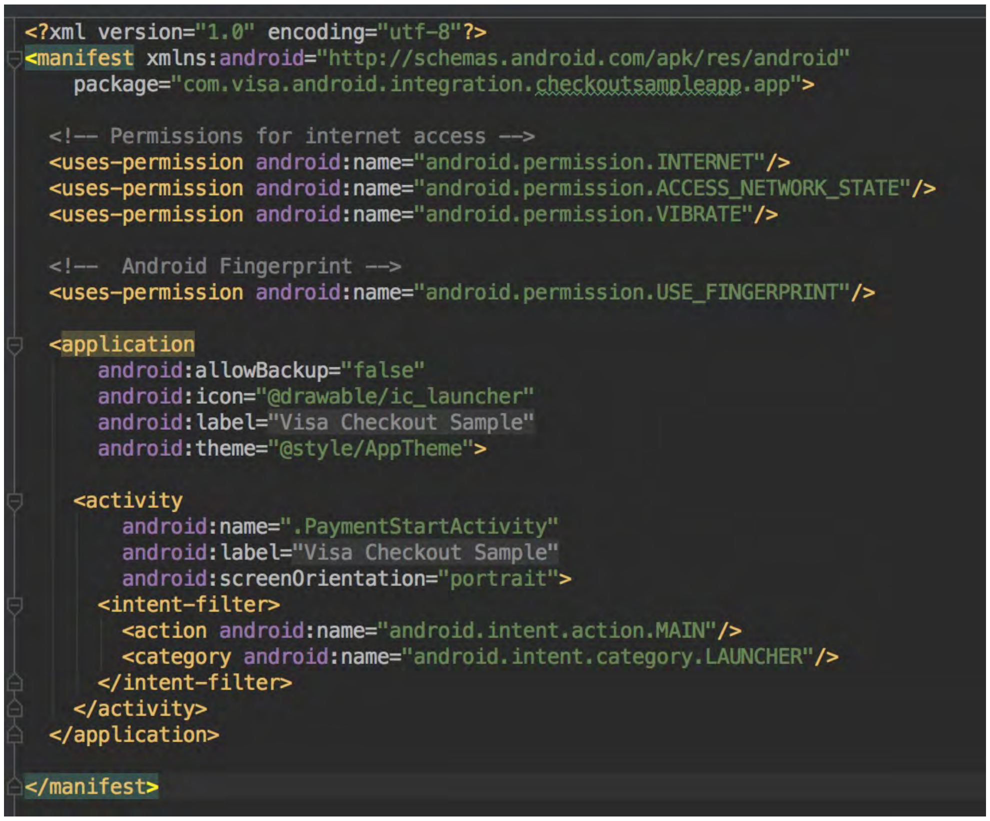


```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion '25.0.2'

    defaultConfig {
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.txt'
        }
    }
}

repositories {
    flatDir { dirs 'libs' }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile(name: 'visacheckout-android-sdk-5.5.1', ext: 'aar')
    compile group: 'com.squareup.okio', name: 'okio', version: '1.11.0'
    compile group: 'com.squareup.okhttp3', name: 'okhttp', version: '3.5.0'
    compile group: 'com.google.code.gson', name: 'gson', version: '2.8.0'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:design:25.3.1'
    compile 'com.android.support:support-v4:25.3.1'
    compile 'com.google.android.gms:play-services-ads:11.0.4'
    compile 'com.google.android.gms:play-services:11.0.4'
}
```

Note: *Ensure that the `AndroidManifest.xml` file specifies a minimum SDK version of `16`:*

```
minSdkVersion 16
```

3. Add permissions to the `AndroidManifest.xml` file. At a minimum, you must add `INTERNET`, `ACCESS_NETWORK_STATE`, `VIBRATE`, and `USE_FINGERPRINT` permissions:

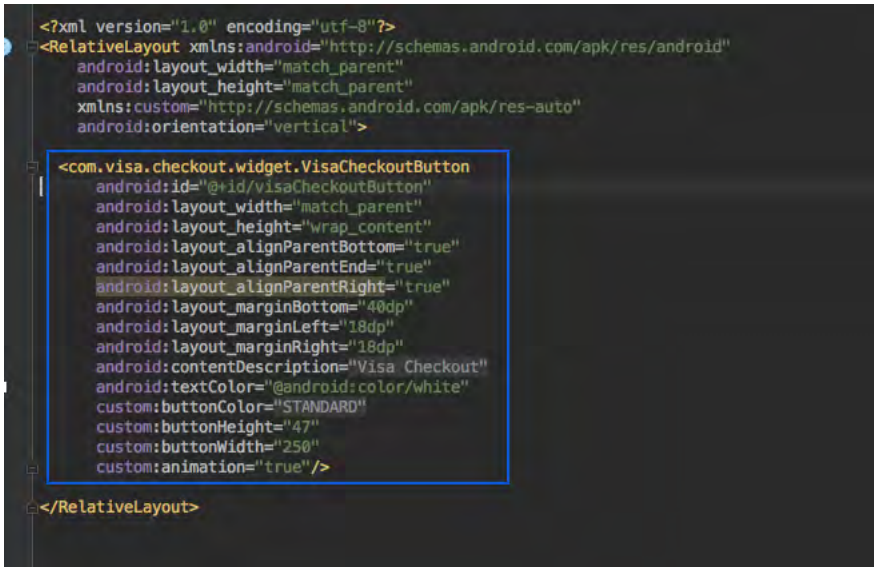


```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.visa.android.integration.checkoutsampleapp.app">

  <!-- Permissions for internet access -->
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
  <uses-permission android:name="android.permission.VIBRATE"/>

  <!--  Android Fingerprint -->
  <uses-permission android:name="android.permission.USE_FINGERPRINT"/>

  <application
      android:allowBackup="false"
      android:icon="@drawable/ic_launcher"
      android:label="Visa Checkout Sample"
      android:theme="@style/AppTheme">

    <activity
        android:name=".PaymentStartActivity"
        android:label="Visa Checkout Sample"
        android:screenOrientation="portrait">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>

</manifest>
```

Note: *If your enabling backups is specified in the* `AndroidManifest.xml` *file, you must disable it:*

`android:allowBackup="false"`

4. In your layout, add a Visa Checkout button, which in this example is a Visa Checkout button:



```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical">

  <com.visa.checkout.widget.VisaCheckoutButton
      android:id="@+id/visaCheckoutButton"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
      android:layout_alignParentEnd="true"
      android:layout_alignParentRight="true"
      android:layout_marginBottom="40dp"
      android:layout_marginLeft="18dp"
      android:layout_marginRight="18dp"
      android:contentDescription="Visa Checkout"
      android:textColor="@android:color/white"
      custom:buttonColor="STANDARD"
      custom:buttonHeight="47"
      custom:buttonWidth="250"
      custom:animation="true"/>

</RelativeLayout>
```

5. In your `MainActivity` class:
   - Call the SDK's `init()` method to initialize the SDK

```
VisaCheckoutSdk.init(
  getApplicationContext(),
  Profile.Environment.SANDBOX,
    "MERCHANT-API-KEY",
    "PROFILE-NAME",
    new VisaCheckoutSdkInitListener(){
      @Override public void status(int code, String message){
       }});
```

There are two signatures for the `init()` method. At a minimum you must specify the following parameters:

- The activity's context, which is obtained by calling the activity's `getApplicationContext()` method
- The environment, which in this example, is the Visa Checkout sandbox
- Your API key
- The name of a profile, which you set up on after creating your account at `developer.visa.com`.
- An object, `VisaCheckoutSdkInitListener`, which receives the status after initialization.

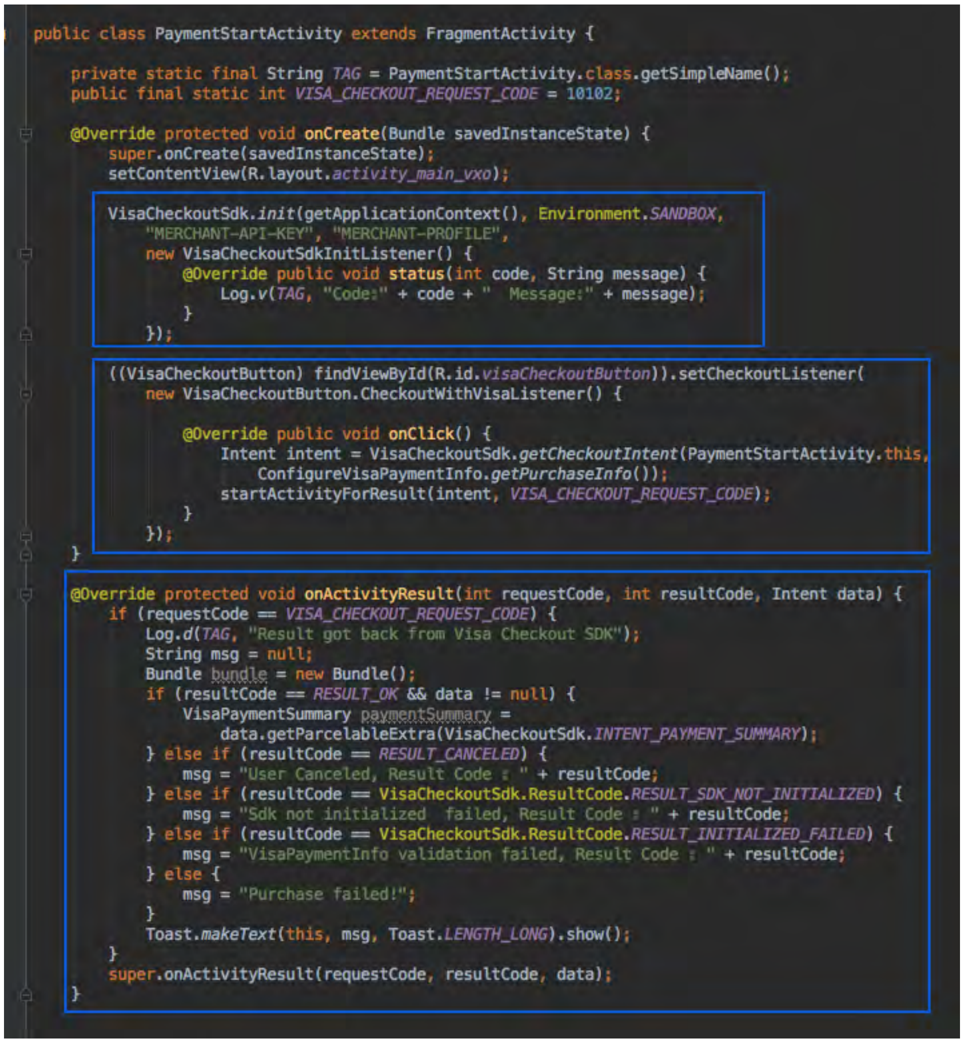
- Set up an `onClickListener` event for the Visa Checkout button that you added via the layout in Step 4.
- Call the SDK's `getCheckoutIntent()` method to obtain an Android `Intent` object with which to launch the Visa Checkout SDK when user taps the button.

  In your call to `getCheckoutIntent()`, as a minimum:

  - Pass the instance of the activity from which Visa Checkout launches
  - Pass the subtotal, total, and currency associated with the transaction
- After obtaining the intent, call the `startActivityForResult()` method to initiate processing.

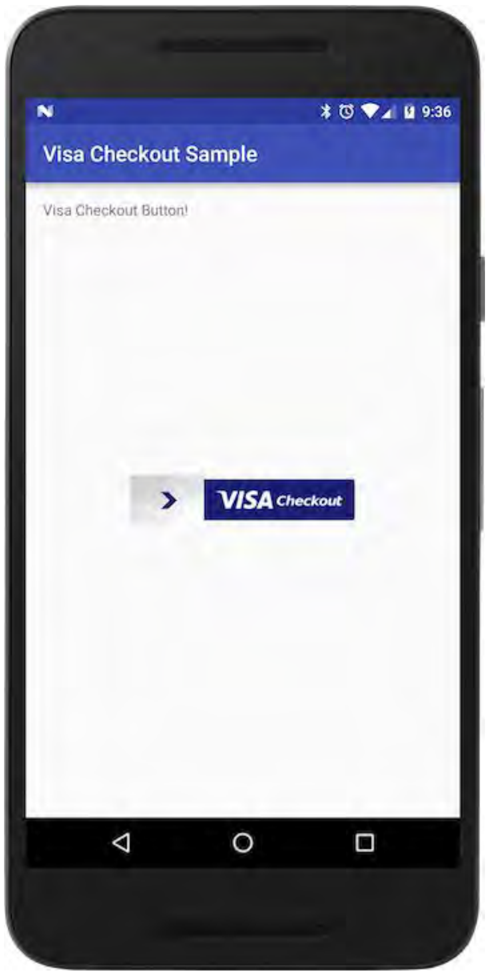
  Pass the `Intent` object and your request code to obtain the result.
- Override the Android `onActivityResult()` method to receive a `PaymentSummary` object from Visa Checkout, which contains the encrypted payload, if requested and available.

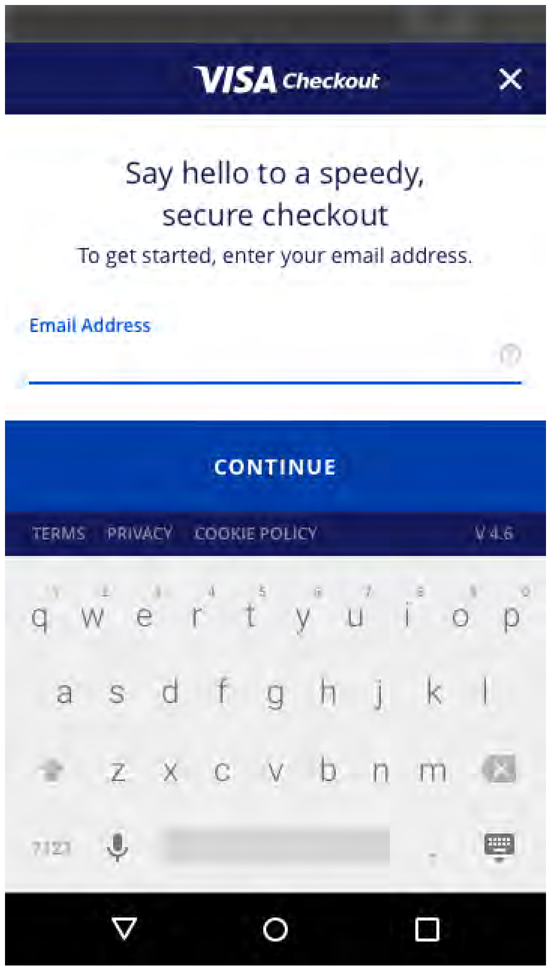
The following screen shows a sample `MainActivity` class that highlights the code described in this step:



```java
public class PaymentStartActivity extends FragmentActivity {

    private static final String TAG = PaymentStartActivity.class.getSimpleName();
    public final static int VISA_CHECKOUT_REQUEST_CODE = 10102;

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_vxo);

        VisaCheckoutSdk.init(getApplicationContext(), Environment.SANDBOX,
            "MERCHANT-API-KEY", "MERCHANT-PROFILE",
            new VisaCheckoutSdkInitListener() {
                @Override public void status(int code, String message) {
                    Log.v(TAG, "Code:" + code + "  Message:" + message);
                }
            });

        ((VisaCheckoutButton) findViewById(R.id.visaCheckoutButton)).setCheckoutListener(
            new VisaCheckoutButton.CheckoutWithVisaListener() {

                @Override public void onClick() {
                    Intent intent = VisaCheckoutSdk.getCheckoutIntent(PaymentStartActivity.this,
                        ConfigureVisaPaymentInfo.getPurchaseInfo());
                    startActivityForResult(intent, VISA_CHECKOUT_REQUEST_CODE);
                }
            });
    }

    @Override protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == VISA_CHECKOUT_REQUEST_CODE) {
            Log.d(TAG, "Result got back from Visa Checkout SDK");
            String msg = null;
            Bundle bundle = new Bundle();
            if (resultCode == RESULT_OK && data != null) {
                VisaPaymentSummary paymentSummary =
                    data.getParcelableExtra(VisaCheckoutSdk.INTENT_PAYMENT_SUMMARY);
            } else if (resultCode == RESULT_CANCELED) {
                msg = "User Canceled, Result Code : " + resultCode;
            } else if (resultCode == VisaCheckoutSdk.ResultCode.RESULT_SDK_NOT_INITIALIZED) {
                msg = "Sdk not initialized  failed, Result Code : " + resultCode;
            } else if (resultCode == VisaCheckoutSdk.ResultCode.RESULT_INITIALIZED_FAILED) {
                msg = "VisaPaymentInfo validation failed, Result Code : " + resultCode;
            } else {
                msg = "Purchase failed!";
            }
            Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
```

6. When you run your app, the Visa Checkout button appears and responds to events. Clicking the Visa Checkout button displays the following **Sign In** screen:




VISA Checkout
Say hello to a speedy, secure checkout
To get started, enter your email address.
Email Address
CONTINUE
TERMS
PRIVACY
COOKIE POLICY
V 4.6

Specifying Profile Overrides and Additional Purchase Information

The Visa Checkout Quick Start shows the basic steps to integrate Visa Checkout into your app. After completing Steps 1–4 in that section, to specify changes to the profile and to provide additional purchase information:

1. Modify your `MainActivity` class to create a `Profile` object using the `ProfileBuilder` method:

```
Profile profile =
    new Profile.ProfileBuilder(
        "MERCHANT-API-KEY",
        Environment.SANDBOX)
        .setProfileName("PROFILE-NAME")
        .setBillingCountries(new String[] {
            Profile.Country.US, Profile.Country.CA})
        .setShippingCountries(new String[] {
            Profile.Country.US, Profile.Country.CA })
        .setCardBrands(new String[] {
            Profile.CardBrand.VISA, Profile.CardBrand.MASTERCARD,
            Profile.CardBrand.AMEX, Profile.CardBrand.DISCOVER})
        .setDateLevel(Profile.DataLevel.SUMMARY)
        .setEnableTokenization(true)
        .setDisplayName("")
        .setAcceptCanadianVisaDebit(true)
        .build();
...);
```

The arguments, in order, are your API key, the environment, in this case the Sandbox, a profile name, followed by calls to the setter that methods specify values that override the profile. The `ProfileBuilder` method's `build()` method creates the profile. For a complete list of setter methods, see Profile Builder.

2. Call the SDK's `init()` method to initialize the SDK with your profile overrides:

```
VisaCheckoutSdk.init(getApplicationContext(), profile,
    new VisaCheckoutSdkInitListener() {
        @Override public void status(int code, String message) {
            Log.v(TAG,"Code:"+code+" Message:"+message);
        }
    });
```

- You must specify the following parameters:
  - The activity's context, which is obtained by calling the activity's `getApplicationContext()` method
  - The profile object you created in the previous step, which includes your API key, the environment, and the profile settings.
  - A `VisaCheckoutSdkInitListener` object, which receives the status after initialization.

3. Set up an `onClickListener` event for the Visa Checkout button that you added via a layout.
   - Create a `PurchaseInfo` object using the `PurchaseInfoBuilder` method:

```
PurchaseInfo purchaseInfo =
   new PurchaseInfo.PurchaseInfoBuilder(new BigDecimal("12.34"),
    PurchaseInfo.Currency.USD).setTax(new BigDecimal("2.14"))
    .setOrderId("2312SD-SD23-DSF234")
    .setThreeDSSetup(true, true)
    .setUserReviewAction(PurchaseInfo.UserReviewAction.PAY)
    .build();
```

   At a minimum, pass the subtotal, total, and currency associated with the transaction. You can call setters on the `PurchaseInfo` object to set additional fields. For a complete list of setter methods, see Purchase Info.

- Call the SDK's `getCheckoutIntent()` method to obtain an Android `Intent` object with which to launch the Visa Checkout SDK when user taps the button.

  In your call to `getCheckoutIntent()`, specify the activity or fragment from which Visa Checkout launches and the `PurchaseInfo` object you created:

  ```
  Intent intent =
      VisaCheckoutSdk.getCheckoutIntent(
        MainActivity1.this, purchaseInfo);
  ```

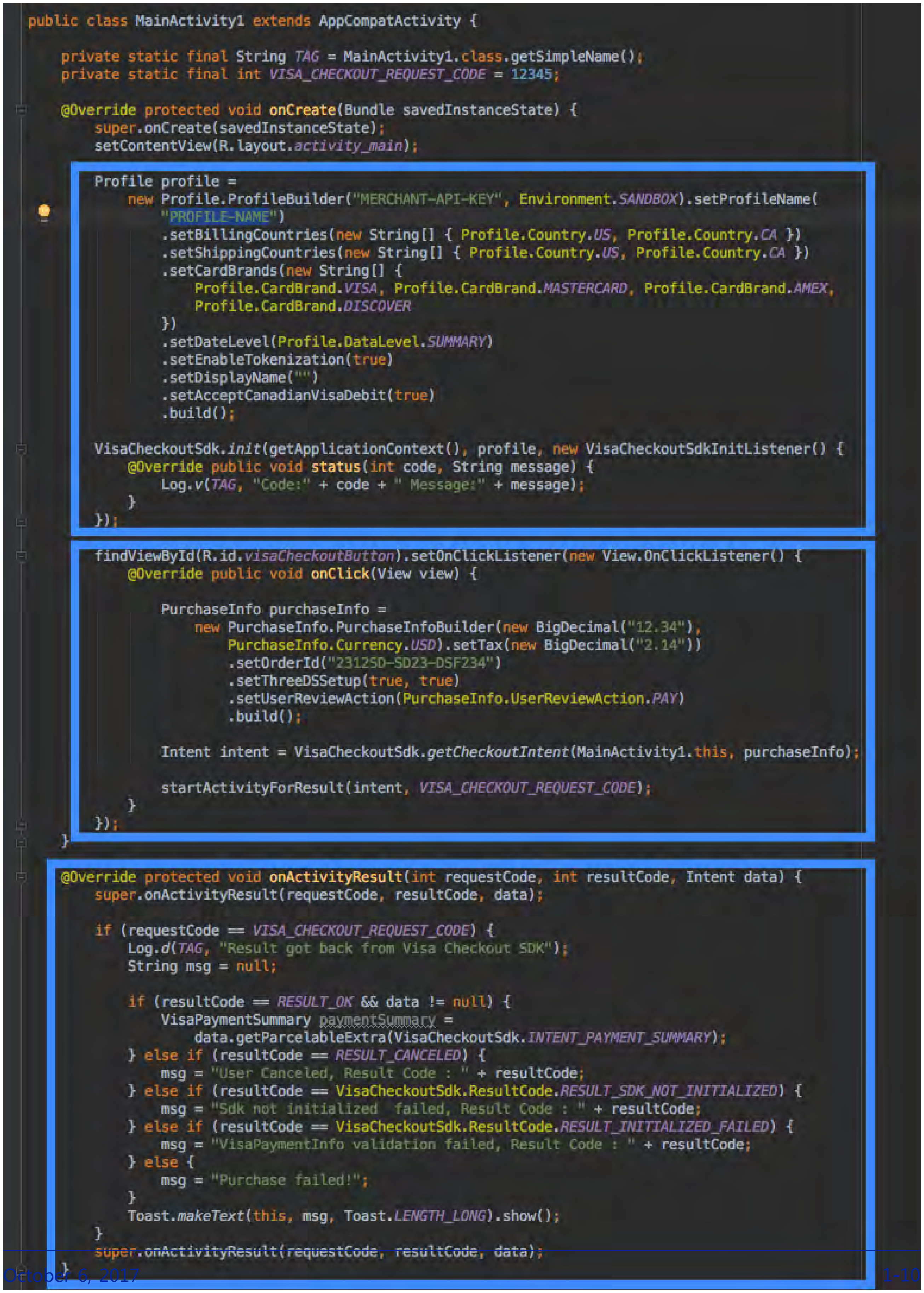
- After obtaining the intent, call the `startActivityForResult()` method to initiate processing.

  Pass the `Intent` object and your request code to obtain the result.

4. Override the Android `onActivityResult()` method to receive a `PaymentSummary` object from Visa Checkout, which contains the encrypted payload, if requested and available.

Note: *Specifying profile overrides and providing additional purchase information are independent of each other. Step 1 explains how to override a profile. Step 3 explains how to provide additional purchase information. Steps 2 and 4 are always required.*

The following screen shows a sample `MainActivity` class that highlights the code as described in these steps:



```
public class MainActivity1 extends AppCompatActivity {

    private static final String TAG = MainActivity1.class.getSimpleName();
    private static final int VISA_CHECKOUT_REQUEST_CODE = 12345;

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Profile profile =
            new Profile.ProfileBuilder("MERCHANT-API-KEY", Environment.SANDBOX).setProfileName(
                "PROFILE-NAME")
                .setBillingCountries(new String[] { Profile.Country.US, Profile.Country.CA })
                .setShippingCountries(new String[] { Profile.Country.US, Profile.Country.CA })
                .setCardBrands(new String[] {
                    Profile.CardBrand.VISA, Profile.CardBrand.MASTERCARD, Profile.CardBrand.AMEX,
                    Profile.CardBrand.DISCOVER
                })
                .setDateLevel(Profile.DataLevel.SUMMARY)
                .setEnableTokenization(true)
                .setDisplayName("")
                .setAcceptCanadianVisaDebit(true)
                .build();

        VisaCheckoutSdk.init(getApplicationContext(), profile, new VisaCheckoutSdkInitListener() {
            @Override public void status(int code, String message) {
                Log.v(TAG, "Code:" + code + " Message:" + message);
            }
        });

        findViewById(R.id.visaCheckoutButton).setOnClickListener(new View.OnClickListener() {
            @Override public void onClick(View view) {

                PurchaseInfo purchaseInfo =
                    new PurchaseInfo.PurchaseInfoBuilder(new BigDecimal("12.34"),
                        PurchaseInfo.Currency.USD).setTax(new BigDecimal("2.14"))
                        .setOrderId("23125D-SD23-DSF234")
                        .setThreeDSSetup(true, true)
                        .setUserReviewAction(PurchaseInfo.UserReviewAction.PAY)
                        .build();

                Intent intent = VisaCheckoutSdk.getCheckoutIntent(MainActivity1.this, purchaseInfo);

                startActivityForResult(intent, VISA_CHECKOUT_REQUEST_CODE);
            }
        });
    }

    @Override protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == VISA_CHECKOUT_REQUEST_CODE) {
            Log.d(TAG, "Result got back from Visa Checkout SDK");
            String msg = null;

            if (resultCode == RESULT_OK && data != null) {
                VisaPaymentSummary paymentSummary =
                    data.getParcelableExtra(VisaCheckoutSdk.INTENT_PAYMENT_SUMMARY);
            } else if (resultCode == RESULT_CANCELED) {
                msg = "User Canceled, Result Code : " + resultCode;
            } else if (resultCode == VisaCheckoutSdk.ResultCode.RESULT_SDK_NOT_INITIALIZED) {
                msg = "Sdk not initialized  failed, Result Code : " + resultCode;
            } else if (resultCode == VisaCheckoutSdk.ResultCode.RESULT_INITIALIZED_FAILED) {
                msg = "VisaPaymentInfo validation failed, Result Code : " + resultCode;
            } else {
                msg = "Purchase failed!";
            }
            Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
```

Testing Your Visa Checkout-Enabled App

For first time apps, use the Google Play Beta-testing and staged rollouts feature to validate the production release before publishing it to all users in the Play Store. For more information, see support.google.com/googleplay/android-developer/answer/3131213?hl=en

Test by making end-to-end payments against in the Visa Checkout sandbox and then in production for common scenarios including:

- Existing consumers (with payment information and shipping address)
- New consumers (sign up and add payment information and shipping address)

Once your app has been tested in the Visa Checkout sandbox, as well as production environments, verify that you have taken the following actions before making the final push to the Google Play Store:

- Operations against the payload are performed only on your back end server.
- All logging has been disabled.
- Appropriate time outs on the user interface have been implemented to handle potential latency issues or network glitches.

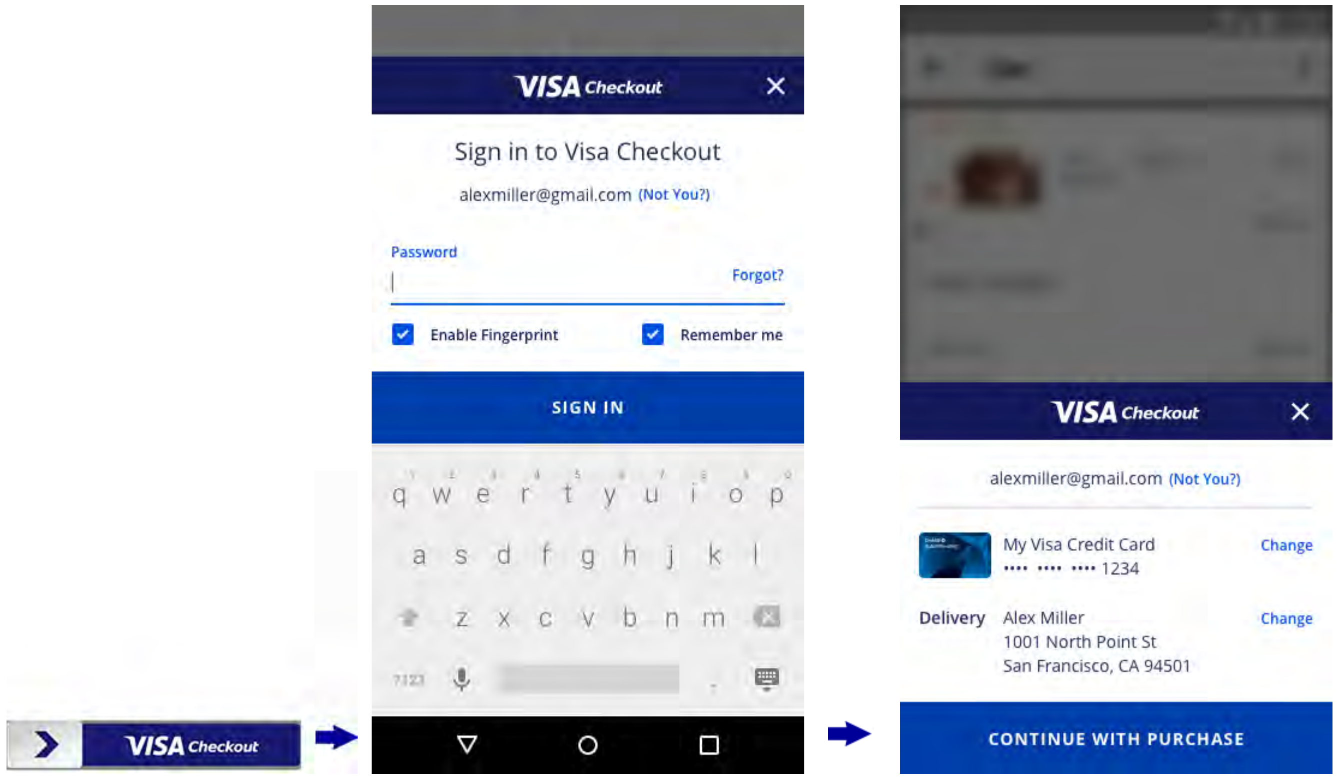
Additional Integration Information 2

The following sections provide additional information that may be useful for your integration.

Visa Checkout Consumer Experience on Android

The consumer taps the checkout with Visa Checkout button in your app, which launches the Visa Checkout flow. The consumer then logs into Visa Checkout to confirm the payment information and shipping address and continue with the purchase, possibly changing shipping or payment information before continuing back to your app. At that point, your app confirms a the successful purchase with Visa Checkout.

The Visa Checkout SDK for Android enables you to integrate a Visa Checkout payment flow into your Android app. After following the integration steps described in this document, your customer can check out and pay with Visa Checkout:



Visa Checkout Integration Options

The Visa Checkout mobile payment flow supports several different integration options in order to provide the consumers with a streamlined experience. These options include standard integration and suppressing shipping information altogether.

Standard Integration

In a standard integration, you specify all required and optional fields to be displayed:

- Countries for which shipping is available
- Accepted payment methods
- Your merchant logo or name
- An amount and associated currency
- Information about the payment instrument

This enables a consumer to login and only have to confirm or change this information.

Skip Shipping Information

In an integration that does not require shipping information, such as a payment for a download, you specify all required and optional fields except the shipping address and you specify that shipping is not required, which causes the shipping address panel to not be displayed.

User Interface Options

You can choose either the standard Visa Checkout button or an express checkout button.

- Visa Checkout button, which can range in size from 154 pixels wide, as shown below. For information about integrating this kind of button, see *Visa Checkout Integration Guide*.
- Express checkout button, which enables the consumer to enter the password on the button image after it is clicked. It has only 1 size, which is 213 pixels wide..

You can also use a payment mark, which is an image of just the card art by itself. For information about integrating this option, see .

Card Art

Enable the card art. If your payment page has a tabbed interface, display the payment mark card art using the following method:

```
VisaCheckoutSdk.getCardArt(getApplicationContext());
```

Public and Private Key Security

If you are calling the Visa Checkout API, you will need to follow these rules to implement essential security controls:

- At a high-level, the security of server communications is provided by the use of public-private key pairs. You communicate to Visa Checkout with your *API key*. You include a *shared secret*, along with data that needs to be protected from tampering, all of which are encrypted using an SHA256-bit hashing algorithm.

  Important:

  > You must only use a key and shared secret provided to you by Visa; specifically, you cannot communicate with Visa Checkout using keys or secrets that are not authorized for your use.

- YOU ARE SOLELY RESPONSIBLE FOR MAINTAINING ADEQUATE SECURITY AND CONTROL OF ANY API OR SHARED SECRET KEYS PROVIDED TO YOU. Because the shared secret ensures secure communications between you and Visa Checkout, you must protect the shared secret, allowing only authorized and authenticated entities, e.g. people, APIs, code, etc., to access the shared secret. The shared secret should never be stored or available unencrypted on a web page. The shared secret must be protected using either hardware- or software- based strong encryption, such as AES. In addition, you must provide your own secure server to store the encrypted shared secret. Because the shared secret is hashed along with changing data, you will need to create hash strings in real time.

Visa Checkout SDK for Android Information Collection

The Visa Checkout SDK for Android collects Visa Checkout enrollment and sign-in information, attributes of the device upon which it is installed, and de-identified usage information of the SDK. Visa Checkout uses and shares such information in accordance with the Visa Checkout privacy policy. The SDK may send information directly to its service providers to facilitate Visa's use and analysis of such information.

Visa recommends you consult your privacy or legal department to determine any required notices or consents in connection with your incorporation and activation of the Visa Checkout SDK for Android.

Proguard Configuration

You can use the following Proguard configuration in your application:

```
# Visa Checkout SDK specific proguard rules.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# suppress debug logs
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    public static *** v(...);
}

-dontwarn com.visa.**
-dontwarn com.google.gson.**
-dontwarn com.threatmetrix.**
-dontwarn com.google.**
#threatmetrix support library
-dontwarn okhttp3.**
-dontwarn okio.**

-keep class com.visa.** { *; }
-keep class com.threatmetrix.** { *; }
-keep class okio.** { *; }
-keep class okhttp3.** { *; }
-keep class com.google.gson.** { *; }

# Gson specific classes
-keep class sun.misc.Unsafe { *; }

-keepattributes Signature
-keepattributes Exceptions

# For Gson
# For using GSON @Expose annotation
-keepattributes *Annotation*

-keepnames class * implements android.os.Parcelable {
    public static final ** CREATOR;
}
```

SDK Reference 3

init()

Initializes the Visa Checkout SDK. You specify your activity's context, profile information, and a listener to retrieve the result.

There are 2 signatures for this method: you can either specify your environment, API key, and the existing profile you want to use, or you can specify a `Profile` object whose methods specify those and additional parameters. With either syntax, you must specify the environment and your API key. For information about parameters you can specify using an `Profile` object, see Profile.

Syntax

Syntax 1:

```
static void init(
  android.content.Context context,
  java.lang.String environment,
  java.lang.String apiKey,
  java.lang.String profileName,
  com.visa.checkout.VisaCheckoutSdkInitListener listener)
```

Syntax 2:

```
static void init(
  android.content.Context context,
  com.visa.checkout.Profile profile,
  com.visa.checkout.VisaCheckoutSdkInitListener listener)
```

Parameters

Parameter	Description
`context`	The activity's context, which is obtained by calling the activity's `getApplicationContext()` method
`environment`	The environment, which is one of the following values: • Profile.Environment.SANDBOX • Profile.Environment.PRODUCTION

Parameter	Description
`apiKey`	Your API key.
`profileName`	The name of a profile, which you set up after creating your account at `developer.visa.com`.
`listener`	A `VisaCheckoutSdkInitListener` object, which receives the status after SDK initialization.
`profile`	A `Profile` object that specifies parameters associated with a profile; for more information, see Profile.

Examples

Syntax 1:

```
VisaCheckoutSdk.init(
  getApplicationContext(),
  Profile.Environment.SANDBOX,
    "MERCHANT-API-KEY",
    "PROFILE-NAME",
    new VisaCheckoutSdkInitListener(){
      @Override public void status(int code, String message){
       }});
```

Syntax 2:

```
Profile profile =
    new Profile.ProfileBuilder(
        "MERCHANT-API-KEY",
        Environment.SANDBOX)
        .setProfileName("PROFILE-NAME")
        .setBillingCountries(new String[] {
            Profile.Country.US, Profile.Country.CA})
        .setShippingCountries(new String[] {
            Profile.Country.US, Profile.Country.CA })
        .setCardBrands(new String[] {
            Profile.CardBrand.VISA, Profile.CardBrand.MASTERCARD,
            Profile.CardBrand.AMEX, Profile.CardBrand.DISCOVER})
        .setDateLevel(Profile.DataLevel.SUMMARY)
        .setEnableTokenization(true)
        .setDisplayName("")
        .setAcceptCanadianVisaDebit(true)
        .build();
...);

VisaCheckoutSdk.init(getApplicationContext(), profile,
    new VisaCheckoutSdkInitListener() {
        @Override public void status(int code, String message) {
            Log.v(TAG,"Code:"+code+" Message:"+message);
        }
    });
```

getCheckoutIntent()

Obtain an Android `Intent` object with which to launch the Visa Checkout SDK. You use the `Intent` object when you call the Android `startActivityForResult()` method.

There are 2 signatures for this method: you can either specify the subtotal, total and currency for the purchase, or you can specify a `PurchaseInfo` object whose methods specify those and additional parameters. With either syntax, you must specify the subtotal, total and currency for the purchase. For information about parameters you can specify using an `PurchaseInfo` object, see Purchase Info.

Syntax

Syntax 1:

```
static android.content.Intent getCheckoutIntent(
  android.app.Activity activity,
  java.math.BigDecimal total,
  java.lang.String currency)
```

Syntax 2:

```
static android.content.Intent getCheckoutIntent(
  android.app.Activity activity,
  com.visa.checkout.PurchaseInfo purchaseInfo)
```

Parameters

Parameter	Description
`activity`	Instance of the activity from which Visa Checkout is launched.
`total`	Total of the payment including all amounts.
`currency`	Currency code.
`purchaseInfo`	A `PurchaseInfo` object that specifies parameters associated with the purchase; for more information, see Purchase Info.

Returns

Returns an Android `Intent` object.

Examples

Syntax 1:

```
Intent intent =
  VisaCheckoutSdk.getCheckoutIntent(
    MainActivity.this,
    new BigDecimal("12.34"),
    PurchaseInfo.Currency.USD);
```

Syntax 2:

```
PurchaseInfo purchaseInfo =
    new PurchaseInfo.PurchaseInfoBuilder(
      new BigDecimal("12.34"), PurchaseInfo.Currency.USD)
      .setTax(new BigDecimal("2.14"))
      .setOrderId("2312SD-SD23-DSF234")
      .setThreeDSSetup(true, true)
      .setUserReviewAction(PurchaseInfo.UserReviewAction.PAY)
      .build();

Intent intent =
  VisaCheckoutSdk.getCheckoutIntent(
    MainActivity.this,
    purchaseInfo);
```

getCardArt()

Obtains the card art.

Syntax

```
static android.graphics.drawable.Drawable
getCardArt(android.content.Context context)
```

Parameters

Parameter	Description
`context`	The activity's context, which is obtained by calling the activity's `getApplicationContext()` method

Returns

Returns an Android `Drawable` object.

Examples

```
VisaCheckoutSdk.getCardArt(getApplicationContext());
```

setCampaignData()

Specifies campaign data.

Syntax

```
static setCampaignData(java.lang.String url)
```

Parameters

Parameter	Description
`url`	A `String` object that contains the campaign URL.

Examples

```
String campaignData = "CAMPAIGN-DATA-URL";
 VisaCheckoutSdk.setCampaignData(campaignData);
```

status()

Implement the `status()` method in the `VisaCheckoutSdkInitListener` object to provide information about the status of the after initialization.

Syntax

```
void status(int code, java.lang.String message)
```

Parameters

Parameter	Description
`code`	An `int` that specifies the result code. It is one of the following `VisaCheckoutSdk.Status` values: • `SUCCESS` • `INVALID_API_KEY` • `UNSUPPORTED_SDK_VERSION` • `OS_VERSION_NOT_SUPPORTED` • `FAIL_TO_INITIALIZE` • `MISSING_PARAMETER` • `INTERNAL_ERROR` • `SDK_PAUSED` • `SDK_RESUMED`
`message`	A `String` object that specifies a message associated with the result code.

Profile

```
public class Profile
extends java.lang.Object
implements android.os.Parcelable
```

Getter and setter methods	Field Description
`String[] getAcceptedBillingCountries()`	Get an override value for billing country codes in the merchant's external or default profiles. The override value limits the selection of eligible cards in the consumer's account. If not set in a profile or overridden here, payments from all listed billing countries are accepted. Override value for billing country codes in the merchant's external or default profiles. The override value limits the selection of eligible cards in the consumer's account. If not set in a profile or overridden here, payments from all listed billing countries are accepted. **Format:** One of the following ISO-3166-1 alpha-2 standard codes: • `AR` - Argentina (Since 2.7) • `AU` - Australia • `BR` - Brazil (Since 2.7) • `CA` - Canada • `CN` - China (Since 2.9) • `CL` - Chile (Since 2.9) • `CO` - Colombia (Since 2.9) • `FR` - France (Since 4.3) • `HK` - Hong Kong (Since 2.9) • `IN` - India (Since 4.6) • `IE` - Ireland (Since 4.3) • `KW` - Kuwait (Since 5.2) • `MY` - Malaysia (Since 2.9) • `MX` - Mexico (Since 2.9) • `NZ` - New Zealand (Since 2.9) • `PE` - Peru (Since 2.9) • `PL` - Poland (Since 4.3) • `QA` - Qatar (Since 5.2) • `SA` - Saudi Arabia (Since 5.12) • `SG` - Singapore • `ZA` - South Africa (Since 2.9) • `ES` - Spain (Since 4.3) • `UA` - Ukraine (Since 5.2) • `AE` - United Arab Emirates (Since 2.9) • `GB` - United Kingdom (Since 4.3) • `US` - United States Since 2.0

Getter and setter methods	Field Description
`String[] getAcceptedCardBrands()`	Get card brands that are accepted. The `VISA` brand must be included if a Canadian Debit card is used. If not set in a profile or overridden here, all listed card brands are accepted. **Format:** Array containing one or more of the following brands: • `VISA` • `MASTERCARD` • `AMEX` • `DISCOVER` • `ELECTRON` (Brazil only; since 3.9) • `ELO` (Brazil only; since 3.9) Since 2.0

Getter and setter methods	Field Description
`String[] getAcceptedShippingCountries()`	Get an override value for shipping country codes in the merchant's external or default profiles. The override value limits the selection of eligible cards in the consumer's account. If not set in a profile or overridden here, payments from all listed shipping countries are accepted. **Format:** One of the following ISO-3166-1 alpha-2 standard codes: • `AR` - Argentina (Since 2.7) • `AU` - Australia • `BR` - Brazil (Since 2.7) • `CA` - Canada • `CN` - China (Since 2.9) • `CL` - Chile (Since 2.9) • `CO` - Colombia (Since 2.9) • `FR` - France (Since 4.3) • `HK` - Hong Kong (Since 2.9) • `IN` - India (Since 4.6) • `IE` - Ireland (Since 4.3) • `KW` - Kuwait (Since 5.2) • `MY` - Malaysia (Since 2.9) • `MX` - Mexico (Since 2.9) • `NZ` - New Zealand (Since 2.9) • `PE` - Peru (Since 2.9) • `PL` - Poland (Since 4.3) • `QA` - Qatar (Since 5.2) • `SA` - Saudi Arabia (Since 5.2) • `SG` - Singapore • `ZA` - South Africa (Since 2.9) • `ES` - Spain (Since 4.3) • `UA` - Ukraine (Since 5.2) • `AE` - United Arab Emirates (Since 2.9) • `GB` - United Kingdom (Since 4.3) • `US` - United States Since 2.0
`String getApiKey()`	Get the API key that Visa Checkout created when you created the Visa Checkout account. You need to use both a live key and a sandbox key. They are different from each other. For more information, see . **Format:** Alphanumeric; maximum 100 characters Since 2.0

Getter and setter methods	Field Description
`String getDataLevel()`	Get the level of consumer and payment information that the `payment.success` event response needs to include. If you request information, permission to receive full information must be configured in Visa Checkout; otherwise, you l always receive only summary information, regardless of the data level that you specify. When onboarded by a partner, the `enablePANAccess` field of the onboarding API determines the default value for `dataLevel`. If `enablePANAccess` is `true` when the merchant's information that is returned by `getDataLevel()` during onboarding, the default `dataLevel` is `true`; otherwise, the default `dataLevel` is `false`. For information about the `enablePANAccess` field in the onboarding API, see the Client API Reference, Partner Edition. **Format:** It is one of the following values: • `SUMMARY` - Summary information • `FULL` - Full information, which is only available if you are configured to receive it • `NONE` - Consumer and payment information is not returned in the `payment.success` event response, in which case the Get Payment Data API must be used to obtain the information. Since 2.5. Since 2.0
`String getDisplayName()`	Get the merchant's name as it appears on the **Review** panel of the lightbox; typically, it is the name of your company. **Format:** Maximum 99 characters, either alphabetic, numeric, spaces, or the following characters: `! @ # $ % ^ & * – ' ?` and period (`.`) Since 2.0
`String getEnvironment()`	Get the environment, which is one of the following values: • Profile.Environment.SANDBOX • Profile.Environment.PRODUCTION
`getExternalProfileId()`	Get the Profile ID, which is created externally by a merchant or partner. Visa Checkout uses the Profile ID to populate settings, such as accepted card brands and shipping regions. The properties set in this profile override properties in the merchant's current profile. **Format:** Alphanumeric; maximum 50 characters Since 2.0
`int getLogoResourceId()`	Get the ID of a drawable resource that includes the logo. the logo appears on the **Review** and **Continue** pages.
`String getMerchantId()`	Get the acquirer-issued merchant ID. **Format:** Alphanumeric Since 2.8
`String getProfileName()`	Get the profile name that you set up after creating your account at `developer.visa.com`.

Getter and setter methods	Field Description
`boolean isAcceptCanadianVisaDebit()`	Whether a Canadian merchant accepts Visa Canada debit cards; ignored for non-Canadian merchants. Visa must be specified as an allowable card brand. **Format:** One of the following values: • `true` - Visa Canada debit cards accepted • `false` - Visa Canada debit cards not accepted Since 2.0
`boolean isEnableTokenization()`	Whether tokenization is enabled. It is one of the following values: • `true` — tokenization enabled • `false` — tokenization disabled
`boolean isFBAppEventsEnabled()`	Whether Facebook tracking is enabled. It is one of the following values: • `true` — tracking enabled • `false` — tracking disabled

Profile Builder

```
public static final class Profile.ProfileBuilder
extends java.lang.Object
```

Getter and setter methods	Field Description
`setAcceptCanadianVisaDebit( boolean acceptCanadianVisaDebit)`	*(Optional)* Set the override for a Canadian merchant to accept or reject Visa Canada debit cards; this is ignored for non-Canadian merchants. Visa must be specified as an allowable card brand. **Format:** One of the following values: • `true` - Visa Canada debit cards accepted • `false` - Visa Canada debit cards not accepted Since 2.0
`setBillingCountries( String[] billingCountries)`	*(Optional)* Set the override value for billing country codes in the merchant's external or default profiles. It limits selection of eligible cards in the consumer's account. If not set in a profile or overridden here, payments from all listed billing countries are accepted. **Format:** One of the following ISO-3166-1 alpha-2 standard codes: • `AR` - Argentina (Since 2.7) • `AU` - Australia • `BR` - Brazil (Since 2.7) • `CA` - Canada • `CN` - China (Since 2.9) • `CL` - Chile (Since 2.9) • `CO` - Colombia (Since 2.9) • `FR` - France (Since 4.3) • `HK` - Hong Kong (Since 2.9) • `IN` - India (Since 4.6) • `IE` - Ireland (Since 4.3) • `KW` - Kuwait (Since 5.2) • `MY` - Malaysia (Since 2.9) • `MX` - Mexico (Since 2.9) • `NZ` - New Zealand (Since 2.9) • `PE` - Peru (Since 2.9) • `PL` - Poland (Since 4.3) • `QA` - Qatar (Since 5.2) • `SA` - Saudi Arabia (Since 5.2) • `SG` - Singapore • `ZA` - South Africa (Since 2.9) • `ES` - Spain (Since 4.3) • `UA` - Ukraine (Since 5.2) • `AE` - United Arab Emirates (Since 2.9) • `GB` - United Kingdom (Since 4.3) • `US` - United States Since 2.0

Getter and setter methods	Field Description
`setCardBrands( String[] cardBrands)`	*(Optional)* Set the card brands that are accepted. The `VISA` brand must be included if `acceptCanadianVisaDebit` is `true`. If not set in a profile or overridden here, all listed card brands are accepted. **Format:** Array containing one or more of the following brands: • `VISA` • `MASTERCARD` • `AMEX` • `DISCOVER` • `ELECTRON` (Brazil only; since 3.9) • `ELO` (Brazil only; since 3.9) Since 2.0
`setDataLevel(String dataLevel)`	*(Optional)* Set the level of consumer and payment information that the `payment.success` event response needs to include. If you request information, permission to receive full information must be configured in Visa Checkout; otherwise, you always receive only summary information, regardless of the data level that you specify. When onboarded by a partner, the `enablePANAccess` field of the onboarding API determines the default value for `dataLevel`. If `enablePANAccess` is `true` when the merchant is onboarded, the default `dataLevel` is `true`; otherwise, the default `dataLevel` is `false`. For information about the `enablePANAccess` field in the onboarding API, see the Client API Reference, Partner Edition. **Format:** It is one of the following values: • `SUMMARY` - Summary information • `FULL` - Full information, which is only available if you are configured to receive it • `NONE` - Consumer and payment information is not returned in the `payment.success` event response, in which case the Get Payment Data API must be used to obtain the information. Since 2.5. Since 2.0
`setDisplayName(String displayName)`	*(Optional)* Set the merchant's name as it appears on the **Review** panel of the lightbox; typically, it is the name of your company. **Format:** Maximum 99 characters, either alphabetic, numeric, spaces, or the following characters: `! @ # $ % ^ & * - ' ?` and period (`.`) Since 2.0
`setEnableTokenization( boolean enableTokenization)`	*(Optional)* Set tokenization to be enabled or disabled. It is one of the following values: • `true` — tokenization enabled • `false` — tokenization disabled (default)

Getter and setter methods	Field Description

`setExternalProfileId(`
`  String externalProfileId)`
	(Optional) Set the Profile ID that a merchant or partner has created externally. Visa Checkout uses the Profile ID to populate settings, such as accepted card brands and shipping regions. The properties that are set in this profile override properties in the merchant's current profile.
	
	Format: Alphanumeric; maximum 50 characters
	
	Since 2.0

`setEnableFbTracking(`
`  String fbAdvertiserId,`
`  String fbApplicationId)`
	(Optional) Passes the Android Advertiser Id and Facebook application Id to track the Facebook event.

`setLogoResourceId(`
`  int logoResourceId)`
	Set the ID of a drawable resource that includes the logo. the logo appears on the **Review** and **Continue** pages.

`setMerchantId(String merchantId)`	*(Required)* Set the acquirer-issued merchant ID. **Format:** Alphanumeric Since 2.8

Getter and setter methods	Field Description
`setProfileName(String profileName)`	Set the profile name that you set up after creating your account at `developer.visa.com`.
`setShippingCountries( String[] shippingCountries)`	Set an override value for shipping country codes in the merchant's external or default profiles. The override value limits the selection of eligible cards in the consumer's account. If not set in a profile or overridden here, payments from all listed shipping countries are accepted. **Format:** One of the following ISO-3166-1 alpha-2 standard codes: • `AR` - Argentina (Since 2.7) • `AU` - Australia • `BR` - Brazil (Since 2.7) • `CA` - Canada • `CN` - China (Since 2.9) • `CL` - Chile (Since 2.9) • `CO` - Colombia (Since 2.9) • `FR` - France (Since 4.3) • `HK` - Hong Kong (Since 2.9) • `IN` - India (Since 4.6) • `IE` - Ireland (Since 4.3) • `KW` - Kuwait (Since 5.2) • `MY` - Malaysia (Since 2.9) • `MX` - Mexico (Since 2.9) • `NZ` - New Zealand (Since 2.9) • `PE` - Peru (Since 2.9) • `PL` - Poland (Since 4.3) • `QA` - Qatar (Since 5.2) • `SA` - Saudi Arabia (Since 5.2) • `SG` - Singapore • `ZA` - South Africa (Since 2.9) • `ES` - Spain (Since 4.3) • `UA` - Ukraine (Since 5.2) • `AE` - United Arab Emirates (Since 2.9) • `GB` - United Kingdom (Since 4.3) • `US` - United States Since 2.0

Purchase Info

```
public class PurchaseInfo
extends java.lang.Object
implements android.os.Parcelable
```

Getter and setter methods	**Field Description**
`String getCurrency()`	Get the currency code. **Format:** It is an ISO 4217 standard alpha-3 code value or `ANY` for any value. Currency codes must be uppercase. Since 2.8
`getCustomData()`	Merchant can pass custom data with key value pair and have it returned in the `PaymentSummary`. **Format:** Alphanumeric; maximum 1024 characters Since 2.0
`String getDescription()`	Get the description that is associated with a payment. **Format:** Alphabetic characters, digits, spaces ( ), periods (**.**), underscores (_), and hyphens (**-**); maximum 100 characters Since 2.0
`BigDecimal getDiscount()`	Get the total for the discounts that are related to the payment. If provided, it is a positive value that represents the amount that needs to be deducted from the total. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0
`BigDecimal getGiftWrap()`	Get the total gift-wrapping charges that are included in the payment. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0
`String getMerchantRequestId()`	Get the Merchant's ID that is associated with the request. Visa Checkout stores this value for your use as a convenience. Since 2.0
`BigDecimal getMisc()`	Get the total uncategorized charges that are included in the payment. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0
`String getOrderId()`	Get the Merchant's order ID that is associated with the payment. **Format:** Alphanumeric; maximum 100 characters Since 2.0

Getter and setter methods	Field Description
`String getPromoCode()`	Get the promotion codes that are associated with the payment. **Format:** Alphabetic characters, digits, space ( ), underscore (_), hyphen (-), exclamation point (!), "at" sign (@), pound sign or hash mark (#), dollar sign ($), percent sign (%), asterisk (*), left/open parenthesis (( ), right/close parenthesis ( )), and plus sign (+). Multiple promotion codes are separated by a period (.); maximum 100 characters for the entire string Since 2.0
`String getReferenceCallId()`	Get the Visa Checkout transaction ID. See the *Visa Checkout Integration Guide*. **Format:** Alphanumeric; maximum 48 characters Since 2.6
`String getReviewMessage()`	Get the message that you want to display on the **Review** page. You are responsible for translating the message. **Format:** Maximum 100 characters, either alphabetic, numeric, spaces, or the following characters: `! @ # $ % ^ & * - ' ?` and period (.). ***Note:*** *If you exceed the maximum number of characters, the message might not appear correctly or it might not appear at all.* Since 2.0
`BigDecimal getShippingHandling()`	Get the total amount for the shipping and handling charges that is included in the payment. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0
`BigDecimal getSubTotal()`	Get the subtotal amount that is included in the payment. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0
`BigDecimal getTax()`	Get the total amount for the tax-related charges that is included in the payment. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0
`getThreeDSSetup()`	Get one or more name-value pairs that you can use to set up Verified by Visa properties with Visa Checkout; see . **Format:** `threeDSSetup` Since 2.8

Getter and setter methods	**Field Description**
`BigDecimal getTotal()`	Get the total amount of the payment, **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0
`String getUserReviewAction()`	Get the button label in the Visa Checkout lightbox. **Format:** One of the following values: • `Continue` - Display **Continue** on the lightbox button (default) • `Pay` - Display **Pay** on the lightbox button ***Note:*** *A valid value for* `total` *must be specified when using* ***Pay*** *on the button; otherwise* ***Continue*** *is displayed.* Since 2.0
`boolean isShippingAddressRequired()`	Whether the shipping address information is available. **Format:** `Address`; see *Visa Checkout Integration Guide*. Since 2.0

Purchase Info Builder

```
public static class final class PurchaseInfo.PurchaseInfoBuilder
extends java.lang.Object
```

Getter and setter methods	**Field Description**

`setCustomData(`
`  @NonNull java.util.HashMap`
	Merchant can pass custom data with a key value pair and have it returned in the `PaymentSummary`.
	
	Format: Alphanumeric; maximum 1024 characters
	
	Since 2.0

`setDescription(String description)`	Set the description that is associated with a payment. **Format:** Alphabetic characters, digits, spaces ( ), periods (**.**), underscores (_), and hyphens (**-**); maximum 100 characters Since 2.0
`setDiscount(BigDecimal discount)`	*(Optional)* Set the total number of discounts that are related to the payment. If provided, it is a positive value that represents the amount to be deducted from the total. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0
`setGiftWrap(BigDecimal giftWrap)`	*(Optional)* Set the total amount of gift-wrapping charges that are included in the payment. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0

Getter and setter methods	**Field Description**

`setMerchantRequestId(`
`  String merchantRequestId)`
	(Optional) Set the Merchant's ID that is associated with the request. Visa Checkout stores this value for your use as a convenience.
	
	Since 2.0

`setMisc(BigDecimal misc)`	*(Optional)* Set the total amount of uncategorized charges that are included n the payment. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits
`setOrderId(String orderId)`	*(Optional)* Set the Merchant's order ID that is associated with the payment. **Format:** Alphanumeric; maximum 100 characters Since 2.0
`setPromoCode(String promoCode)`	*(Optional)* Set the promotion codes that are associated with the payment. **Format:** Alphabetic characters, digits, space ( ), underscore (_), hyphen (**-**), exclamation point (!), "at" sign (`@`), pound sign or hash mark (#), dollar sign ($), percent sign (`%`), asterisk (*), left/open parenthesis (( ), right/close parenthesis ( )), and plus sign (+). Multiple promotion codes are separated by a period (**.**); maximum 100 characters for the entire string Since 2.0

`setReferenceCallId(`
`  String referenceCallId)`
	(Optional) Set the Visa Checkout transaction ID. See the *Visa Checkout Integration Guide*.
	
	Format: Alphanumeric; maximum 48 characters
	
	Since 2.6

`setReviewMessage(`
`  String reviewMessage)`
	Set the message that you want to display on the **Review** page. You are responsible for translating the message.
	
	Format: Maximum 100 characters, either alphabetic, numeric, spaces, or the following characters: `! @ # $ % ^ & * - ' ?` and period (**.**).
	
	Note: *If you exceed the maximum number of characters, the message might not appear correctly or it might not appear at all.*
	
	Since 2.0

`setShippingAddressRequired(`
`  boolean shippingAddressRequired)`
	Set the shipping address.
	
	Format: `Address`; see the *Visa Checkout Integration Guide*.
	
	Since 2.0

`setShippingHandling(`
`  BigDecimal shippingHandling)`
	(Optional) Set the total amount for shipping and handling charges that are included in the payment.
	
	Format: Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits
	
	Since 2.0

Getter and setter methods	Field Description
`setTax(BigDecimal tax)`	*(Optional)* Set the total amount of tax-related charges in the payment. **Format:** Numeric; maximum 9 digits before an optional decimal point and 4 decimal digits Since 2.0
`setThreeDSSetup( boolean threeDSActive, boolean threeDSSuppressChallenge)`	*(Optional)* Set up Verified by Visa properties with Visa Checkout; see the *Visa Checkout Integration Guide*. **Format:** `threeDSSetup` Since 2.8
`setUserReviewAction( String userReviewAction)`	Set the button label in the Visa Checkout lightbox. **Format:** One of the following values: • `Continue` - Display **Continue** on the lightbox button (default) • `Pay` - Display **Pay** on the lightbox button ***Note:*** *A valid value for* `total` *must be specified when using* ***Pay*** *on the button; otherwise* ***Continue*** *will be displayed.* Since 2.0
`setVisaUserInfo( VisaUserInfo visaUserInfo)`	Set user information, such as the billing and shipping addresses for easy onboarding.

Visa Payment Summary

```
public final class VisaPaymentSummary
extends java.lang.Object
implements android.os.Parcelable
```

Getter and setter methods	Field Description
`String getCallId()`	Get the Visa Checkout transaction ID. See the *Visa Checkout Integration Guide*. **Format:** Alphanumeric; maximum 48 characters Since 2.6
`String getCardBrand()`	Get the card brands that are accepted. The `VISA` brand must be included if a Canadian Debit card is used. If not set in a profile or overridden here, all listed card brands are accepted. **Format:** Array containing one or more of the following brands: • `VISA` • `MASTERCARD` • `AMEX` • `DISCOVER` • `ELECTRON` (Brazil only; since 3.9) • `ELO` (Brazil only; since 3.9) Since 2.0
`String getCardType()`	Get the kind of card. **Format:** It is one of the following values: • `CREDIT` • `DEBIT` • `CHARGE` • `PREPAID` • `DEFERRED_DEBIT` • `NONE` Since 2.0

Getter and setter methods	Field Description
`String getCountryCode()`	Get the override value for the country code, which controls how text displays in the Visa Checkout checkout button and lightbox. By default, Visa Checkout determines the country from the consumer's IP address. Do not use the `countryCode` attribute. See the *Visa Checkout Integration Guide*. **Format:** One of the following ISO-3166-1 alpha-2 standard codes: • `AR` - Argentina (Since 2.7) • `AU` - Australia • `BR` - Brazil (Since 2.7) • `CA` - Canada • `CN` - China (Since 2.9) • `CL` - Chile (Since 2.9) • `CO` - Colombia (Since 2.9) • `FR` - France (Since 4.3) • `HK` - Hong Kong (Since 2.9) • `IN` - India (Since 4.6) • `IE` - Ireland (Since 4.3) • `KW` - Kuwait (Since 5.2) • `MY` - Malaysia (Since 2.9) • `MX` - Mexico (Since 2.9) • `NZ` - New Zealand (Since 2.9) • `PE` - Peru (Since 2.9) • `PL` - Poland (Since 4.3) • `QA` - Qatar (Since 5.2) • `SA` - Saudi Arabia (Since 5.2) • `SG` - Singapore • `ZA` - South Africa (Since 2.9) • `ES` - Spain (Since 4.3) • `UA` - Ukraine (Since 5.2) • `AE` - United Arab Emirates (Since 2.9) • `GB` - United Kingdom (Since 4.3) • `US` - United States The value of the `country` attribute must be compatible with the value of the `locale` attribute. Since 2.0
`String getEncKey()`	Get the encrypted key to decrypt `encPaymentData`. You use your shared secret to decrypt this key. Since 2.0

Getter and setter methods	Field Description
`String getEncPaymentData()`	Get the encrypted consumer data to process the transaction. You decrypt the consumer data by first unwrapping the encrypted key value. You can then use the unwrapped key to decrypt the encrypted key value. Since 2.0
`String getLastFourDigits()`	Get the last 4 digits of the payment instrument. **Format:**Numeric; maximum 4 digits Since 3.0
`String getPostalCode()`	Get the postal code that is associated with the billing address. The postal code is only available to contractually-enabled merchants. **Format:** Depends on the country code; maximum 7 characters: • `US` must be 5 digits • `CA` must be 6 characters separated by a space or a hyphen, e.g. `A0A 0A0` • `AU` must be 4 digits Since 3.0
`long getTokenReceivedDate()`	Get Tokenized payment data.
`setCallId(String callId)`	*(Optional)* Set the Visa Checkout transaction ID. See the *Visa Checkout Integration Guide*. **Format:** Alphanumeric; maximum 48 characters Since 2.6
`setCardBrand(String cardBrand)`	*(Optional)* Card brands that are accepted. The `VISA` brand must be included if a Canadian Visa Debit card is used. If not set in a profile or overridden here, all listed card brands are accepted. **Format:** Array containing one or more of the following brands: • `VISA` • `MASTERCARD` • `AMEX` • `DISCOVER` • `ELECTRON` (Brazil only; since 3.9) • `ELO` (Brazil only; since 3.9) Since 2.0

Getter and setter methods	**Field Description**
`setCardType(String cardType)`	Set the kind of card. **Format:** It is one of the following values: • `CREDIT` • `DEBIT` • `CHARGE` • `PREPAID` • `DEFERRED_DEBIT` • `NONE` Since 2.0

Getter and setter methods	Field Description
`setCountryCode( String countryCode)`	*(Optional)* Set the override value for the country code, which controls how text displays in the Visa Checkout checkout button and lightbox. By default, Visa Checkout determines the country from the consumer's IP address. Do not use the `countryCode` attribute unless explicit control over the display is required. For more information, see . **Format:** One of the following ISO-3166-1 alpha-2 standard codes: • `AR` - Argentina (Since 2.7) • `AU` - Australia • `BR` - Brazil (Since 2.7) • `CA` - Canada • `CN` - China (Since 2.9) • `CL` - Chile (Since 2.9) • `CO` - Colombia (Since 2.9) • `FR` - France (Since 4.3) • `HK` - Hong Kong (Since 2.9) • `IN` - India (Since 4.6) • `IE` - Ireland (Since 4.3) • `KW` - Kuwait (Since 5.2) • `MY` - Malaysia (Since 2.9) • `MX` - Mexico (Since 2.9) • `NZ` - New Zealand (Since 2.9) • `PE` - Peru (Since 2.9) • `PL` - Poland (Since 4.3) • `QA` - Qatar (Since 5.2) • `SA` - Saudi Arabia (Since 5.2) • `SG` - Singapore • `ZA` - South Africa (Since 2.9) • `ES` - Spain (Since 4.3) • `UA` - Ukraine (Since 5.2) • `AE` - United Arab Emirates (Since 2.9) • `GB` - United Kingdom (Since 4.3) • `US` - United States The value of the `country` attribute must be compatible with the value of the `locale` attribute. Since 2.0
`setEncKey(String encKey)`	Set the encrypted key to decrypt `encPaymentData`. You use your shared secret to decrypt this key. Since 2.0

Getter and setter methods	Field Description
`setEncPaymentData( String encPaymentData)`	Set the encrypted consumer data to process the transaction. You decrypt the consumer data by first unwrapping the `encKey` value, then using that unwrapped key to decrypt this value. Since 2.0
`setLastFourDigits( String lastFourDigits)`	Set the last 4 digits of the payment instrument. **Format:**Numeric; maximum 4 digits Since 3.0
`setPostalCode( String postalCode)`	Set the postal code that is associated with the billing address. Only available to contractually-enabled merchants. **Format:** Depends on the country code; maximum 7 characters: • `US` must be 5 digits • `CA` must be 6 characters separated by a space or a hyphen, e.g. `A0A 0A0` • `AU` must be 4 digits Since 3.0

Visa User Address

```
public final class VisaUserAddress
extends java.lang.Object
implements android.os.Parcelable
```

Getter and setter methods	Field Description
`String getCity()`	Get the name of the city that is associated with the billing address. The name of the city is only available to contractually-enabled merchants. **Format:** Alphanumeric; maximum 100 characters Since 3.0
`String getLine1()`	Get the first line of the address. **Format:** Alphanumeric; maximum 140 characters Since 2.0
`String getLine2()`	Get the second line of the address, if available. **Format:** Alphanumeric; maximum 140 characters Since 2.0
`String getLine3()`	Get the third line of the address, if available. It is used only to include the name of the suburb for New Zealand addresses. **Format:** Alphanumeric; maximum 140 characters Since 2.7

Getter and setter methods	Field Description
`String getPersonName()`	Get the addressee's complete name. **Format:** Alphanumeric; maximum 49 characters Since 2.0
`String getPhone()`	Get the phone number that is associated with the billing address. Only available to contractually-enabled merchants. **Format:** Numeric or hyphens, parentheses, period, or plus sign, varies by country; maximum 15 characters Since 4.6
`String getPostalCode()`	Get the postal code that is associated with the billing address. The postal code is only available to contractually-enabled merchants. **Format:** Depends on the country code;, maximum 7 characters: • `US` must be 5 digits • `CA` must be 6 characters separated by a space or a hyphen, e.g. `A0A 0A0` • `AU` must be 4 digits Since 3.0
`String getStateProvinceCode()`	Get the name of the state or province code that is associated with the billing address. The name of the state or province code is only available to contractually-enabled merchants. You can select some of the country names, such as the US, AU ,IE, and other countries from a dropdown menu. If the name of a country cannot be selected from the dropdown menu, set the value to 100. **Format:** Alphanumeric; maximum 100 characters Since 3.0

Getter and setter methods	Field Description
`String getTwoCharacterCountryCode()`	Get a two-character country code, such as US and other country codes. **Format:** One of the following ISO-3166-1 alpha-2 standard codes: • `AR` - Argentina (Since 2.7) • `AU` - Australia • `BR` - Brazil (Since 2.7) • `CA` - Canada • `CN` - China (Since 2.9) • `CL` - Chile (Since 2.9) • `CO` - Colombia (Since 2.9) • `FR` - France (Since 4.3) • `HK` - Hong Kong (Since 2.9) • `IN` - India (Since 4.6) • `IE` - Ireland (Since 4.3) • `KW` - Kuwait (Since 5.2) • `MY` - Malaysia (Since 2.9) • `MX` - Mexico (Since 2.9) • `NZ` - New Zealand (Since 2.9) • `PE` - Peru (Since 2.9) • `PL` - Poland (Since 4.3) • `QA` - Qatar (Since 5.2) • `SA` - Saudi Arabia (Since 5.2) • `SG` - Singapore • `ZA` - South Africa (Since 2.9) • `ES` - Spain (Since 4.3) • `UA` - Ukraine (Since 5.2) • `AE` - United Arab Emirates (Since 2.9) • `GB` - United Kingdom (Since 4.3) • `US` - United States The value of the `country` attribute must be compatible with the value of the `locale` attribute. Since 4.6
`setCity(String value)`	Set the name of the city that is associated with the address. **Format:** Alphanumeric; maximum 100 characters Since 3.0

Getter and setter methods	Field Description
`setLine1(String value)`	First line of the address. **Format:** Alphanumeric; maximum 140 characters Since 2.0
`setLine2(String value)`	Second line of the address, if it exists. **Format:** Alphanumeric; maximum 140 characters Since 2.0
`setLine3(String value)`	Third line of the address, if it exists. It is used only to hold the name of the suburb for New Zealand addresses. **Format:** Alphanumeric; maximum 140 characters Since 2.7
`setPersonName( String value)`	Set the addressee's complete name. **Format:** Alphanumeric; maximum 256 characters Since 2.0
`setPhone(String value)`	Set the phone number that is associated with the billing address. Only available to contractually-enabled merchants. **Format:** Numeric or hyphens, parentheses, period, or plus sign, valid for the country; maximum 15 characters Since 3.0
`setPostalCode( String value)`	Set the postal code that is associated with the billing address. Only available to contractually-enabled merchants. **Format:** Depends on `countrycode`, maximum 7 characters: • `US` must be 5 digits • `CA` must be 6 characters separated by a space or a hyphen, e.g. `A0A 0A0` • `AU` must be 4 digits Since 3.0

Getter and setter methods	Field Description
`setStateProvinceCode( String value)`	Set the name of the state or province code that is associated with the billing address. Only available to contractually-enabled merchants. **Format:** Alphanumeric; maximum 100 characters Since 3.0
`setTwoCharacterCountryCode( String twoCharacterCountryCode)`	Set a two-character country code, such as US and other country codes. **Format:** One of the following ISO-3166-1 alpha-2 standard codes: • `AR` - Argentina (Since 2.7) • `AU` - Australia • `BR` - Brazil (Since 2.7) • `CA` - Canada • `CN` - China (Since 2.9) • `CL` - Chile (Since 2.9) • `CO` - Colombia (Since 2.9) • `FR` - France (Since 4.3) • `HK` - Hong Kong (Since 2.9) • `IN` - India (Since 4.6) • `IE` - Ireland (Since 4.3) • `KW` - Kuwait (Since 5.2) • `MY` - Malaysia (Since 2.9) • `MX` - Mexico (Since 2.9) • `NZ` - New Zealand (Since 2.9) • `PE` - Peru (Since 2.9) • `PL` - Poland (Since 4.3) • `QA` - Qatar (Since 5.2) • `SA` - Saudi Arabia (Since 5.2) • `SG` - Singapore • `ZA` - South Africa (Since 2.9) • `ES` - Spain (Since 4.3) • `UA` - Ukraine (Since 5.2) • `AE` - United Arab Emirates (Since 2.9) • `GB` - United Kingdom (Since 4.3) • `US` - United States The value of the `country` attribute must be compatible with the value of the `locale` attribute. Since 4.6

Visa User Info

```
public final class VisaUserInfo
extends java.lang.Object
implements android.os.Parcelable
```

Getter and setter methods	Field Description
`getBillingAddress()`	Get the billing address that is associated with the payment instrument. **Format:** `Address`; see the *Visa Checkout Integration Guide*. Since 2.0
`String getEmailAddress()`	Get the E-mail address that is associated with the payment instrument. It is a valid E-mail address for the consumer who makes the payment. **Format:** Alphanumeric, valid E-mail address; maximum 256 characters
`String getFirstName()`	Get the consumer's given (first) name. **Format:** Alphabetic or the following characters: spaces, **'** (single quote), ` (back tick), ~ (tilde), **.** (period), and **-** (hyphen); maximum 24 characters Since 2.0
`String getLastName()`	Get the consumer's surname (last name). **Format:** Alphabetic or the following characters: spaces, **'** (single quote), ` (back tick), ~ (tilde), **.** (period), and **-** (hyphen); maximum 24 characters Since 2.0
`String getPhoneNumber()`	Get the phone number that is associated with the billing address. Only available to contractually-enabled merchants. **Format:** Numeric or hyphens, parentheses, period, or plus sign, valid for the country; maximum 15 characters Since 3.0
`String getShippingAddress()`	Get the shipping address. **Format:** `Address`; see the *Visa Checkout Integration Guide*. Since 2.0

`setBillingAddress(`
`  VisaUserAddress billingAddress)`
	Set the billing address that is associated with the payment instrument.
	
	Format: `Address`; see the *Visa Integration Guide*.
	
	Since 2.0

`setEmailAddress(`
`  String emailAddress)`
	Set the E-mail address that is associated with the payment instrument. It is a valid E-mail address for the consumer who makes the payment.
	
	Format: Alphanumeric, valid E-mail address; maximum 256 characters

Getter and setter methods	Field Description
`setFirstName( String firstName)`	Set the consumer's given (first) name. **Format:** Alphabetic or the following characters: spaces, `'` (single quote), `` ` `` (back tick), ~ (tilde), `.` (period), and `-` (hyphen); maximum 24 characters Since 2.0
`setLastName( String lastName)`	Set the consumer's surname (last name). **Format:** Alphabetic or the following characters: spaces, `'` (single quote), `` ` `` (back tick), ~ (tilde), `.` (period), and `-` (hyphen); maximum 24 characters Since 2.0
`setPhoneNumber( String phoneNumber)`	Set the phone number that is associated with the billing address. Only available to contractually-enabled merchants. **Format:** Numeric or hyphens, parentheses, period, or plus sign, valid for the country; maximum 15 characters Since 3.0
`setShippingAddress( VisaUserAddress shippingAddress)`	Set the shipping address. **Format:** `Address`; see the *Visa Checkout Integration Guide*. Since 2.0

SDK Upgrade and Discontinuance Information A

Unless otherwise noted, all features from previous versions are included in the upgrade version.

Installing 5.5 for the First Time

- Support for 5.5 is expected to discontinue December 31, 2018.
- Follow the Quick Start chapter for easy integration
- Create separate mobile and web merchant profiles

Upgrading from 5.5 to 5.5.1

- Remove data-binding dependency

Upgrading from 5.2 to 5.5

- Support for 5.2 is expected to discontinue on September 30, 2018
- Follow the Quick Start chapter for easy integration
- Create separate mobile and web merchant profiles
- Include Google Play Services and Google Play Services Ads libraries

Upgrading from 4.6 to 5.5

- Support for 4.6 is expected to discontinue on April 30, 2018
- Follow the Quick Start chapter for easy integration
- Create separate mobile and web merchant profiles
- Remove Event bus dependency for VisaCheckout
- Replace VisaPaymentButton and VisaExpressCheckoutButton with VisaCheckoutButton
- Update Proguard rules per previous instructions

Upgrading from 4.3 to 5.5

- Support for 4.3 is expected to discontinue on September 30, 2017
- Follow the Quick Start chapter for easy integration

- Create separate mobile and web merchant profiles
- Remove Event bus dependency for VisaCheckout
- Replace VisaPaymentButton and VisaExpressCheckoutButton with VisaCheckoutButton
- Update Proguard rules per previous instructions

Upgrading from 4.0 to 5.5

- Support for 4.0 has been discontinued as of May 31, 2017

What's New in Prior Releases B

What's New in Version 5.2

Optimized Design

New welcome screen removes friction points and drives increased conversion and reduces abandonment (based on usability studies).

Global Market Expansion

The Mobile SDKs now support Visa Checkout merchant transactions in Central and Eastern Europe, Middle East and Africa (CEMEA). As part of this launch:

Note: *Merchant support tools are in English only.*

- Kuwait (English), Qatar (English), Saudi Arabia (English), Ukraine (English, Ukrainian (UI only))
- Address and phone formats have been updated to support the formats of these countries
- Only Roman character input (extended ASCII) is supported; Arabic text display is not supported
- Consumers, merchants, partners, and issuers from the Crimea region are not able to enroll in Visa Checkout because of international sanctions.

What's New in Version 4.6

This manual is intended for experienced programmers writing Java code for Android apps.

New App Framework

The application framework makes it easier to develop apps with the SDK. You should review the Quick Start information before starting.

Streamlined, Express Checkout Experience

Two touch, express checkout for returning users.

- Touch for fingerprint authentication (if supported and activated on device)

- Touch to confirm payment information.

New ATOM Design and Framework

Scalable, modern design provides consistent UI, navigation and actions using common components and built for global market expansion

Optimized Design

Mini-layer removes friction points and preserves merchant experience which will reduce abandonment and increase conversion (based on numerous usability studies)

Global Market Expansion

The Mobile SDKs now support Visa Checkout merchant transactions in India. This includes updated address and phone number localizations. In addition, the Mexico address form has been revised to provide more accurate shipping and billing information. Note: merchant tools are in English only.

Annual Policy Update

New privacy, terms and conditions, and other policy screens are available in support of Visa's annual policy update. This will insure our users are protected by the most recent legal information available.

Features Removed

- Card scanning during the add card process – removed due to library size, failure rate, permissions required, and non-use.
- Import contacts during add address process – removed due to failure rate, permissions required, and non-use.
- Welcome screen – removed due to image rendering issues, potential negative impact to account creation, and non-use.
- Custom headers - removed due to image rendering issues, potential negative impact to account creation, and non-use.

What's New in Version 4.3

- Visa Checkout billing and shipping are supported for the following additional countries:
  - France
  - Ireland
  - Poland
  - Spain
  - United Kingdom
- The **Pay** button in the lightbox can be used with any currency supported by Visa Checkout.
- If you are a merchant in United Kingdom, France, Spain, Ireland, or Poland, you must allow issuers to authenticate consumers from these countries on the first use of a card by the consumer in Visa Checkout. Contact your Visa Checkout representative to enable and configure Verified by Visa for Visa cards and SecureCode for Mastercard cards to perform authentication when a consumer's card is first used. If you fail to

enable Verified by Visa and SecureCard for this purpose, you are responsible for allowing the issuer to authenticate the consumer on a card's first use. If you accept American Express cards from these countries, you must allow the authentication of the consumer outside of Visa Checkout on every transaction; American Express Safekey support is not currently provided by Visa Checkout. For more information, see the *Visa Checkout Integration Guide*.

- Compliance with European Union privacy requirements is provided through the display of a cookie/privacy banner with a link to the privacy policy and the option to allow consumers to opt-out of non-essential cookies and consumer emails. The privacy policy and ability to opt-out of non-essential cookies is available to all Visa Checkout users.
- During enrollment of a Polish consumer, an additional consent check box appears to support mandated international sanctions verification.

What's New in Version 4.0

- Two new payment types are supported for Brazil: `ELECTRON` and `ELO`.
- Portuguese language strings are supported for Brazil.
- The first line of a Brazilian address is subdivided into 3 fields within the consumer information payload: street number (`streetNumber`), street name (`streetName`), and additional location information (`additionalLocation`); the complete first line (`line1`) remains available.
- Address auto-complete is also supported.
- A library delegate method, `GAISharedInstance`, is available for use with Google Analytics.

What's New in Version 3.7

Version 3.7 of the SDK provides support for the following new features:

- Apple iPad Tablet Support

  Consumers can now use Visa Checkout in merchant-provided iPad apps.
- Android Tablet Support

  Consumers can now use Visa Checkout in merchant-provided Android tablet apps.
- Expanded Selection of Buttons and Marks

  A number of new buttons and marks have been provided, which allow merchants to utilize a wider variety of new designs when integrating the Visa Checkout SDK into their apps. See the Branding Requirements chapter of the *Visa Checkout Getting Started Guide* for more information.
- Enhanced Card on File Support

  Merchants have the ability to provide Visa Checkout consumers the option of placing their credit cards on file with the merchant to use as payment on the merchant's app. Enhancements were made to the call-to-action buttons to help consumers understand the flow.
- User Interface and Design Enhancements

  Improvements were made to the carousel motion for credit cards and addresses. Changes were made in the location of the camera button to provide better visibility for the consumer. Also, input fields are highlighted in blue to provide users with visibility for more efficient data input.

- Singapore and Malaysia support

  Merchants will now have enhancements in form fields, which will help process transactions in Singapore and Malaysia. For Singapore, city defaults to Singapore. For Malaysia, a value for city is required.

- Messaging Improvements

  Updates to provide more consumer-friendly messaging for various scenarios.

In addition, this version contains minor bug fixes and code optimizations.

What's New in Version 3.5

Version 3.5 of the SDK provides support for the following new features:

- Enhanced Support for Apple Touch ID

  Customers can access their Visa Checkout account using Apple's Touch ID, which uses touch authentication.

- Support for Android Touch Authentication

  Customers can access their Visa Checkout account using touch authentication.

- Market Expansion – Latin America

  Visa Checkout SDK now supports Spanish for countries in the Latin America market, including: Mexico, Argentina, Chile, Colombia, and Peru, and Portuguese in Brazil.

- Language Support – Chinese

  Visa Checkout SDK now has Visa Checkout UI language support for displaying Chinese (Simplified, Traditional). Consumers must enter all account information, including names and addresses, in English.

- Messaging Improvements

  Updates to provide more consumer-friendly messaging for various scenarios.

In addition, this version contains minor bug fixes and code optimizations.

Document Revision History C

- Version 2.2, July 8, 2014
- Version 2.3, August 5, 2014
- Version 2.4, September 17, 2014
- Version 2.6, December 8, 2014
- Version 2.7, February 10, 2015
- Version 2.9, May 5, 2015
- Version 3.3, September 9, 2015
- Version 3.5, December 11, 2015
- Version 3.7, March 27, 2016
- Version 4.0, May 27, 2016
- Version 4.3, September 13, 2016
- Version 4.6, February 9, 2017
- Version 5.2, July 21, 2017
- Version 5.5, September 27, 2017