CardConnect Consumer SDK and Android Pay

**Steps to use Android Pay Consumer SDK APIs**

In order to be able to integrate with Android Pay, the CardConnect SDK Consumer version provides two services that allow the implementer application to process payment information coming from Android Pay through the CardConnect processor. For more information consult <https://developers.google.com/android-pay/tutorial>:

1. Get public key for merchant:

CardConnect will provide a public key for the merchant in order to be able to request and further decrypt Android Pay encrypted information on CardConnect Server:

|  |
| --- |
| CCConsumer.*getInstance*().getApi().getPublicKey(**new** CCConsumerApiCallbacks() {  @Override  **public void** onApiResponse(@NonNull CCConsumerApiResponse ccConsumerApiResponse) {  *//Hide any visual indicator for asynchronous process*  **if** (ccConsumerApiResponse.getCCConsumerError() != **null**) {  *//Not able to fetch Public Key should display an error indicator here*  } **else if** (ccConsumerApiResponse **instanceof** CCConsumerApiPublicKeyResponse) {   *//Prepare and initialize SupportWalletFragment*  }  } }); |

1. Generate a CardConnect payment token based on the Android Pay encrypted payload:

When requesting the FullWallet object from Android Pay with the final payment information, the implementer should proceed to call CardConnect Card Secure API to generate a payment token:

|  |
| --- |
| *//Get Android Pay Payment Token*  PaymentMethodToken token = fullWallet.getPaymentMethodToken();  *//Create an Android Pay token request for CardConnect* CCConsumerAndroidPayGetTokenRequest request =  **new** GsonBuilder().create().fromJson(token.getToken(), CCConsumerAndroidPayGetTokenRequest.**class**);  CCConsumer.*getInstance*().getApi().generateTokenFromAndroidPayload(request, **new** CCConsumerApiCallbacks() {  @Override  **public void** onApiResponse(@NonNull **final** CCConsumerApiResponse ccConsumerApiResponse) {  *//Hide any visual indicator for asynchronous process*  **if** (ccConsumerApiResponse.getCCConsumerError() != **null**) {  *//Not able to generate token from CardConnect should display an error indicator here*  } **else if** (ccConsumerApiResponse **instanceof** CCConsumerApiAndroidPayTokenResponse) {  *//Proceed and create a CardConnect Authorization request with the token response*  }  } }); |

**Steps to integrate with Android Pay through the Consumer Integrated UI**

1. Include Google play library:

The Android Pay API is part of the Google Play Services, so in order for the Consumer SDK to be able to use Android Pay APIs you need to import the Google Play Services Library.

consumerCompile **'com.google.android.gms:play-services-wallet:9.4.0'**

Note: This dependency is required when calling Consumer integrated UI. In case the implementer application wants to display payment methods without Android Pay.

1. Enable Android Pay in Consumer Application:

To enable Android Pay in your app, you need to add the following Android Pay meta-data to the <application> tag of your AndroidManifest.xml

<**meta-data  
 android:name="com.google.android.gms.wallet.api.enabled"  
 android:value="true"** />

1. Display Buy with Android Pay Button payment option:

In order for the Consumer SDK integrated UI to show the Buy with Android Pay button as a valid payment option, the Android Pay Application needs to be installed on the User’s device and at least needs to have payment card set.

1. Pass Total to Consumer SDK Integrated UI:

When launching the Integrated UI, pass an extra parameter reflecting the total in case the user wants to pay with Android Pay. Total amount must follow “0.00” format to be a valid total used.

|  |
| --- |
| ApiBridgeImpl apiBridgeImpl = **new** ApiBridgeImpl(); Intent intent = **new** Intent(**this**, PaymentAccountsActivity.**class**); intent.putExtra(PaymentAccountsActivity.***API\_BRIDGE\_IMPL\_KEY***, apiBridgeImpl);  intent.putExtra(CCConsumerAndroidPayActivity.***ANDROID\_PAY\_TOTAL\_AMOUNT\_KEY***, **"5.00"**);  startActivityForResult(intent, PaymentAccountsActivity.***PAYMENT\_ACTIVITY\_REQUEST\_CODE***); |

1. Enable Android Pay on Consumer SDK:

By default, Android Pay is disabled in the Consumer SDK because it behaves as a unified bundle with several features either for Consumer User or Merchant User. In order to enable the Android Pay UI, the implementer should use the global configuration class defined:

|  |
| --- |
| CCConsumerAndroidPayConfiguration.*getInstance*().setAndroidPayUiEnabled  (**true**); |

1. Declare Confirmation Activity in order for the implementer to be able to configure the UI and the application to have access to it.

|  |
| --- |
| <**activity  android:name="com.cardconnect.consumersdk.androidpay.CConsumerConfirmationActivity"  android:theme="@style/ConsumerAppImplementer.Translucent"** /> |

1. Received Android Pay Account Information:

After confirming Android Pay selected method on Confirmation Activity. An CCConsumerApiAndroidPayTokenResponse object response with payment and other additional user information will be returned. Use this token to proceed with the authorization and the rest of the information to display a receipt or final confirmation page.

|  |
| --- |
| @Override **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  **super**.onActivityResult(requestCode, resultCode, data);   *//Get Account selected by integrated UI flow* **if** (requestCode == PaymentAccountsActivity.***PAYMENT\_ACTIVITY\_REQUEST\_CODE*** && resultCode == ***RESULT\_OK***) {   *//Check for Android Pay selected Payment (Only available on Consumer facing Applications)* **if** (data.hasExtra(PaymentAccountsActivity.***ANDROID\_PAY\_TOKEN\_RESPONSE\_KEY***)) {  CCConsumerApiAndroidPayTokenResponse response =  data.getParcelableExtra(PaymentAccountsActivity.***ANDROID\_PAY\_TOKEN\_RESPONSE\_KEY***);  Log.*d*(***TAG***,response.toString());  }  } } |

1. Deploy:

Follow this link <https://developers.google.com/android-pay/deployment> to proceed with production Android Pay integration.

Note: To set test or production environment call

|  |
| --- |
| CCConsumerAndroidPayConfiguration.*getInstance*().setTestMode(**false**); |

1. Configuration:

Consumer Sdk provides a Singleton class to configure all Android Pay related information. Consult the Javadoc reference for a class called CCConsumerAndroidPayConfiguration.

In addition to the configuration class, if the implementer wants to override some of the UI elements contained on Confirmation Activity the implementer should create a theme extending AppConsumerTheme.Translucent theme from Consumer SDK and apply it to the Confirmation Activity. Here is an example:

|  |
| --- |
| <**style name="ConsumerAppImplementer.Translucent"parent="AppConsumerTheme.Translucent"**>  *<!--//****TODO Override Attributes****-->*    </**style**> |

Note: For more specific information regarding Theme configuration attributes check theme configuration resources.