|  |  |
| --- | --- |
| **summary:** | HMS Core Wallet Kit |
| **url:** | HMSWalletKit |
| **categories:** | HMS |
| **status:** | Published |
| **feedback link:** | [https://developer.huawei.com/consumer/en/support/feedback](https://developer.huawei.com/consumer/cn/support/feedback) |
| **authors:** | HUAWEI |

# HMS Core Wallet Kit

## Introduction

Duration: 1:00

### Overview

Duration: 1 minute

HUAWEI Wallet Kit allows users to claim passes of merchants, including loyalty cards, gift cards, offers, boarding passes, event tickets, and transit passes, and add them to HUAWEI Wallet. You can provide a button or link in your app for users to add passes to their HUAWEI Wallet and view and manage them.

You need to perform the following operations:

* Register an account on HUAWEI Developer. Create and configure an app and enable the HUAWEI Wallet Kit API in [AppGallery Connect](https://developer.huawei.com/consumer/en/service/josp/agc/index.html).
* Select and register HUAWEI Wallet Kit in the app, and configure a merchant service account and callback address. Use a tool to generate a public key and synchronize the key to the Huawei server.
* Create a project with Android Studio IDE and configure connection to the HMS Core Wallet SDK.
* Call the HUAWEI Wallet Kit API to push a template of merchant information to the Huawei server.
* Call and debug the HMS Core Wallet SDK. For details, please refer to "Integrating the HMS Core Wallet SDK" in the HUAWEI Wallet Kit development guide.

### What You Will Create

In this codelab, you will use the created demo project to:

* Call the HMS Core Wallet SDK to construct objects so that passes can be added to HUAWEI Wallet.

### What You Will Learn

* Create an app in [AppGallery Connect](https://developer.huawei.com/consumer/en/service/josp/agc/index.html).
* Enable the HUAWEI Wallet Kit API.
* Import the HMS Core Wallet SDK.
* Call the HMS Core Wallet SDK to add passes to HUAWEI Wallet.

## What You Will Need

Duration: 1:00

### Hardware Requirements

* A computer (desktop or laptop) that runs Windows 10
* A Huawei mobile phone with Huawei Mobile Services 3.0.0.300 or later, which will be used for debugging and running the demo project

|  |
| --- |
| **Note:** Please prepare the preceding hardware environment and relevant devices in advance. |

### Software Requirements

* Android Studio 3.X
* Java JDK (1.8 or later)
* SDK Platform (19 or later)
* Gradle (4.6 or later)

|  |
| --- |
| **Note:** Please prepare the preceding software environment in advance. |

## Preparing for the Integration

Duration: 20:00

HUAWEI HMS Core integration requires the following preparations

* Creating an AGC Application.
* Creating an Android Studio Project.
* Generating a signature certificate.
* Generating a signature certificate fingerprint.
* Configuring the signature certificate fingerprint.
* Adding the application package name and save the configuration file.
* Configure the Maven address and AGC gradle plug-in.
* Configure the signature file in Android Studio.

For details, see the [HUAWEI HMS Core Integration Preparation](https://developer.huawei.com/consumer/en/codelab/HMSPreparation/index.html).

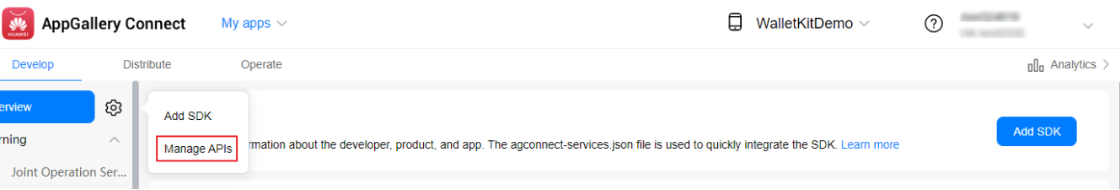
|  |
| --- |
| **Note:** You need to register as a developer to complete the operations above. |

## Enabling HUAWEI Wallet Kit

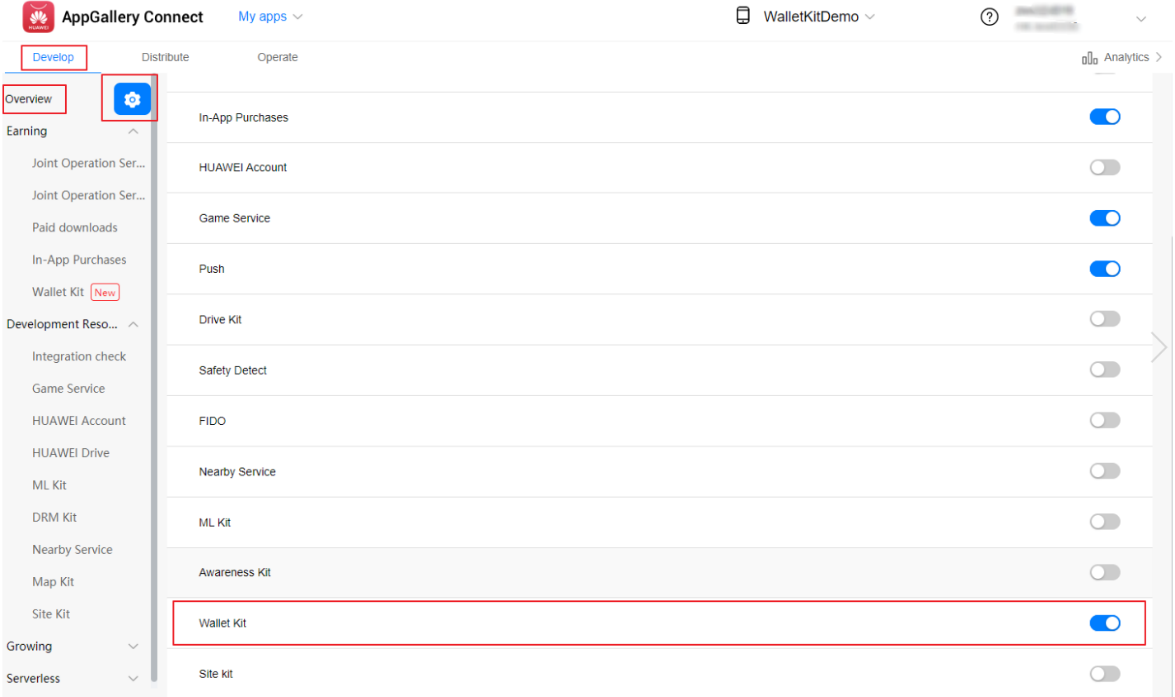
Duration: 5:00

### Enabling the HUAWEI Wallet Kit API

* Sign in to AppGallery Connect and go to **My apps > Develop > Overview > Manage APIs**.



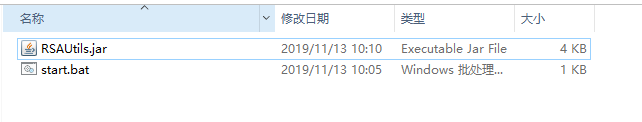
* Enable the HUAWEI Wallet Kit API.



You have now successfully enabled the HUAWEI Wallet Kit API for your app, for example, WalletKitDemo shown in the preceding figure.

### Generating a Public-Private Key Pair

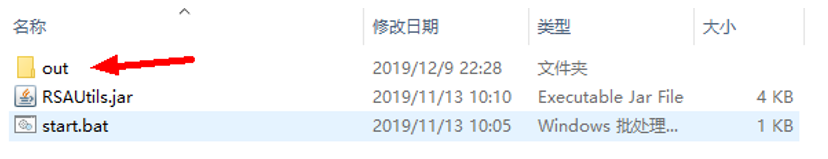
* Download the RSAUtils.rar package, decompress it, and open the **RSAUtils** folder.



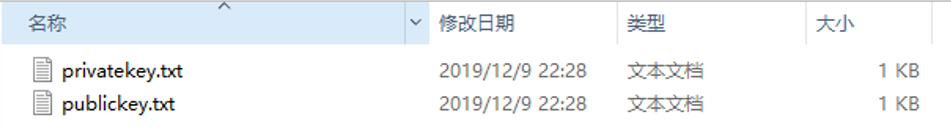
* Double-click the **start.bat** file.



* After the file is successfully executed, the path to the generated public and private keys is displayed. By default, the keys are in the **out** folder in the current path.



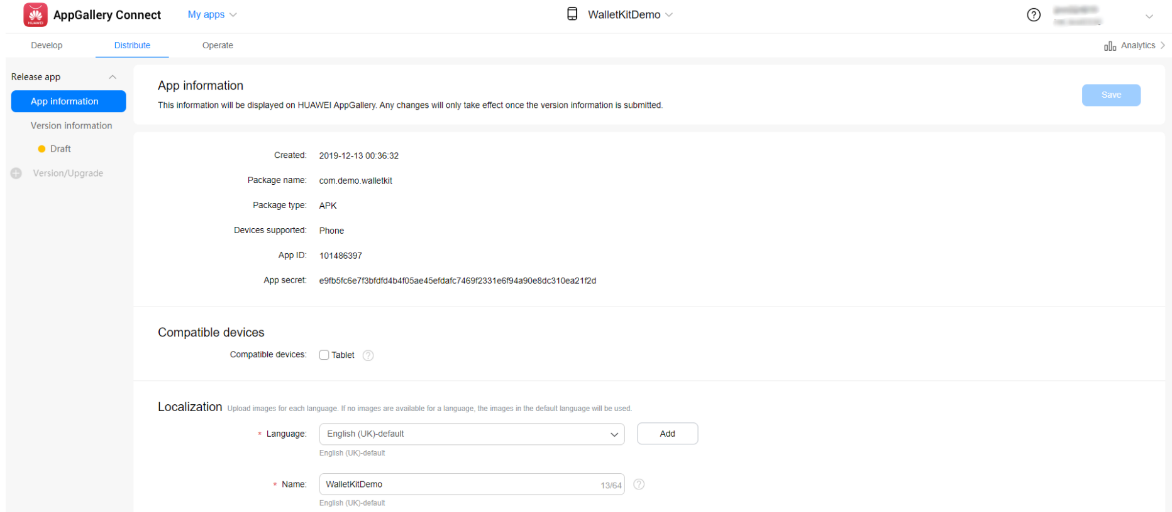
* The following figure displays files in the **out** folder.



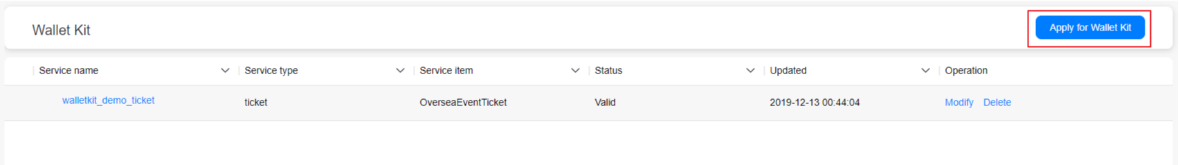
|  |
| --- |
| **Note:**  **publickey.txt:** public key file, which is used for registering HUAWEI Wallet Kit. The public key is verified by the Huawei server when a user attempts to add a pass to HUAWEI Wallet.  **privateKey.txt**: private key file. The private key is used to sign the JSON string of the object. Please store the file properly. |

### Registering HUAWEI Wallet Kit

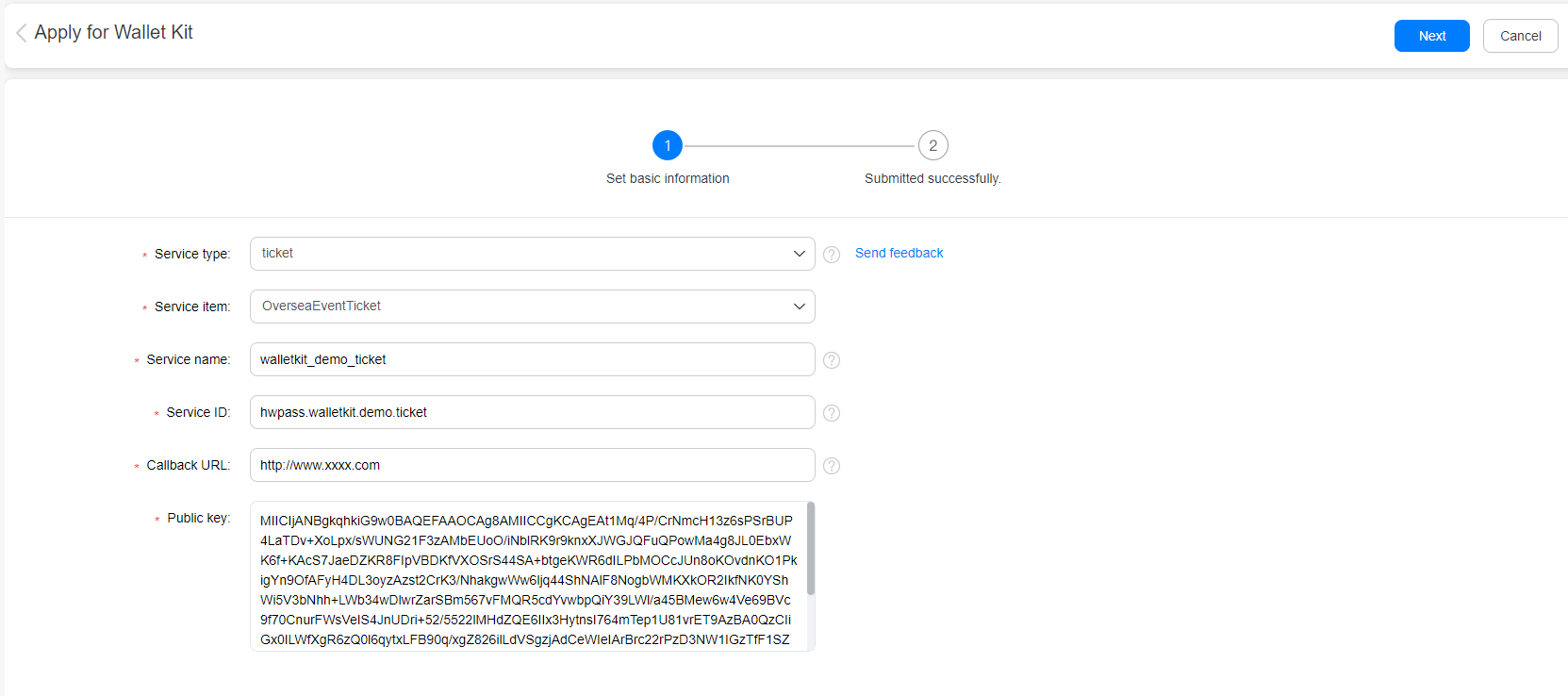
* Sign in to AppGallery Connect and go to **My apps**.



* Click the **Develop** tab and go to **Earning** > **WalletKit** from the navigation panel on the left. Click **Apply for Wallet Kit** in the upper right corner on the page displayed.



* Configure basic service information, such as the service type, service ID, and callback URL, enter the public key, and click **Next**.

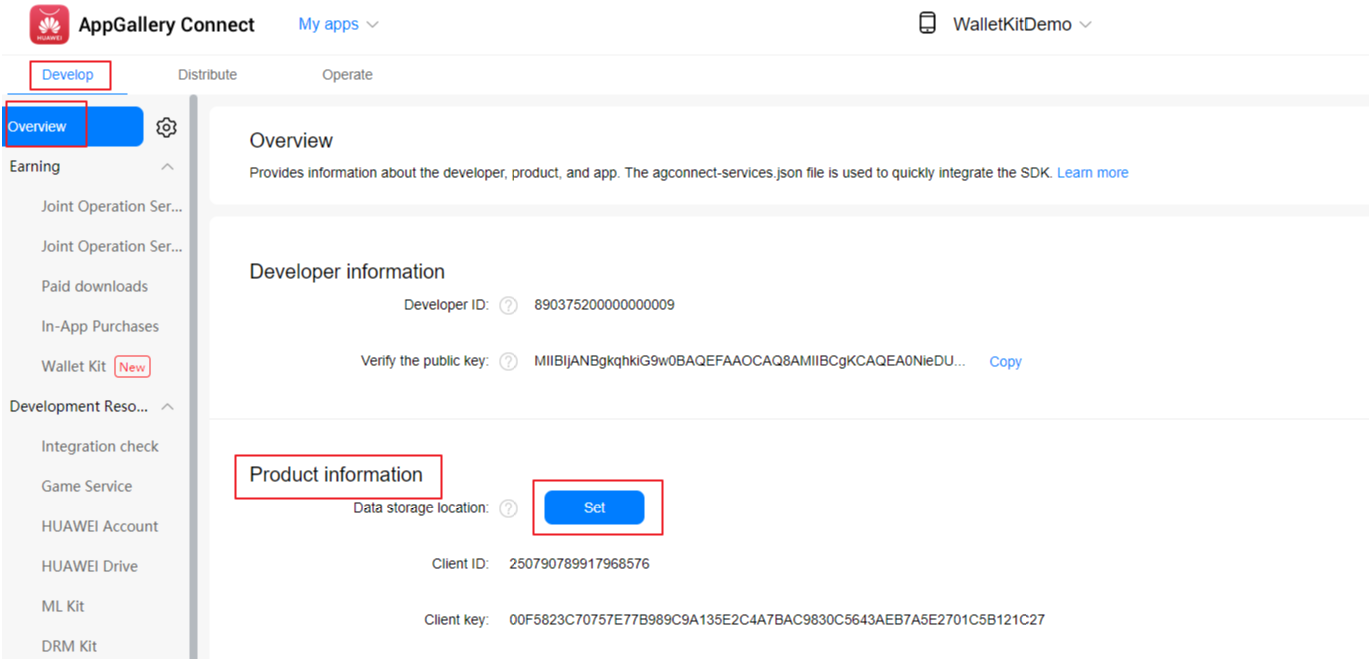


## Integrating the HMS Core Wallet SDK

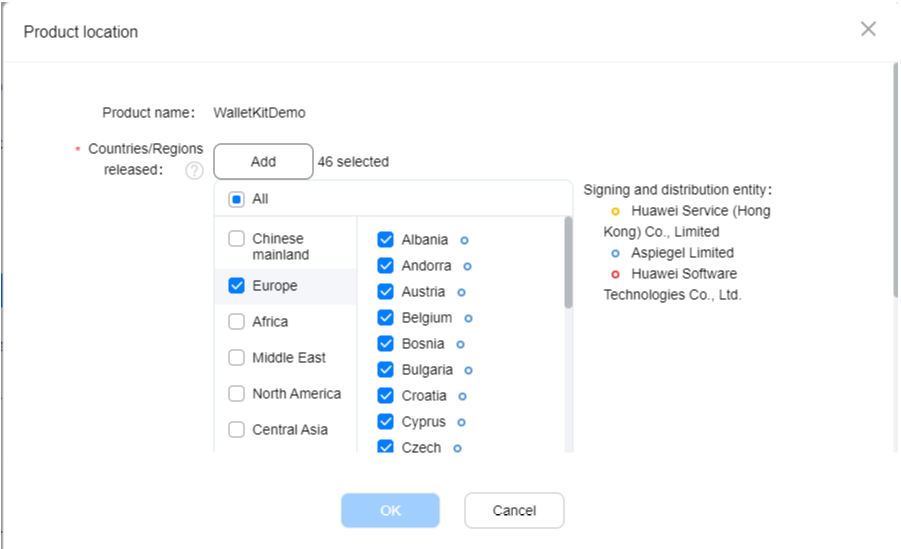
Duration: 5:00

### Obtaining the Configuration File

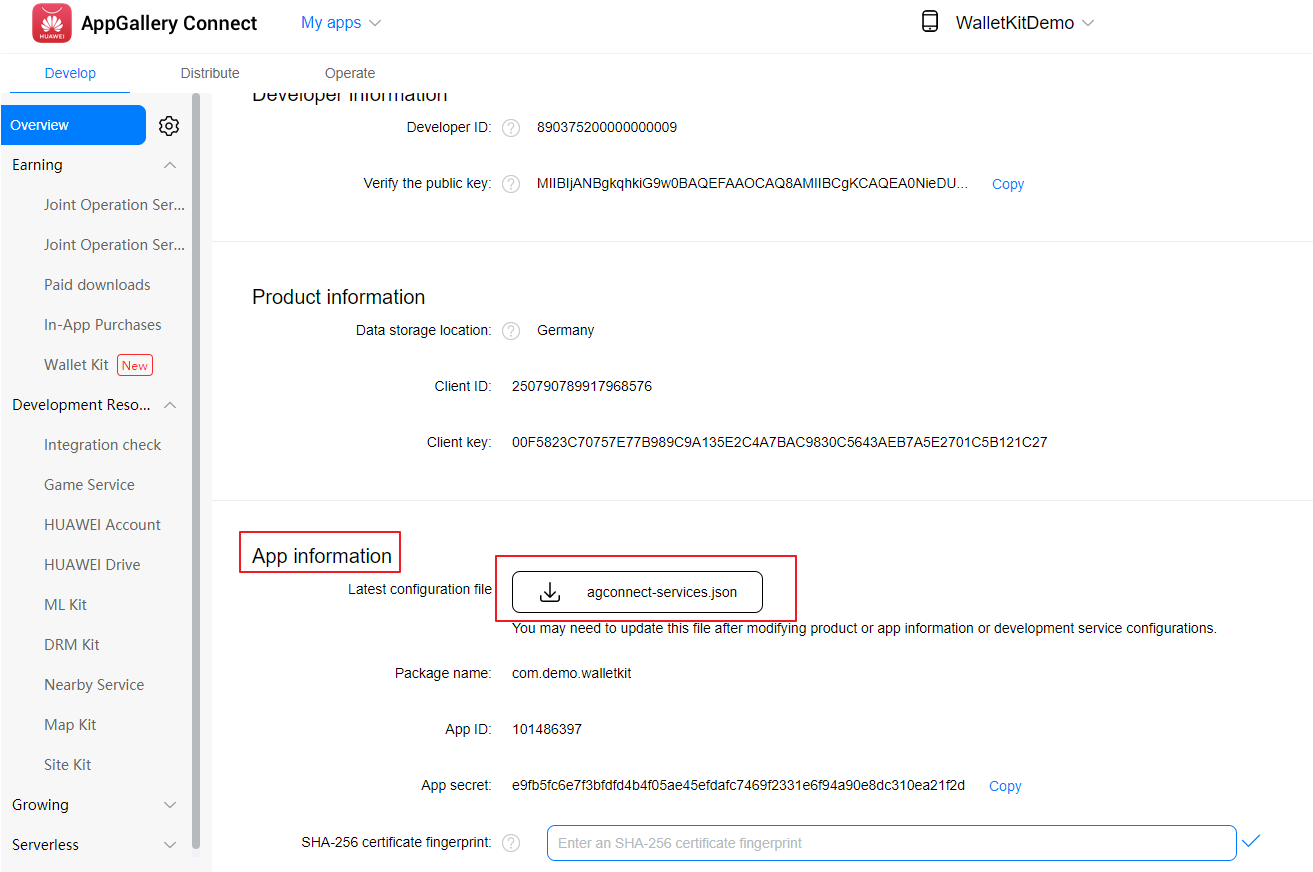
* Sign in to AppGallery Connect, go to **My apps** > **Develop** > **Overview**, select the app created in the upper right corner, and click **Set** next to **Data storage location**.



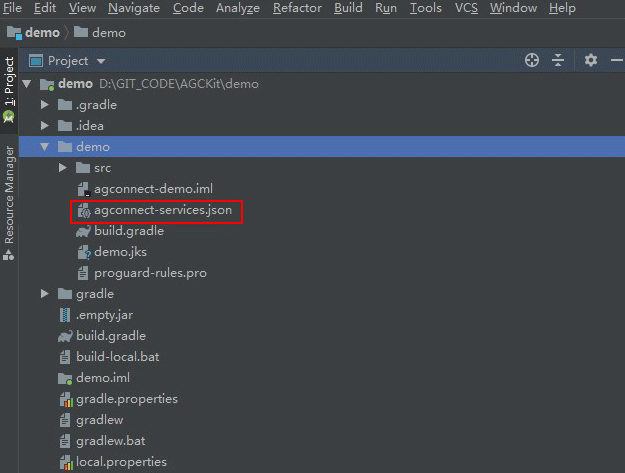
* Select countries and regions as required and click **OK**.



* In the **App information** section, download the **agconnect-services.json** file.

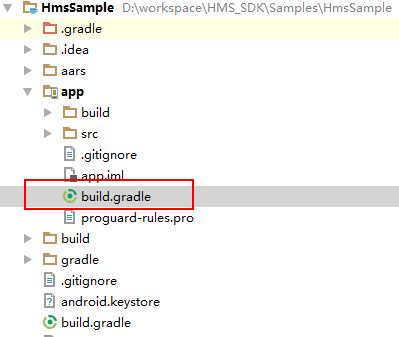


* Save the **agconnect-services.json** file to the root directory of the app in the Android Studio demo project.



### Configuring Dependency for the HMS Core Wallet SDK

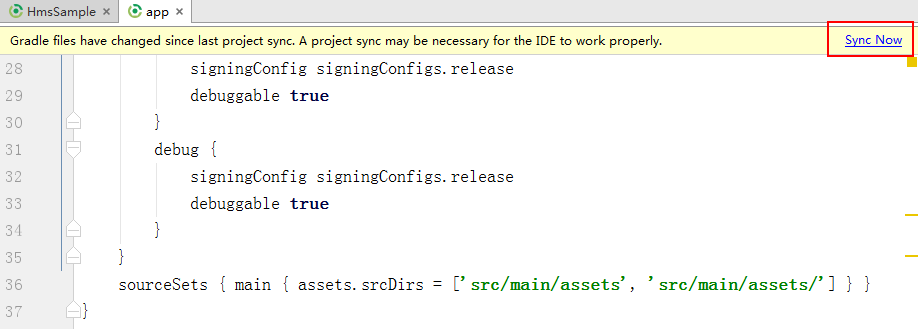
* Open the **build.gradle** file in the **app** directory.



* Add the following build dependencies in **dependencies** and replace {version} with the current version number of the HMS Core Wallet SDK, which is 2.0.0.300.

|  |
| --- |
| dependencies {  implementation 'com.huawei.hms:wallet:2.0.0.300' } |

* Click **Sync Now** to synchronize the project.



### Configuring Obfuscation Scripts

|  |
| --- |
| **Note:** If you need to use the code obfuscation function, please configure the obfuscation configuration file properly to avoid affecting the HMS Core Wallet SDK, which could lead to function errors. |

* Open the obfuscation script file **proguard-rules.pro** of your Android Studio project and add obfuscation configurations to the file.

|  |
| --- |
| -ignorewarning  -keepattributes \*Annotation\*  -keepattributes Exceptions  -keepattributes InnerClasses  -keepattributes Signature  -keepattributes SourceFile,LineNumberTable  -keep class com.hianalytics.android.\*\*{\*;}  -keep class com.huawei.updatesdk.\*\*{\*;}  -keep class com.huawei.hms.\*\*{\*;} |

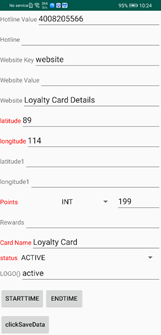
* If you have used AndResGuard, add it to the whitelist in the obfuscation script file.

|  |
| --- |
| "R.string.hms\*",  "R.string.connect\_server\_fail\_prompt\_toast",  "R.string.getting\_message\_fail\_prompt\_toast",  "R.string.no\_available\_network\_prompt\_toast",  "R.string.third\_app\_\*",  "R.string.upsdk\_\*",  "R.layout.hms\*",  "R.layout.upsdk\_\*",  "R.drawable.upsdk\*",  "R.color.upsdk\*",  "R.dimen.upsdk\*",  "R.style.upsdk\*" |

## UI Design

Duration: 20:00

The HUAWEI Wallet Kit API is used to add passes to HUAWEI Wallet. You need to configure pass information when calling this API. For example, to add a loyalty card through this API, you need to configure the loyalty card number, loyalty points, and member name on the loyalty card. In this codelab, you can create a layout page in your Android Studio project and design the UI according to the following figure.



## Develop the Function of Adding Passes to HUAWEI Wallet

Duration: 30:00

|  |
| --- |
| **Note:** HUAWEI Wallet Kit sends a JWT to the Huawei server to generate a pass and bind the pass to a user. |

### Sample Code for Adding a Loyalty Card to HUAWEI Wallet

#### Displaying the Save to Huawei Wallet Button

Display the **Save to Huawei Wallet** button in an app. For details about button specifications, please refer to [Huawei Icon Specifications](https://developer.huawei.com/consumer/cn/service/hms/catalog/HwJointOperationAppV3.html?page=hmssdk_jointOper_devguide_icon_rules).

|  |
| --- |
| **Note:** The codelab sample implements the button using the Button widget. During practical development, please implement the button in compliance with the *Huawei Icon Specifications*. |

#### Constructing a Request and Sending It to the Huawei Server

* Construct a loyalty card object.

|  |
| --- |
| LoyaltyWalletObject.PassBuilder builder = LoyaltyWalletObject.newPassBuilder();  builder.setPassTypeIdentifier(typeId)  .setCardNumber(cardNumber)  .setPassBarcodeAlternateText(barcodeAlternateText)  .setPassBarcodeType(barcodeType)  .setPassBarcodeValue(barcodeValue)  .setPassStyleIdentifier(styleIdentifier)  .setOrganizationPassId(organizationPassId)  .setPassIssuerName(issuerName)  .setPoints(points)  .setPassProgramName(programName)  .setPassState(state)  .setPassTimeDuration(timeDuration); // Add the longitude and latitude.   List<LatLng> locations = new ArrayList<>();  locations.add(new LatLng(latitudSecond,longitudSecond));  builder.addPassLocations(locations); // Generate a loyalty card object.   LoyaltyWalletPass walletPassObj = builder.build(); |

* Convert the object to a JWT.

|  |
| --- |
| // Fixed public key provided by HUAWEI used to encrypt the sessionKey   String sessionKeyPublicKey = "";  // sessionkey is used to encrypt the payload data.   String sessionKey = RandomUtils.generateSecureRandomFactor(16);  // Encrypt JSON as payload of JWT   byte[] sessionKeyBytes = EncodeUtil.hex2Byte(sessionKey);  String objectJson = hwWalletObject.toJson();  String payLoadEncrypt = AESUtils.encryptAESCBC(objectJson.getBytes(EncodeUtil.UTF\_8), sessionKeyBytes);  // Encrypt sessionKey and set into header of JWT   Map<String, Object> headerMap = new HashMap<>();  try {  String sessionKeyPlaintext = RSA.encrypt(sessionKey.getBytes(), sessionKeyPublicKey, "RSA/ECB/OAEPwithSHA-256andMGF1Padding", "UTF-8");  headerMap.put("sessionKey", sessionKeyPlaintext);  } catch (Exception e) {  System.*out*.println(e.getMessage());  }   JwtBuilder builder = Jwts.builder().setHeader(headerMap).setPayload(payLoadEncrypt);  String userToken = builder.compact();  String content = userToken.substring(0, userToken.length() - 1);   String sign = "";  String charset = "utf-8";  try {  // Use the private key generated during service registration to sign the JWT and encode the signed JWT using Base64.   PKCS8EncodedKeySpec priPKCS8 = new PKCS8EncodedKeySpec(Base64.getDecoder().decode(privateKey));  KeyFactory keyf = KeyFactory.getInstance("RSA");  PrivateKey priKey = keyf.generatePrivate(priPKCS8);   java.security.Signature signatureObj = java.security.Signature.*getInstance*(SIGN\_ALGORITHMS256);  signatureObj.initSign(priKey);  signatureObj.update(content.getBytes(charset));   byte[] signed = signatureObj.sign();  sign = Base64.getEncoder().encodeToString(signed);  } catch (Exception ex) {  ...  }  String jwt\_content = userToken + sign; |

* Create **CreateWalletPassRequest**.

|  |
| --- |
| CreateWalletPassRequest request = CreateWalletPassRequest.newBuilder()  .setJwt(jwt\_content)  .build(); |

* Create a Client instance.

|  |
| --- |
| // Currently, the walletOptions parameter can only be set to null.  WalletPassClient walletObjectsClient = Wallet.getWalletPassClient(PassTestActivity.this, walletOptions); |

* Create a task.

|  |
| --- |
| Task<AutoResolvableForegroundIntentResult> task = walletObjectsClient.createWalletPass(request); |

* Execute the task.

|  |
| --- |
| // SAVE\_FLAG is a task execution flag used to receive the task execution result.  ResolveTaskHelper.excuteTask(task, XXXActivity.this, SAVE\_FLAG); |

* Receive the loyalty card adding result.

|  |
| --- |
| public void onActivityResult(int requestCode, int resultCode, Intent data) {  switch (requestCode) {  case SAVE\_TO\_ANDROID:  switch (resultCode) {  case Activity.RESULT\_OK:  Toast.makeText(this, "save success", Toast.LENGTH\_LONG).show();  break;  case Activity.RESULT\_CANCELED:  Toast.makeText(this, "cancel by user", Toast.LENGTH\_LONG).show();  break;  default:  if (data != null) {  int errorCode =  data.getIntExtra(  WalletCommonConstants.EXTRA\_ERROR\_CODE, -1);  Toast.makeText(this, "fail, [" + errorCode + "]:" + analyzeErrorCode(errorCode), Toast.LENGTH\_LONG).show();  } else {  Toast.makeText(this, "fail:data is null", Toast.LENGTH\_LONG).show();  }  break;  }  } } |

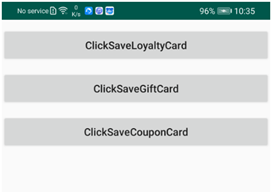
## Packing and Testing

Duration: 3:00

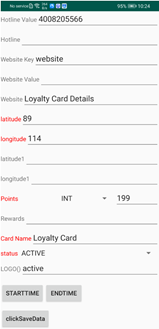
* Tap



to run the project you have created in Android Studio to generate an APK. Then install the APK on the test phone. The screen is shown in the following figure.



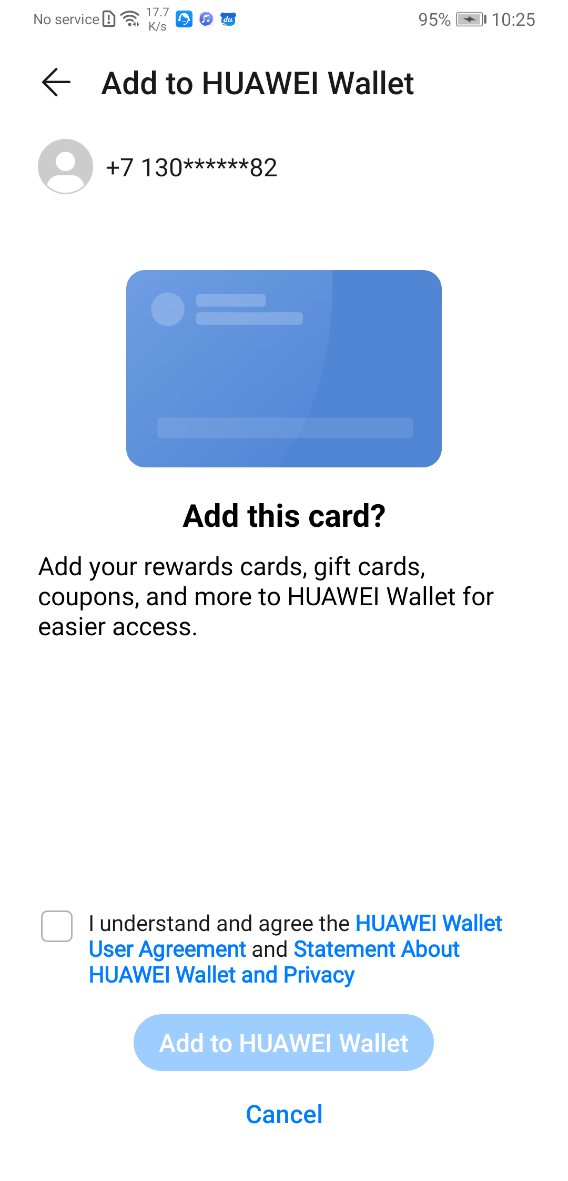
* Tap **ClickSaveLoyaltyCard**. The loyalty card configuration page is displayed.



* Enter mandatory information, such as the member name and merchant name. Tap **clickSaveData**. The following page is displayed.



* Tap **walletKitSDK add card**. The Wallet Kit adapter is started to display the confirmation page.



* If it is the first time that you add a loyalty card, the agreement checkbox is displayed. Select the checkbox and tap **Add to HUAWEI Wallet**. The Huawei server returns a result code. Result code 0 indicates that the loyalty card is successfully added.

## Congratulations

Duration: 0:00

Well done. You have successfully completed this codelab and learned:

* How to create an app in [AppGallery Connect](https://developer.huawei.com/consumer/en/service/josp/agc/index.html).
* How to enable the HUAWEI Wallet Kit API.
* How to import the HMS Core Wallet SDK and call the HUAWEI Wallet Kit API to add a pass to HUAWEI Wallet.

## Reference

For details about HUAWEI Wallet Kit APIs, please refer to

<https://developer.huawei.com/consumer/cn/service/hms/catalog/huaweiid_v3.html>.

Download the demo source code that is used in this codelab.

##### [Download](https://obs.cn-north-2.myhwclouds.com/hms-ds-wf/sdk/CodeLab_WalletKit_Demo.zip)