Amazon Native Plugin

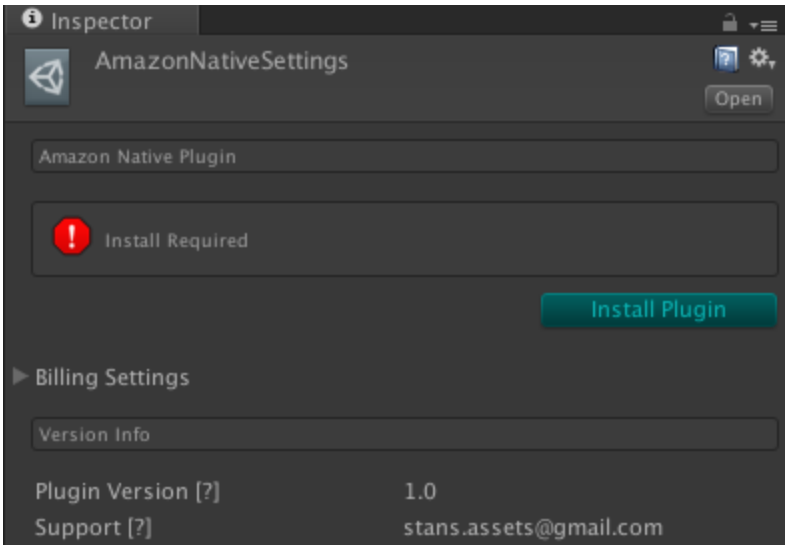
Anyone can comment

Short Overview

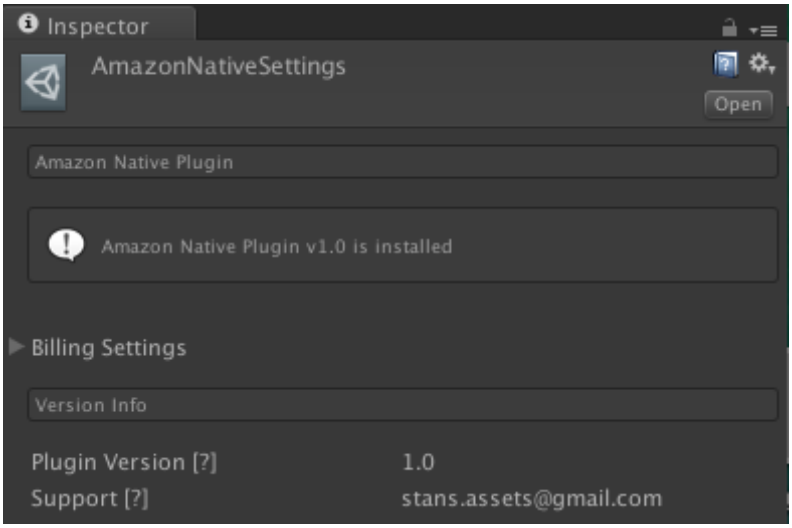
This plugin, will provide easy and flexible functionality for Amazon native functions which are not available from clean Unity. (In-app purchases, GameCircle, etc).

Setup

Plugin may require some small set up action in order to compile in your device with no issue. If plugin isn't installed the settings window will be opened automatically in the Inspector view. If for some reason it wasn't opened after you imported the plugin, go to: **Window → Amazon Native → Edit Settings**



And just press **Install Plugin** button. If installation was completed successfully you should see message as on picture below:


Inspector
AmazonNativeSettings
Open
Amazon Native Plugin
Amazon Native Plugin v1.0 is installed
Billing Settings
Version Info
Plugin Version [?]
1.0
Support [?]
stans.assets@gmail.com

How to update

1. Version Notes

With every new update I try to make the plugin better. Add new features, improve stability, usability and code base structure.

When a new version is available, you can find out what's new in the version and version history by pressing version number on Asset Store Plugin Page:

2. Updating

I recommend to check Version Notes before updating.

Sometimes in order to implement new feature or improve code structure I have to change some of plugin files / folder or method names.

It will be of course described in version notes. But if you simple click update in the Asset

Store, you may get duplicated or conflicted files.

After new plugin version is downloaded and unpacked to your project the settings window will be opened automatically in the Inspector view. If for some reason it wasn't opened after you imported the plugin, go to:
Window → Amazon Native → Edit Settings
And just press **Update** button. If installation was completed successfully you should see message as on picture below.

Released Apps with the plugin

Billing

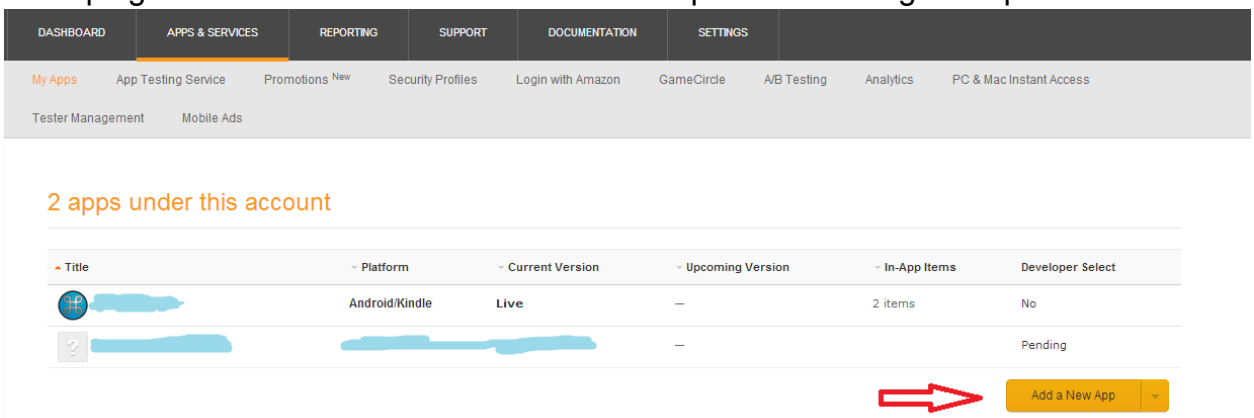
Setup

Make sure that:

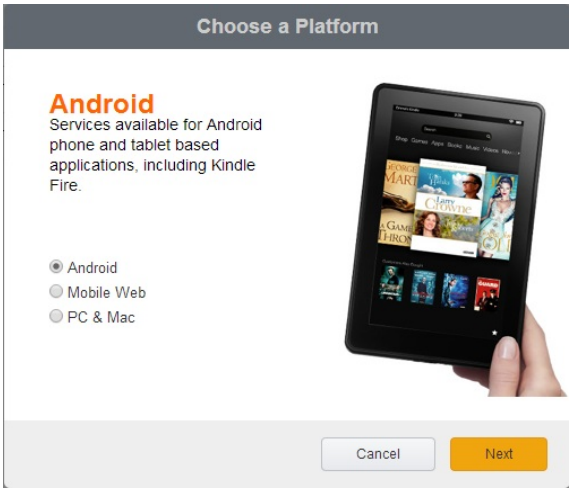
- AndroidManifest.xml is inside your **Assets/Plugins/Android** folder
- api_key.txt is inside your **Assets/Plugins/Android/assets**
- amazon_gc_prototype folder and amazon_gc_prototype.zip is inside your **Assets/Plugins/Android/res/raw**
- amazon_gc_styles is inside in **Assets/Plugins/Android/res/values**
- the obligatory presence of the folders **drawable** and **layout** all files
- **amazon.sdktester** in inside **Assets/Plugins/Android**

In-App Purchasing handles the details of purchase flow, payment processing, receipts, and rights management for the purchasable content. Now with unique receipt identifiers, you can easily ensure that customers receive purchased in-app items quickly, as well as track transactions and fulfillment.

1. Create new account at Amazon Developer Console if you have not registered previously.
2. To implement in-apps in your application you should create new amazon application in **https://developer.amazon.com/myapps.html** and pass some info to the plugin. See instructions below how to set up and run billing example scene.



3.
4. Create new Application in Amazon Developer Console. You needed choose Android platform:



5. Then fill in all fields when creating an application. After these manipulations we have an existing application.
6. Copy and enter your SKU to AndroidManifest.xml in **Assets/Plugins/Android**
7.

```
<?xml version="1.0" encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
android:installLocation="preferExternal"
package="ENTER_YOUR_SKU" android:versionName="1.0"
android:versionCode="1">
```

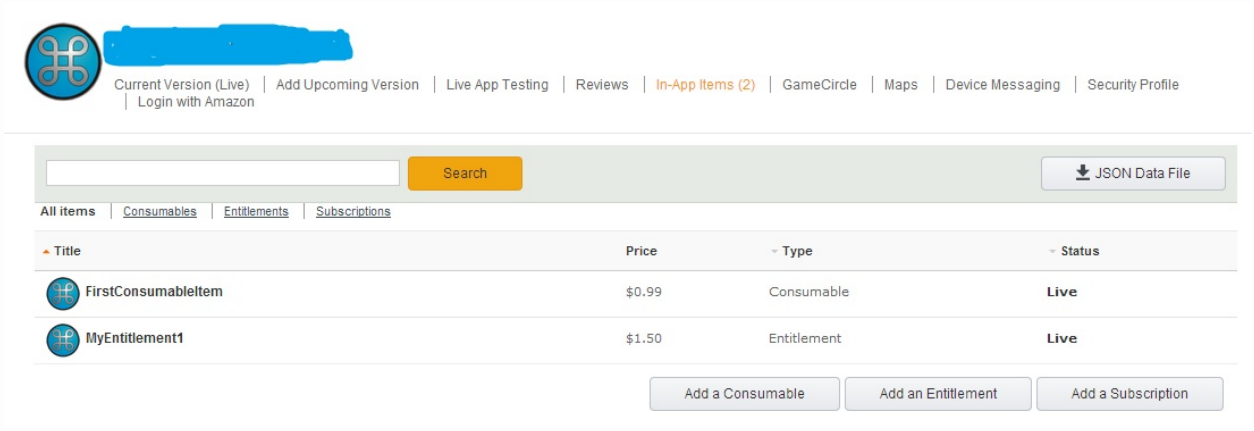
And here:

```
<activity
android:name="com.amazon.identity.auth.device.authorization.AuthorizationActivity"
    android:allowTaskReparenting="true"
    android:launchMode="singleTask"
    android:theme="@android:style/Theme.NoDisplay" >
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT" />
            <category android:name="android.intent.category.BROWSABLE" />
            <data android:scheme="amzn"
                  android:host="ENTER_YOUR_SKU" />
        </intent-filter>
</activity>
```

8. You needed to fill in all the tabs in your application, so that it satisfies all the requirements of Amazon's Developer Console:

9. Now go directly to a tab In-App Items and create products of any type you need by pressing respectively:

- Add a Consumable
- Add an Entitlement
- Add a Subscription



Setting Up for Test Purchases

To test your In-app Billing implementation with actual in-app purchases, you will need to click to download the JSON Data File, there by obtaining the file **amazon.sdktester** for conducting TEST PURCHASES and then **copy this file to your device**.

App Tester

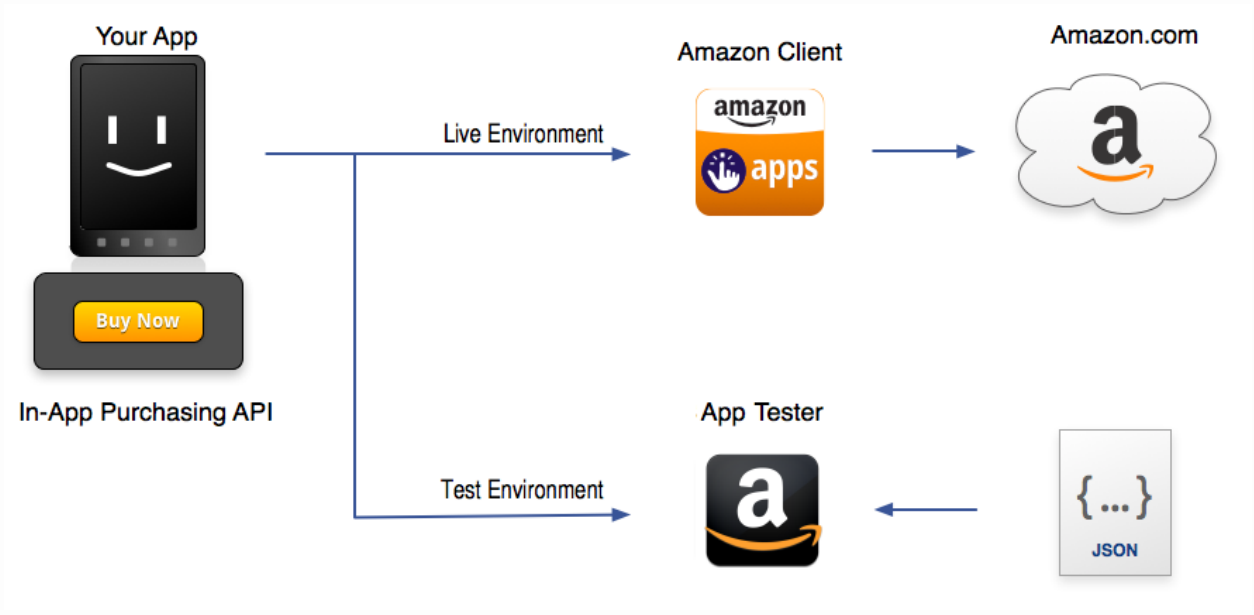
The App Tester is a developer tool that allows users of the Amazon Mobile App SDK to test their implementation in a production-like environment before submitting it to Amazon for publication. This will allow developers to construct test cases that cover all responses generated by the Amazon Mobile App SDK APIs, and give them confidence their apps will be published and run correctly.

In a live environment, your app makes API calls to the Amazon Mobile App SDK libraries.

These libraries, in turn, make IPC calls to the Amazon Appstore on the device. The Amazon Appstore then communicates with the Amazon services to fulfill requests such as the purchase of a consumable item, or a subscription to a magazine.

Only apps that have been approved and published through the Amazon Apps & Games Developer Portal can communicate with the Amazon Appstore. This is a problem for developers who want to see how their apps run in a live environment before it is submitted for publication. The App Tester was created to solve this problem.

This document will describe the use of the App Tester to test apps using the Amazon In-App Purchasing APIs (IAP APIs). This is the mechanism of shopping at amazon API:



The API libraries will detect whether your app has been published through the Amazon Apps & Games Developer Portal. If it has, the API will exchange data with the Amazon Appstore. If the API detects that the app has not been published through the portal, the API will exchange data with App Tester. This ensures that you will be able to use the same codebase for testing that you intend to submit for publication.

Since your app's interface with the Amazon Appstore is isolated to the API libraries, your app will not know that it is talking with App Tester. Because you have control over the responses

given by the App Tester, you can test your app's behavior under any response condition.

Before using the App Tester, developers should be familiar with the IAP APIs as described in the documentation available within the Amazon Apps & Games Developer Portal. Testing an app using the App Tester requires:

- An Android Development Environment that allows logging via ADB (Logcat)
- An Android device supporting minimum SDK version 10
- An Android app that uses the Amazon In-App Purchasing APIs
- The Amazon App Tester
- A JSON data file (amazon.sdktester.json) that contains IAP item information

It is assumed that the developer has an environment that supports ADB and can capture logging information from the device under test via Logcat.

For help developing an application that uses the IAP APIs, you can see detailed documentation and sample code on the Amazon Apps & Games Developer Portal.

You can download the App Tester from the Amazon Appstore.

The following section will describe how to create a JSON data file that can be used with the App Tester, followed by instructions for configuring and using the App Tester as part of your test procedures.

Using App Tester

Once you have loaded the app under test, the App Tester , and the JSON data file onto the device, you are ready to start testing the IAP API portions of your application. This section will walk you through each of the screens and explain their use during the test cycle.

Signing and Uploading apk with Unity

To be able to create in-app purchases you should upload your apk file to the developer console. Apk must be signed with your private key.

Next step is app configuration.

You have to choose your bundle bundle ID- This is your **SKU.**

A bundle ID otherwise known as a **package** in Android is the unique identifier for all Android apps. It needs to be unique as when you upload it to AmazonStore it identifies and publishes your app use the package name as the unique app identification.

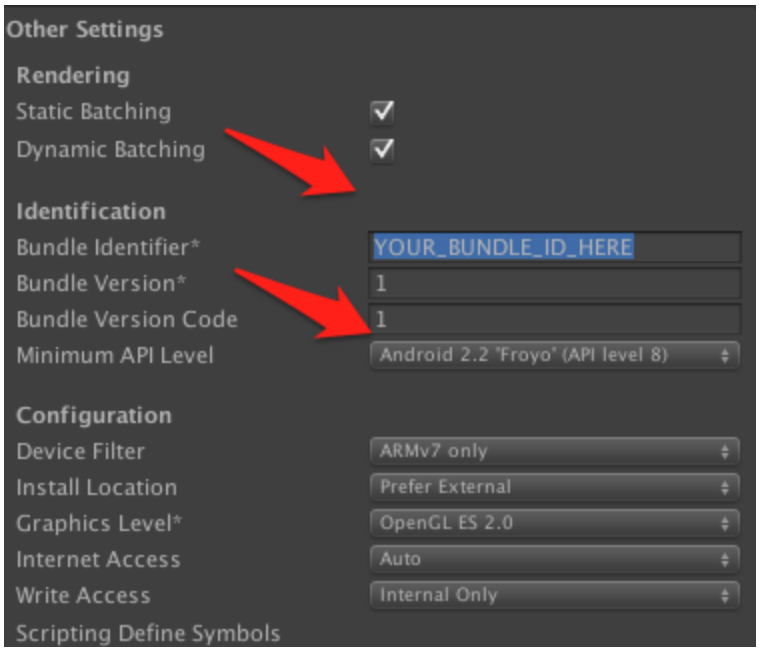
Really it is the only thing which is necessary to identify your app, and generally it has 3 parts:

com.example.testapp

Where **example** is generally the company/publishers name, and **testapp** is the app name.

You will not be able to upload an APK to the store which has the same package as another app already in the store.

When you bundle ID is ready add it to the Unity application setting and to the AndroidManifest.xml.



You can build your signed apk file now. Just press **build** button.

Note: You should have latest android SDK on your computer, to make Unity able build apk

file.

Note: Android plugin should be included to your application, if you will build signed application without plugin included, application will not have permissions to use billing.

After signed apk is created you can upload it to the Amazon App Dev Console.

Choose **Binary files** than upload your apk.

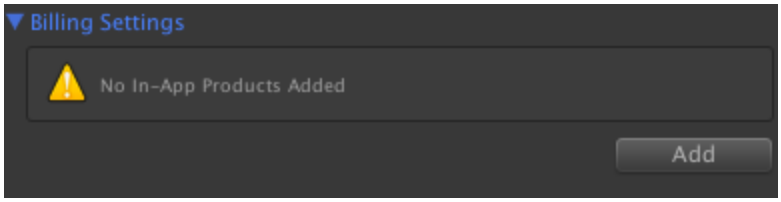
Classes Documentation

SA_AmazonBillingManager class.

API methods:

Add product's SKU, with will be registered after billing initialization, do this before calling loadStore function

You can ignore this function if you already set all your product id's in settings(Window -> Amazon Native -> Settings):




```
public void Initialize();
public void Initialize(string[] products_ids);
```

Get's registred products details.Triggers ITEM_DATA_REQUEST_RECEIVED event.

```
private void Init(string [] product_ids)
```

Purchase the product. Triggers PURCHASE_RECEIVED event.

```
public void Purchase(string SKU)
```

Events:

Fires when load ITEMS Data flow end's with success or fail. Event data contains AMN_ItemDataResult

ITEM_DATA_REQUEST_RECEIVED

Fires when purchase product flow end's with success or fail. Event data contains AMN_PurchaseResult.

PURCHASE_RECEIVED

Fires when a purchase update request flow end's with success or fail. Event data contains AMN_PurchaseUpdatesResult.

PURCHASE_UPDATES_RECEIVED

Callback when the SDK is available. Event data contains AMN_SdkAvailableResult.

SDK_AVAILABLE

Callback for when the user ID is available. Event data contains AMN_GetUserIdResult.

GET_USER_ID_RESPONSE

SA_AmazonItem class

Getters:

item sku

public string Sku;

item Title

public string Title;

item description

public string Description;

item Type

public string Type;

item Price

public string Price;

item SmallIconUrl

public string SmallIconUrl;

SA_AmazonReceipt class

Getters:

receipt sku

public string Sku;

receipt Type

public string Type;

receipt Token

public string Token;

receipt SubscriptionStartDate

public string SubscriptionStartDate;

receipt SubscriptionEndDate

public string SubscriptionEndDate;

AMN_ItemDataResult class

Contains information about unavailable ITEMS.

public List<string> UnavailableSkus

Contains information about available ITEMS.

public List<SA_AmazonItem> AvailableItems;

AMN_PurchaseResult class

Contains information about purchased receipt.

public SA_AmazonReceipt Receipt

contains response message

public string Reason;

AMN_PurchaseUpdatesResult class

Contains information about revoked skus.

public List<string> revokedSkus;

Contains information about receipts.

public List<SA_AmazonReceipt> receipts;

AMN_SdkAvailableResult class

Contains information about result of sdk

public true isAvailable;

AMN_GetUserIdResult class

Contains userID

public string UserId;

GameCircle

Before you begin

- You should have your Android development environment set up.
- You should have a physical device running Android 2.3 or higher for testing.

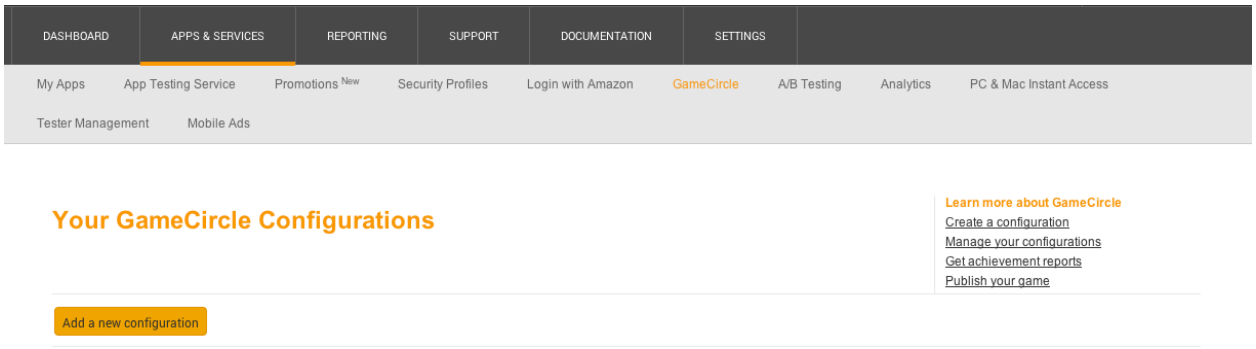
Check / Download all necessary files and prepare your device

if you did not install in advance in the Billing Setup, please click on the link and follow the installation having from 1 to 8 inclusive item.

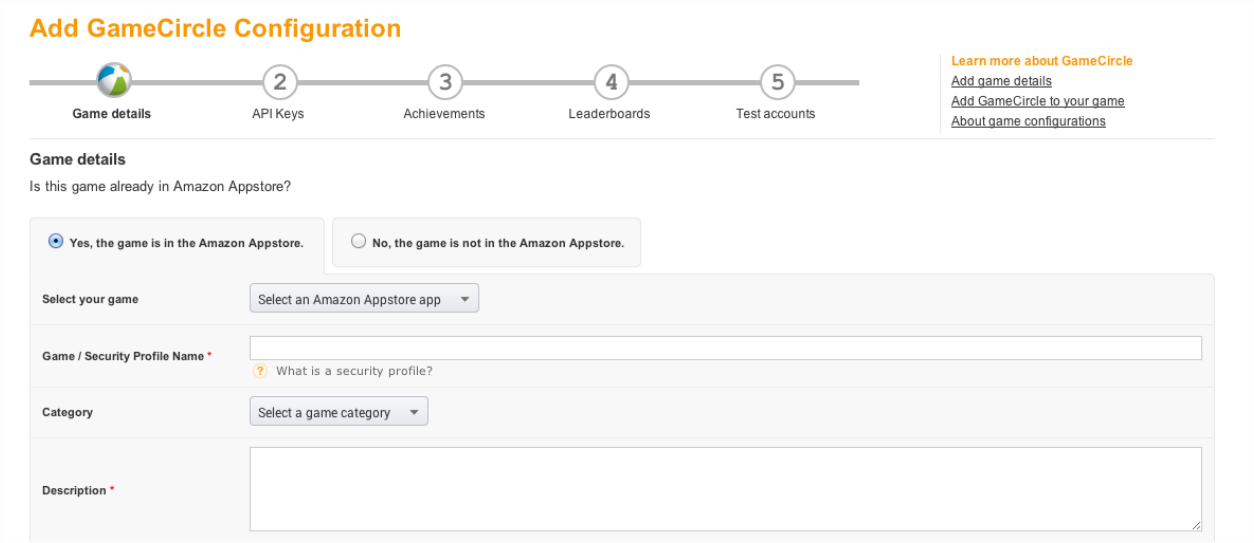
Set up the game in the Developer Console

The Amazon App Developer Console is where you manage game services for your game, and configure metadata for authorizing and authenticating your game.

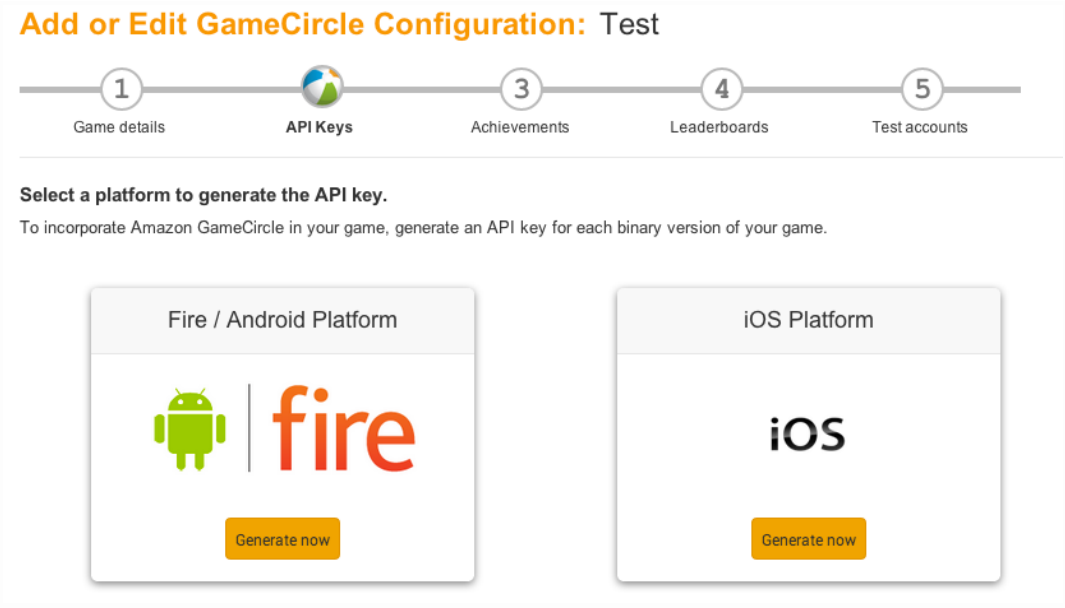
Now select GameCircle and click Add New Configuration:



Select your app from the proposed connection with GameCircle:

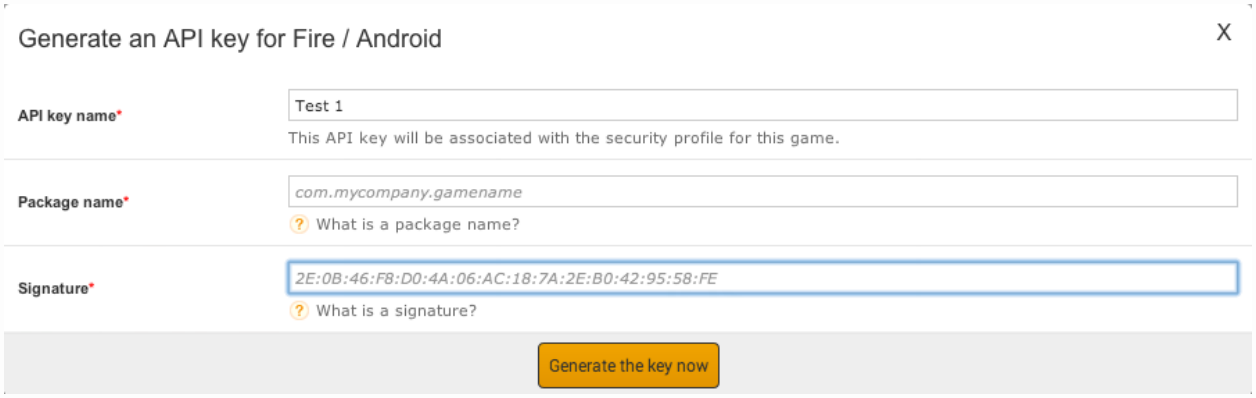


After you fill out all fields in this step, click **Save And Continue**. Go to the API Keys. To incorporate Amazon GameCircle in your game, generate an API key for each binary version of your game. Choose Android platform:



Select any appropriate name for the API Key Name, Package name and Signature. Clicking on the question mark tells you how must look package name. Select a platform **to generate the API key**. It is very important that you comply with the points and the

sequence. How to obtain a Signature you must go to the link. In the future, the API KEY will be needed to link our project with Amazon



After successfully obtaining the API KEY, click Continue and proceed to the creation of our achievements and leaderboards. Worth noting is the fact that you must fill in all fields to add all the icons, follow all installation is very delicate moment. After successfully creation Achievements & Leaderboards clicking on Continue.

Last step. To test unpublished leaderboards and achievements for your game, create test accounts here by adding Amazon GameCircle nicknames to the list below. You can clear leaderboard scores and achievement progress for a test account at any time. And click Finish. That`s all.

Modify your code

To run the game, you need to configure AndroidManifest, you needed to change 2 fields:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:installLocation="preferExternal" package="YOUR_BUNDLE_ID"
android:versionName="2.0" android:versionCode="2">

<data android:scheme="amzn" android:host="YOUR_BUNDLE_ID" />
```

Paste you API_KEY to api_key.txt is inside your **Assets/Plugins/Android/assets.**

Classes Documentation

SA_AmazonGameCircleManager class

API methods:

Should be called on application start. It will create connection to the play service and sign in user if user was signed before. Best practice to call it only once. Any way other calls will be ignored by the plugin.

public void connect()

Events:

Fires when Initializes app. Event data AMN_InitializeResult;

INITIALIZE_RESULT

Fires for Request the local player information. Event data *AMN_RequestPlayerDataResult;*

REQUEST_PLAYER_DATA_RECEIVED

Fires for requests a list of all achievements was received. Event data contains AMN_RequestAchievementsResult

REQUEST_ACHIEVEMENTS_RECEIVED

Fires when an achievement was changed. Event data contains AMN_UpdateAchievementResult

UPDATE_ACHIEVEMENT_RECEIVED

Fires for request all leaderboards for this game. Event data contains

AMN_RequestLeaderboardsResult

REQUEST_LEADERBOARDS_RECEIVED

Fires when submit a score to leaderboard. Event data contains AMN_SubmitLeaderboardResult

SUBMIT_LEADERBOARD_RESULT

API methods:

Show default Google Play Achievements UI

```
public void ShowAchievementsOverlay()
```

Show default Google Play Leaderboards UI

```
public void ShowLeaderboardsOverlay()
```

Show Amazon GameCircle UI

```
public void ShowGCOverlay()
```

Trigger player info request

```
public void RetrieveLocalPlayer()
```

Trigger submit score request

```
public void SubmitLeaderBoardProgress(string leaderBId, long score)
```

Trigger leaderboards info request

```
public void RequestLeaderboards()
```

Trigger Achievements info request

```
public void RequestAchievements()
```

```
Trigger update achievement points
public void UpdateAchievementProgress(string achieve_id, float score)
```

AMN_InitializeResult class

```
Contains Init result.
public bool  isSuccess;

public string  Error;
```

AMN_RequestPlayerDataResult class

```
Contains result message
public string Error;

Contains player info
public SA_AmazonGSPlayer Player;
```

AMN_RequestAchievementsResult class

```
Contains result message
public string Error;

Contains list of achievements
public List<SA_GCAchievement> AchievementList;
```

AMN_UpdateAchievementResult class

```
Contains result message
public string Error;

Contains achievementID
public string  AchievementID
```

AMN_RequestLeaderboardsResult class

```
Contains result message
public string Error;

Contains list of achievements
public List<SA_GCLeaderboard> LeaderboardsList;
```

AMN_SubmitLeaderboardResult class

```
Contains result message
public string Error;

Contains LeaderboardID
public string  LeaderboardID
```

SA_AmazonGSPlayer class.

Getters:

```
Contains playerID
public string  PlayerId;

Contains player Alias
public string Alias;

Contains player Avatar URL
public string AvatarUrl;
```

SA_GCAchievement class.

```
Contains Achievement Title
public string Title;

Contains Achievement Id
public string Id;

Contains Achievement Description
public string Description;

Contains Achievement Progress
public float Progress;

Contains Achievement PointValue
public int PointValue;
```

```
Contains Achievement state of Hide
public bool IsHidden;

Contains Achievement state of Unlock
public bool IsUnlocked;

Contains Achievement Position
public int Position;

Contains Achievement DateUnlocked
public DateTime DateUnlocked;
```

SA_GCLeaderboard class.

```
Contains Leaderboard Id
public string Id;

Contains Leaderboard Name
public string Name;

Contains Leaderboard DisplayText
public string DisplayText;

Contains Leaderboard ScoreFormat
public string ScoreFormat;

Contains Leaderboard ImageUrl
public string ImageUrl;
```

PlayMaker Actions

The plugin contains playmaker actions.

The actions scripts can be found in the zip archive at:

Assets/Extensions/AmazonNative/Addons/PlayMakerActions

You can simply unrar it to the same folder and Amazon Native actions will appear under playmaker actions menu.

Actions list:

Billing

- AMN_InitBilling
- AMN_Purchase

GameCircle

- AMN_GameCircleInit
- AMN_CheckConnection
- AMN_LoadGCAchievements
- AMN_LoadGCLeaderboards
- AMN_SubmitLeaderboardScore
- AMN_UpdateAchievement

Frequently Asked Questions

Any of plugin functions is not working.

Plugin will work only on **real device**, do not try to use in in the Unity Editor all plugin function calls will be simply ignored.

I Prefer to use official Amazon Service SDK

The Amazon Native plugin hase own event driven implementation of Amazon Service SDK.

Can I use this plugin with other Android Plugins from Asset Store

Yes you can. All you need to do is to update existing AndroidManifest.xml: Enter your SKU to AndroidManifest.xml in **Assets/Plugins/Android**

```
<?xml version="1.0" encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
android:installLocation="preferExternal"
package="ENTER_YOUR_SKU" android:versionName="1.0"
android:versionCode="1">
```

And here:

```
<activity
android:name="com.amazon.identity.auth.device.authorization.AuthorizationActivity"
    android:allowTaskReparenting="true"
    android:launchMode="singleTask"
    android:theme="@android:style/Theme.NoDisplay" >
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT" />
            <category android:name="android.intent.category.BROWSABLE" />
            <data android:scheme="amzn"
                  android:host="ENTER_YOUR_SKU" />
        </intent-filter>
</activity>
```

How to get logcat log

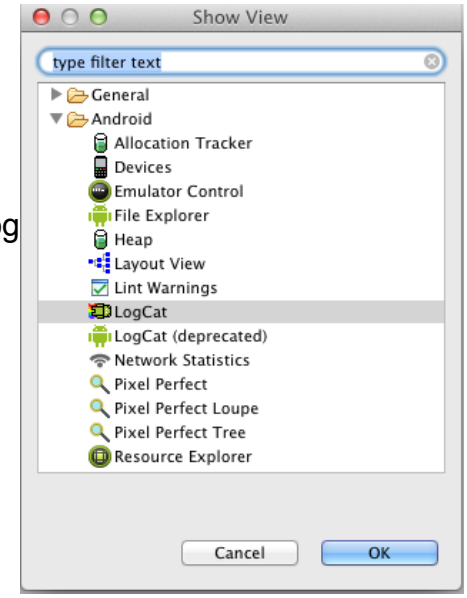
For getting the logcat log you can use should:

1. Enable USB debugging in your device.
2. Connect device to computer
3. Use console command `$ adb -d logcat`

Instead of console command you may use any other visual log viewers for android. For example from the ADT(Eclipse) with you got [Android SDK download page](#).

To do this open ADT, choose **Window** → **Show View** → **Other…**

It will open Show View window. Choose **Android** → **LogCat** And you will able to see the logs from your device.



Example Scenes

In App Purchases

Billing Example

This example scene can be found at
Assets/Extensions/AmazonNative/Examples/Scenes/Billing/BillingAPI.

The controller script **SA_AmazonBillingExample.cs** is attached to the **_Controller** gameobject.

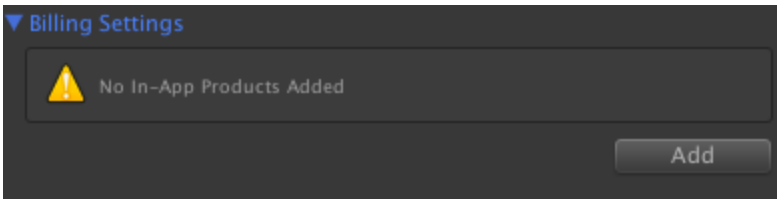
Scene will demonstrate usage for the following of In-App Purchases API.

[Setup]

Make sure you have finished billing setup guide. Before testing add test account at Amazon Developer Console and complete your billing setup for test Purchase.

Also make sure what you copied amazon.sdktester on your device and have completed all settings

Windows → Amazon Native → Edit Settings



The example scene is using Android **my own product_ids.** How ever you can change this by replacing product_ids in **SA_AmazonBillingExample.cs** class

```
private string SKU_EXAMPLE = "first_Item";
```

GameCircle

GameCircle Example

This example scene can be found at
Assets/Extensions/AmazonNative/Examples/Scenes/GameCircle/GameCircleAPI

The controller script **SA_AmazonGCExample.cs** is attached to the **_Controller** gameobject.

Scene will demonstrate usage for the following Game Circle APIs

- Connecting
- Retrieving Player Info
- Achievements
- Leaderboards

[Setup]

You need to complete the [Game Circle Setup]() Guide

And update **SA_AmazonGCExample.cs** script with your info

```
//example
private const string leaderboard_id = "MyLeaderBoard1";
//private const string leaderboard_id = "REPLACE_WITH_YOUR_ID";

private const string achieve_id = "MyAchiev1";
//private const string achieve_id = "REPLACE_WITH_YOUR_ID";
```