Virtual Good Promotion for Unity

Technical Integration Documentation

San Francisco, June 25, 2012

This document explains how to integrate PlayHaven's *Virtual Good Promotion* and *In-App Purchase Tracking* tools into a Unity iOS project.

Please follow these steps to integrate VGP:

[0. Define promotional/discounted IAP Products in iTunes Connect]
1. Add promotional IAP Products to your PlayHaven Dashboard account
2. Create Virtual Good Promotions in the PlayHaven Dashboard and assign them to placements
3. Request content for those placements in your Unity project, using *PlayHavenContentRequester* objects
4. Integrate Virtual Good Promotions with your application's in-app purchase process
5. Report all In-App Purchases back to PlayHaven for user segmentation purposes

Steps (1) and (2) are preparatory stages before the actual code implementation, and (3), (4) and (5) represent the actual technical integration of VGP & IAP Tracking.

Let's go over each step in detail:

1. Add promotional IAP Products to your PlayHaven Dashboard account

In your PlayHaven Dashboard game settings, check out the *IAP Products* section and *add an IAP product* (https://dashboard.playhaven.com/#/publisher/gameSettings/4626/iapProducts). The fields you will need to fill out correspond exactly to your iTunes Connect account settings: Product ID, Description, Type, Price Tier, Cleared for Sale.

2. Create Virtual Good Promotions in the PlayHaven Dashboard and assign them to placements

In your PlayHaven Dashboard account, add a *Virtual Good Promotion* (https://dashboard.playhaven.com/#/publisher/virtualGoods/<YOUR_GAME_ID>/edit), pick one of the IAP Products you have defined at step 1, set a quantity ("1" by default, can be used to sell IAP Products in "bulk") and assign it to one or more of your game's *placements* (http://help.playhaven.com/customer/portal/articles/243504-what-are-placements).

3. Request content for those placements in your Unity project, using PlayHavenContentRequester objects

The main Unity Documentation file explains how to request content for your placements. In a nutshell, you need to add a *PlayHavenContentRequester* component to one of your game objects, assign it to a placement and call its *Request()* method. This will issue an HTTP content request to PlayHaven's Client API, which will then respond with the promotion data, depending on the placement you specified earlier.

4. Integrate Virtual Good Promotions with your application's in-app purchase process

Once the API responds with a Virtual Good Promotion, the *OnPurchasePresented* event of your *PlayHavenManager* instance will be fired. Therefore, your application needs to register a method for this event:

```
public delegate void PurchasePresentedTriggerHandler(int requestId, Purchase purchase);
public event PurchasePresentedTriggerHandler OnPurchasePresented;
```

The *OnPurchasePresented* handler will provide a *PlayHaven.Purchase* object to your application. The *purchase* object is a key element of the VGP integration, and contains the attributes below. Save a reference to it for tracking this PlayHaven-powered IAP in step (5) of the integration.

- *productIdentifier* (the same one you defined in step (1))
- *quantity* (the number of IAP Products purchasable through this VGP)
- *receipt* (for App Store verification)

The *purchase* object contains all the information you need to initiate an in-app purchase with Apple. Since there has been no interaction with the user yet, you will need to call the appropriate method(s) in your application to actually prompt the user for the purchase and complete the transaction. Some applications have a singleton class (e.g. *PurchaseManager*) in charge of initiating and completing the virtual good purchases. If say there is a *doPurchase* method on this class, you will need to call it and pass the *productIdentifier* and *quantity* fields you collected from the *purchase* object. Treat this as a regular end-user initiated purchase; PlayHaven's Virtual Good Promotion can be forgotten for the time being.

Once the user finishes the VGP powered transaction (*buy*, *cancel* or *error*), the VGP content unit (i.e. ad unit) needs to be dismissed so that gameplay can be resumed. To achieve this, call the *ProductPurchaseResolutionRequest* method on the scene's *PlayHavenManager* instance:

```
public void ProductPurchaseResolutionRequest(PurchaseResolution resolution)
```

The *resolution* is a simple enum field that should be set to one of the following values, depending on whether the Apple transaction was completed, canceled or resulted in an error:

```
public enum PurchaseResolution { Buy, Cancel, Error };
```

Please note that PlayHaven's Unity plugin sets a timeout for the VGP content unit, which will be dismissed after a few seconds of inactivity, though calling *ProductPurchaseResolutionRequest* is recommended for proper synchronization of the transaction and gameplay.

5. Report all In-App Purchases back to PlayHaven for user segmentation purposes

Call the method below (exposed by the *PlayHavenManager* instance) to report the resolution of ALL your application's IAPs back to PlayHaven. If your project has a centralized location/singleton class that streamlines your virtual good purchase process, you will need to identify the code that receives Apple's resolution, build a *Purchase* object (defined in the *PlayHaven* namespace), and call this method:

```
public void ProductPurchaseTrackingRequest(Purchase purchase, PurchaseResolution resolution)
```

The *purchase* argument (of type *Purchase*) should be identical in format to the one received earlier through the *PurchasePresentedTriggerHandler* delegate, while *resolution* is a simple enum field, defined at step (4):

```
public enum PurchaseResolution { Buy, Cancel, Error };
```

Please note that step (5) is crucial in segmenting users based on purchase behavior. All in-app purchase transactions should be reported back to PlayHaven (both VGP initiated and regular IAPs). Without this part of the integration, there would be no way to target PlayHaven promotions based on how much your users spend in your game.

These are the essential functions/events that should be remembered:
public event PurchasePresentedTriggerHandler OnPurchasePresented;
public void ProductPurchaseResolutionRequest(PurchaseResolution resolution)
public void ProductPurchaseTrackingRequest(Purchase purchase, PurchaseResolution resolution)

PlayHaven - Unity - VGP Compatibility

The *Virtual Good Promotion* feature is accessible starting with PlayHaven's Unity Plugin's 1.10.5 version. Please note that VGP is currently only available for iOS projects, with Android soon to follow.

Testing VGP & Dashboard Metrics

Please note that testing VGP in your IAP Sandbox Environment might inflate the number of transactions reported by PlayHaven's Dashboard, as there is no way for our API to distinguish between "real" and test transactions. This is important to keep in mind before unleashing the full power of VGP in your live application.

SAMPLE CODE (C#)

* Create a C# script component (*PlayHavenIAPPurchaseHandler*) and assign it to your *PlayHavenManager* game object:

```
using UnityEngine;
using System.Collections;

public class PlayHavenIAPPurchaseHandler : MonoBehaviour
{
  private PlayHavenManager playHaven;

  void Awake()
  {
    playHaven = PlayHavenManager.instance;
    playHaven.OnPurchasePresented +=
HandlePlayHavenManagerinstanceOnPurchasePresented;
  }

  void HandlePlayHavenManagerinstanceOnPurchasePresented (int requestId,
PlayHaven.Purchase purchase)
  {
    PurchaseManagerSingleton.buyProduct(purchase.productIdentifier,
purchase.quantity);
    PurchaseManagerSingleton.OnIAPCompleted += CompleteVGP;
  }

  void CompleteVGP(PlayHaven.PurchaseResolution resolution)
  {
      playHaven.ProductPurchaseResolutionRequest(resolution);
  }
```

```
}


////....switching to your application's code...

using PlayHaven;

public class PurchaseManager : MonoBehavior
{
  private PlayHavenManager playHaven;
  public event IAPCompleteTriggerHandler OnIAPCompleted;
  public delegate void
IAPCompletedTriggerHandler(PlayHaven.PurchaseResolution resolution);

  void buyProduct(string product_id, int quantity)
  {
    // first, initiate and complete the transaction with Apple
    // let's say the response is saved in "apple_response" (resolution,
receipt etc.)

    // Report the resolution back to the PlayHaven-powered content unit to
dismiss it
    resolution = PlayHaven.PurchaseResolution.Buy;
    switch(apple_response.resolution) {
      case "success"  : resolution = PlayHaven.PurchaseResolution.Buy; break;
      case "cancel"   : resolution = PlayHaven.PurchaseResolution.Cancel;
break;
      case "error"    : resolution = PlayHaven.PurchaseResolution.Error;
break;
      default         : resolution = PlayHaven.PurchaseResolution.Error;
break;
    }

    if (OnIAPCompleted != null)
    {
      OnIAPCompleted(resolution);
    }

    // Track all IAPs with PlayHaven
    PlayHaven.Purchase purchase = new PlayHaven.Purchase();
    purchase.productIdentifier = product_id;
    purchase.resolution = resolution;
    purchase.receipt = apple_response.receipt;

    playHaven.ProductPurchaseTrackingRequest(purchase, resolution);
  }
}
```