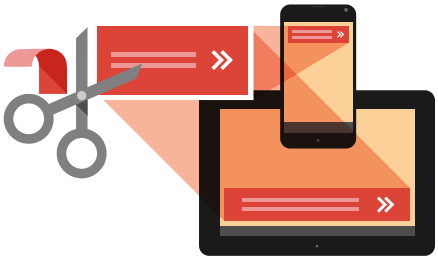
admob by Google



# TECHNICAL GUIDE FOR ANDROID DEVELOPERS COMBINING IN-APP ADVERTISING AND IN-APP PURCHASING



## ABOUT ADMOB

AdMob by Google has been helping app developers build app businesses since 2006. A leading mobile advertising network serving billions of ads daily, AdMob helps developers worldwide monetize and promote their mobile and tablet apps.

**www.google.com/admob**

## CASE STUDY

Watch the Hill Climb Racing video case study to hear how the developer, Fingersoft, successfully combined IAP and IAA within their app.

Learn how to successfully target your users through in-app advertising (IAA) and in-app purchasing (IAP) depending on their previous purchasing behaviour within your app.

In May 2013, Google announced that in-app purchases through the Google Play Store were up 700 percent year over year. While there's no doubt that IAP is a fast-growing trend, AdMob research shows that only 85% of users will actually make an in-app purchase.[1] So, how do you monetize the other 15% of your users not willing to buy items in your app? This guide explains how you can do that.

The following instructions are based on the Google Play In-app Billing v3 Library, and borrows helper classes from the sample project provided with the library.

### STEP 1

Initialize the in-app billing service, and wait for setup to be 'finished'.

```
protected void onCreate(Bundle savedInstanceState) {
  ...
  iabHelper = new IabHelper(this, BASE_64_ENCODED_PUBLIC_KEY);
  iabHelper.enableDebugLogging(true);
  iabHelper.startSetup(new IabHelper.OnIabSetupFinishedListener() {
    @Override
    public void onIabSetupFinished(IabResult result) {
      // Setup finished.
    }
  });
}
```

### STEP 2

Once the setup has finished, we can query the in-app billing service to see what items or subscriptions the user owns. Here is the implementation for onIabSetupFinished:

```
public void onIabSetupFinished(IabResult result) {
  if (!result.isSuccess()) {
    // Setup failed. Determine default behavior.
    return;
  }
  iabHelper.queryInventoryAsync(mGotInventoryListener);
}

IabHelper.QueryInventoryFinishedListener mGotInventoryListener =
      new IabHelper.QueryInventoryFinishedListener() {
    @Override
    public void onQueryInventoryFinished(IabResult result, Inventory
        inventory) {
      // Query inventory finished.
    }
};
```

### STEP 3

The `mGotInventoryListener` should check what purchases were made. The sample implementation below checks the test successful purchase, which is `android.test.purchased`. You'll want to check for your own purchase.

```
public static final String SKU_TEST_SUCCESSFUL_PURCHASE =
    "android.test.purchased";

public void onQueryInventoryFinished(IabResult result, Inventory
    inventory) {
  if (result.isFailure()) {
    // Query inventory failed. Determine default behavior.
    return;
  }
  // Check if user already make the purchase.
  Purchase testPurchase =
      inventory.getPurchase(SKU_TEST_SUCCESSFUL_PURCHASE);
  boolean hasTestPurchaseInventory =
      (testPurchase != null && verifyDeveloperPayload(testPurchase));
  if (hasTestPurchaseInventory) {
    // User made purchase!
  } else {
    // User has not made purchase. You can decide to show them ads.
    requestAds();
  }
}
```

### STEP 4

In a real application, you should also verify that the purchase was actually made. This adds additional security to make it harder for users to trick your app into thinking the purchase was made. When creating a purchase request, you can specify a developer payload with additional information specific to that request. Then you can verify the purchase by comparing that purchases payload:

```
private boolean verifyDeveloperPayload(Purchase purchase) {
  // Check purchase.developerPayload and make sure it matches the
  // payload you provided when you made the purchase request.
  return purchase.developerPayload.equals("Your secret payload string");
}
```

### STEP 5

If you determine that the user did not make a test purchase, then you can show ads. Here is a simple example showing how to add in AdMob ads:

```
private AdView adView;
private void requestAds() {
  // Create an AdView instance with your ad unit id.
  adView = new AdView(this, AdSize.SMART_BANNER, "YOUR_AD_UNIT_ID");

  // Find the container in your layout to place the ad.
  RelativeLayout container =
      (RelativeLayout) findViewById(R.id.adContainer);
  container.addView(adView);

  // Show the ad.
  adView.loadAd(new AdRequest());
}
```

### STEP 6

The above shows how to check for previous purchases and not show ads if previous purchases were made. You'll likely also want to turn off ads the moment the user makes a purchase.

When the user indicates they'd like to make a purchase, you can call `launchPurchaseFlow` on the `IabHelper`:

```
iabHelper.launchPurchaseFlow(this, SKU_TEST_SUCCESSFUL_PURCHASE,
    RC_REQUEST, mPurchaseFinishedListener, payload);
```

where `SKU_TEST_SUCCESSFUL_PURCHASE` is your purchase string ("`android.test.purchased`" in this example), `RC_REQUEST` is an arbitrary request code, and payload is a string that you should verify later to make sure the purchase was valid. `mPurchaseFinishedListener` is implemented as follows:

```
IabHelper.OnIabPurchaseFinishedListener mPurchaseFinishedListener =
    new IabHelper.OnIabPurchaseFinishedListener() {
  @Override
  public void onIabPurchaseFinished(IabResult result, Purchase purchase)
{
    if (result.isFailure()) {
        // Setup failed. Determine default behavior.
        return;
    }
    if (!verifyDeveloperPayload(purchase)) {
        // Purchase not valid. Determine default behavior.
        return;
    }

    if (purchase.getSku().equals(SKU_TEST_SUCCESSFUL_PURCHASE)) {
        // Successfully bought test purchase.
        removeAds();
    }
  }
};
```

### STEP 7

If the purchase is successful and verified, you can remove the ads for the remainder of the user's session:

```
private void removeAds() {
  if (adView != null) {
    // Find the container in your layout where you replaced the ad.
    RelativeLayout container =
        (RelativeLayout) findViewById(R.id.adContainer);

    // Remove the ad.
    container.removeView(adView);

    // Set adView to null.
    adView = null;
  }
}
```

### Success

You can now show ads to your users who haven't made an in-app purchase! Check out the sample app for a complete implementation. This sample was written using Google Mobile Ads SDK v6.4.1, not the new AdMob APIs in Google Play services.

1. *Mobile Apps Consumer Study*, *AdMob and Parks Associates*, Oct 2013

admob by Google