# AppCross In-App Billing Specification

Version 0.0.1

內容

[AppCross In-App Billing Specification 1](#_Toc385511275)

[Version 0.0.1 1](#_Toc385511276)

[1. 概述 3](#_Toc385511277)

[2. 內購商品(In-App Product) 3](#_Toc385511278)

[2.1 販賣方式 3](#_Toc385511279)

[2.2 商品類型 4](#_Toc385511280)

[3. AppCross Bookshelf 購物流程 5](#_Toc385511281)

[3.1 Checkout商家 5](#_Toc385511282)

[3.2 內購式商品(Purchasing Items)購買流程 9](#_Toc385511283)

[4. 建立內購商品 12](#_Toc385511284)

[5. 測試應用程式 14](#_Toc385511285)

[5.1 用測試帳號測試 14](#_Toc385511286)

[5.2 使用靜態回應的方式測試IAP購買機制 15](#_Toc385511287)

[6. AppCross Bookshelf In-App Billing程式實做說明 17](#_Toc385511288)

[6.1 準備要使用 In-app Billing 的應用程式 17](#_Toc385511289)

## 概述

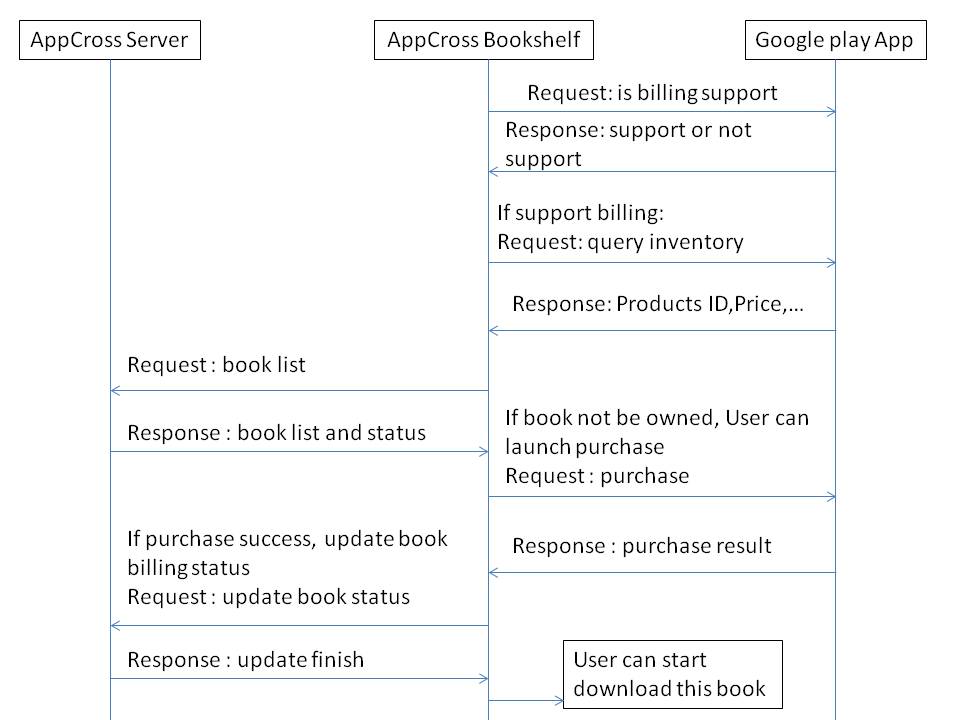
AppCross Bookshelf使用 Google In-App Billing API 透過裝置中內建的 Google Play app 來存取 In-app Billing 服務，Google Play app 會在AppCross Bookshelf 及 Google Play server 間傳送計費的請求及回應，AppCross Bookshelf 不會直接和 Google Play server 溝通，相反的，AppCross Bookshelf 是透過 IPC(Inter process communication) 傳送付費的請求給 Google Play app 然後接收從 Google Play app 傳回的結果。AppCross Bookshelf 不用管理任何網路連線的事。

## 內購商品(In-App Product)

### 販賣方式

在 AppCross Bookshelf 中的內購商品為數位書籍，透過 AppCross Bookshelf 賣給使用者，使用 In-app Billing 販賣數位內容，使用者不可以在購買後的"試用時間"內由使用者自己取消購買，當使用者付費購買內購商品時，是沒有退款的機制讓使用者自己取消購買(但可寫信給開發者，由開發者決定是否取消該次付款)。

Google Play 沒有提供任何形式的遞送內容的機制，AppCross Server必須自行傳遞數位商品給購買的使用者。AppCross Bookshelf內購式商品永遠明確的只和AppCross Bookshelf 產生關聯，也就是說，一個 App 不能購買由另一個 App 所發佈的內購式商品，即使是由同一位開發者所販賣的。



### 商品類型

AppCross Bookshelf可以在 Google Play Developer Console (開發者控制台)中定義要販售的商品。類型有：

* managed in-app products：託管式商品。
* unmanaged in-app products：非託管式商品。
* subscriptions：訂閱的內容。

託管的意思是指由 Google Play 掌握及追蹤在AppCross Bookshelf 中，單一使用者(Device Google Account)所購買的內購商品的所有權，所有的購買行為是受管理的，意即 Google Play 持續追蹤使用者的內購商品擁有權。使用者不能擁有多個內購商品的副本，在任何時間點只能有一個商品，非託管的意思就是由開發者自己做這件事。

## AppCross Bookshelf 購物流程

### Checkout商家

要使用In-App Billing購物機制，商家必須先成為 Google Checkout Merchant (Checkout商家)才能使用 In-app Billing 的服務，啟用Google Wallet成為Checkout商家。

Google Wallet啟用方式：

* 開發者先到Google Developer Console(<https://play.google.com/apps/publish/>)選擇設定頁，然後點選Google Wallet啟用。



* 設定商家資訊，設定完即可開始使用Google Wallet



* AppCross Bookshelf所有的In-App Billing紀錄都會記錄在Google Wallet裡



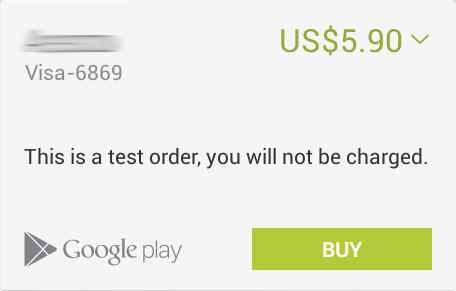
當使用者開始購買一項商品，AppCross Bookshelf 會送出一個指定商品的付款請求，接著 Google Play 會控管整個交易的結帳細節，包含請求及驗證付款格式並且處理金流。當結帳流程完成後，Google Play 會送出購買明細給AppCross Bookshelf，像是訂單號碼、訂購日期和時間以及付款金額，AppCross Bookshelf 完全不用控制整個交易流程，全部交由 Google Play 處理。

### 內購式商品(Purchasing Items)購買流程



1. AppCross Bookshelf 一啟動會執行In-App Billing Start setup送出 isBillingSupported 的請求，向 Google Play 確認目前這個 In-app Billing API 版本是否支援付款。
2. 當 AppCross Bookshelf 啟動後會檢查使用者已擁有哪些購買項目。AppCross Bookshelf會送出 query Inventory請求，查詢使用者購買過哪些內購式商品。如果請求成功，會回傳一個 Bundle，包含已購買商品的ID清單、個人的購買明細清單及為購買簽名的清單。
3. AppCross Bookshelf也可以查詢使用者有哪些商品可以購買， AppCross Bookshelf送出 query Inventory的請求，查詢請求中指定產品ID清單。查詢成功則回傳一個 Bundle，包含相關的產品細節，像是價格、標題、說明及購買類型。
4. AppCross Bookshelf啟動後，如果查詢到用戶有購買過哪些內購式商品時，會執行Consume Purchase 將此項內購式商品設定為可再購買，因為AppCross Bookshelf的內購式商品都是數位書籍的消費價格，書本是否可重複消費將由AppCross Server決定，多本數位書籍將共用AppCross Bookshelf的內購式商品。在Android In-App Billing Version 3 裡，內購式商品如果是被託管的，也就是說，內購商品的所有權是受 Google Play 管理，當使用成功購買一項商品時，購買記錄會被儲存在 Google Play，當一項商品被購買後，它就會成為"被擁有(owned)"狀態，當處於"被擁有"狀態時，不能再度在 Google Play 中重覆購買。必須對被使用者擁有的商品送出消除的請求，才能再度在 Google Play 中購買該項商品，消除該商品會讓狀態恢復到"未被擁有"的狀態。
5. AppCross Bookshelf執行購買請求時會送出 Launch Purchase，指定該項商品的ID來購買。開發者在"Google Developer Console"網站可以看到內購式商品的商品ID。

購買請求送出後Google Play 會回傳一個 Bundle 內含一個 PendingIntent 可以讓使用者開始付款結帳的界面。



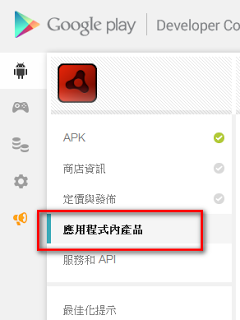
AppCross Bookshelf 會呼叫 startIntentSenderForResult 來啟動這個 PendingIntent。

當這個結帳流程結束(指使用者購買成功或取消購買)，Google Play 會回傳一個 Intent 到 AppCross Bookshelf 的 onActivityResult 方法，結果碼(result code)會表示這次的購買是成功的或被取消，在回傳的 Intent 中包含了購買商品的資訊，有一個由 Google Play 產生的 purchaseToken 字串，可用來識別這次的交易。

## 建立內購商品

在Google Developer Console建立可被購買的AppCross Bookshelf 數位商品，步驟如下：

1. 建立一個簽名(signed)過的AppCross Bookshelf APK檔，也就可以發佈到 Google Play 的應用程式。
2. 在Google Developer Console打開早先建立的AppCross Bookshelf應用程式，切換到 APK 頁面，然後上傳簽名過的 APK 檔。上傳到 Developer Console成功的話，就可以看到"應用程式內產品"可以新增項目。

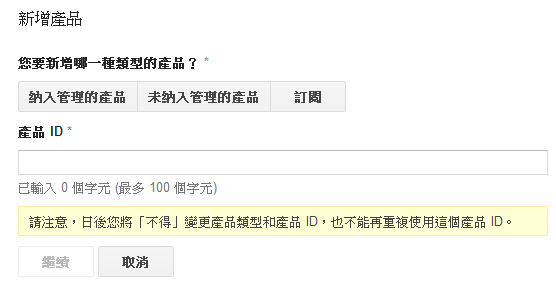


1. 切換到"應用程式內產品"頁面，點選"新增產品"，選擇"商品類型"，輸入產品 ID，這個 ID 就是 SKU，是個唯一值不能重覆。
2. 完成產品資訊的填寫並儲存後，將它啟用。現在AppCross Bookshelf 就可以購買該商品了。

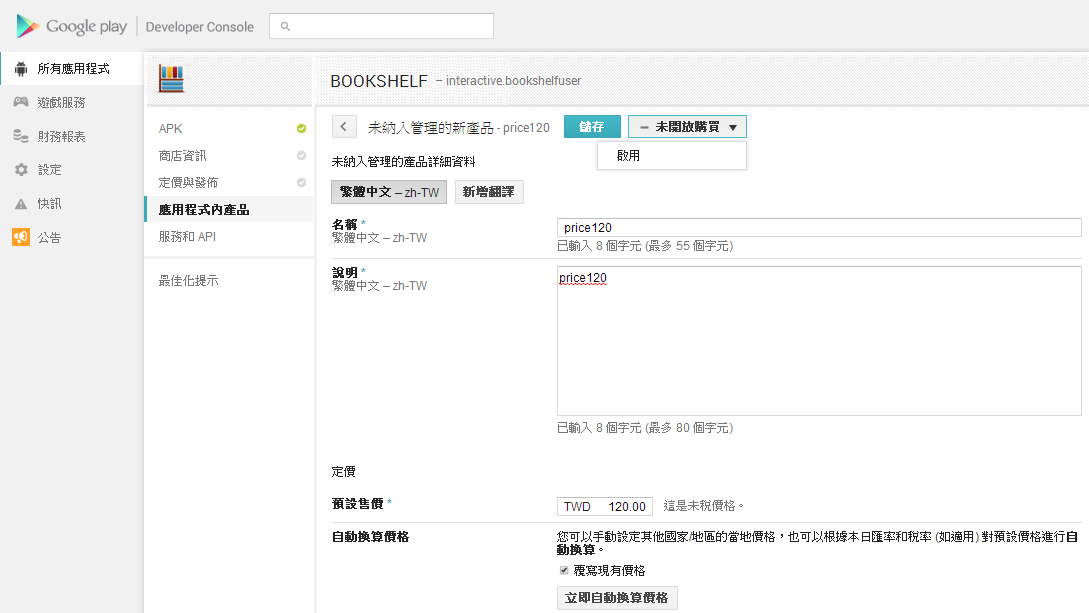
* 點選應用程式內產品



* 點選新增產品



* 輸入商品名稱、價格，然後按啟用即可



## 測試應用程式

### 用測試帳號測試

因為 Google 不允許使用開發者帳號直接購買內購式商品，所以開發者必須在原本的帳號之下建立一個測試帳號，如此才能購買已上傳但未發佈的 App。

在開發者控制台新增一個(或多個)測試帳號。進入開發者控制台->設定->帳戶詳細資料，找到"授權測試"項目，新增測試用 Gmail 帳戶。

透過 adb 的方式安裝已簽署過的 APK 檔(此APK是已經上傳到開發者控制台的那個)。這裡要注意的是，裝置的 Android 版本必須是 2.2 版 (level 8) 以上，裝置中的 Google Play App 必須是 3.9.16 版以上。

如果你用你的開發者帳號去測試購買商品時，會出現"內容發佈者無法購買這項商品"的訊息，你必須使用測試帳號來登入你的裝置，才能用這個測試帳號來購買，另外帳號必須有設定信用卡資料。

### 使用靜態回應的方式測試IAP購買機制

透過發送靜態性質的金流回應(static billing responses)至APP來購買那些已先設置在網站內的特殊商品。

測試IAP機制，必須在實際裝置上安裝APP，無法使用模擬器來測試IAP機制。用來測試的裝置也有以下條件：Android1.6以上的標準Android版本、安裝了最新版的Android市集。如果Android市集軟體並非最新，APP就無法透過Android市集軟體發送出IAP金流請求。

建議第1次使用IAP機制時，跑一下靜態回應(static responses)這個流程。這能幫助驗證APP是否有對Android市集傳來的主要回應做出正確的處理。

如果要在靜態回應的環境底下去實作IAP機制，必須使用一些已預先設置的產品ID來做金流請求測試。每個已預先設置的產品ID皆會回傳一個特殊的靜態回應，而且不會有任何金錢上的交易發生。

不需要在軟體發佈網站上的APP產品列表裡列出這些已預先設置的商品，Android市集已經知道這些商品了。 也不需要在執行靜態回應測試這些已預設的商品ID上，將APP真實的發佈出去。可以很簡單的僅將APP安裝在裝置上，寫出裝置的log，並且發出這些已預設的商品的金流請求。

底下4組是預先設置好的產品ID，提供做靜態IAP機制回應的測試：

**android.test.purchased**  
當使用這個產品ID來發出金流請求時，Android市集會發出一個虛擬的購買商品成功回應回來。這個回應包含了JSON字串，這個字串含蓋了假的購買資訊(舉例：一個假的訂單ID)。

**android.test.canceled**  
當使用這個產品ID來發送金流請求時，Android市集會回應一個類似真正購買過程遭取消的反應回來。這可能發生當使用者在訂單流程中，發生了錯誤的狀況下。像是使用者使用了錯誤的信用卡、或者在付費之前，使用者按下了取消按鈕。

**android.test.refunded**  
當使用這個產品ID來發送金流請求時，Android市集會回應一個類似真實的產品被退費的反應回來。商品退費是無法透過Android市集的IAP機制服務來完成的(需使用merchant網站來退費)。在透過Google Checkout 帳戶跑完商品退費流程，退費訊息就會經由Android市集發送至APP裡。

**android.test.item\_unavailable**  
當使用這個產品ID做出金流請求時，Android市集會回應類似發送了一個發佈網站裡沒有列出的內部商品的購買請求的反應回來。

在AppCross Bookshelf 內做靜態測試，只要修改BillingHandler.java檔內的產品ID即可測試。

## AppCross Bookshelf In-App Billing程式實做說明

### 準備要使用 In-app Billing 的應用程式

* 下載 Library

打開 Android SDK Manager，拉到最下方的 Extras，找到 Google Play Billing Library 勾選後安裝。之後就會在你的 Android SDK 目錄中找到該 Library，位置如下：

/{你的SDK目錄}/extras/google/play\_billing/in-app-billing-v03

* 在開發者控制台(Google Developer Console)新增 App

當你在開發者控制台新增一個 App 時，會產生一個公開金鑰，只有在第一次建立 App 時才會產生這個金鑰，之後會用在你的 App 裡，用來和 Google Play 通訊。

執行步驟：

登入開發者控制台，你必須同時擁有 Google Checkout 帳戶。

在"所有應用程式"頁面中，

點選"新增應用程式"

輸入應用程式名稱

點選"準備商店資訊"

在"服務和API"頁面，記下由 Google Play 產生的 public license key，必須在應用程式中加入這組鑰匙。



* 在專案中加入 In-app Billing Library

要使用 Version 3 API 必須在專案中加入 IInAppBillingService.aidl 檔，透過它來和 Google Play 服務溝通。這個檔案可以在之前下載的 Library 中找到。

複製 IInAppBillingService.aidl 檔案到AppCross Bookshelf專案(在Eclipse 中 Import 到 /src 目錄)。

建置(Buile)專案，應該會在 /gen 目錄中看到自動產生的 IInAppBillingService.java。

在專案中加入 Sample 專案中 /util 目錄下的所有檔案，記得修改 package 名稱。

* 設定權限

在 AndroidManifest.xml 中的 permission 加入：

<uses-permission android:name="com.android.vending.BILLING" />

* 連結到 Google Play

綁定 (bind) Activity 到 Google Play 的 In-app Billing 服務，利用它來送出請求給 Google Play。在範例檔中提供了數個簡便的類別，去掌控綁定到 In-app Billing 服務的任務，所以你不必去處理網路連線的問題。

在 AppCross Bookshelf Source內的BillingHandler.java 中建立一個 IabHelper 物件，用來設定和 Google Play 的同步通訊。

之前在開發者控制台取得 Public Key 就是被編碼過的金鑰，利用 IabHelper 來取得還原後的 Public Key，程式碼如下：

IabHelper mHelper;

String base64EncodedPublicKey = "被編碼過的 PUBLIC\_KEY";

mHelper = new IabHelper(this, base64EncodedPublicKey);

接下來處理服務的綁定。呼叫 IabHelper 的 startSetup 方法，因為設定是非同步的，所以要建立一個設定完成的偵聽器，當 IabHelper 設定完成後，該偵聽器會被呼叫。在設定的同時，會一併檢查是否支援 In-app Billing API Version 3，程式碼如下：

mHelper.startSetup(new IabHelper.OnIabSetupFinishedListener() {

   public void onIabSetupFinished(IabResult result) {

      if (!result.isSuccess()) {

         // Oh noes, there was a problem.

         Log.d(TAG, "Problem setting up In-app Billing: " + result);

      }

         // IAB is fully set up!

   }

});

如果設定成功，就可使用 mHelper(IabHelper) 和 Google Play 服務做溝通。可以在 App 啟動時去查詢使用者擁有哪些內購式商品，然後記得在結束 Activity 時，要去釋放 mHelper ，如下：

@Override

public void onDestroy() {

   if (mHelper != null) mHelper.dispose();

   mHelper = null;

}

* 查詢已購買產品

查詢時呼叫 mHelper(IabHelper) 物件的方法 queryInventoryAsync(QueryInventoryFinishedListener)，參數是偵聽完成事件的偵聽器。程式碼如下：

mHelper.queryInventoryAsync(mGotInventoryListener);

IabHelper.QueryInventoryFinishedListener mGotInventoryListener = new IabHelper.QueryInventoryFinishedListener() {

public void onQueryInventoryFinished(IabResult result,

Inventory inventory) {

if (result.isFailure()) {

// handle error here

}

else {

// does the user have the premium upgrade?

mIsPremium = inventory.hasPurchase(SKU\_PREMIUM);

// update UI accordingly

}

}

};

* 購買內購式商品

要讓使用者購買某項內購式商品，要在程式內呼叫 mHelper(IabHelper) 物件的 launchPurchaseFlow(Activity, String, int, OnIabPurchaseFinishedListener, String) 方法，必須在 Activity 的主執行緒中呼叫，該方法的參數說明：

參數1：目前這個 Activity 本身。

參數2：產品ID(SKU)。

參數3：請求碼，任意正整數。Google Play 會回傳這個請求碼給 Activity 的 onActivityResult 。

參數4：偵聽完成事件。

參數5：用來附加其他資訊的字串，可以是空的。

使用範例碼：

mHelper.launchPurchaseFlow(this, SKU\_GAS, 10001, mPurchaseFinishedListener, "");

如果購買成功，Google Play 會將資料儲存在 Purchase 物件中回傳，然後偵聽就會收到，程式碼如下：

IabHelper.OnIabPurchaseFinishedListener mPurchaseFinishedListener = new IabHelper.OnIabPurchaseFinishedListener() {

   public void onIabPurchaseFinished(IabResult result, Purchase purchase)

   {

      if (result.isFailure()) {

         Log.d(TAG, "Error purchasing: " + result);

         return;

      }

      else if (purchase.getSku().equals(SKU\_GAS)) {

         // consume the gas and update the UI

      }

      else if (purchase.getSku().equals(SKU\_PREMIUM)) {

         // give user access to premium content and update the UI

      }

   }

};