Ultimate Mobile

Anyone can comment

Short Overview

Ultimate Mobile is a combo package with contains following assets.

Android Native Plugin
IOS Native
WP8 Native
Google Mobile Ads SDK
Google Analytics SDK
Mobile Native PopUps

As bonus package also contains Unified API for:

- In-App purchasing - **IOS**, **Android**, **WP8**
- Game Services (Achievements, Leaderboards) - **IOS**, **Android**
- Advertisement - **IOS**, **Android**, **WP8**
- Social Sharing - **IOS**, **Android**
- Google Analytics Tracking - **All Unity Platforms**
- Native Pop Ups and Preloaders - **IOS**, **Android**, **WP8**

Note: You not have to use Unified API, you always can use API of any plugin in package.

More unified API will be available in future releases. Feel free to send feedback or feature request to support team.

Warning: Plugin does not contains reach social API like Android Native or Mobile Social Plugin. Only native posting is available.

Setup

All you have to do is copy files from

Assets/Plugins/StansAssets to → **Assets/Plugins**

If you already have some files under **Assets/Plugins/Android** it mean that you may already have some android plugin installed. In this case, please read this sections with will help you to merge the plugins:

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

How to compile androidnative.jar from eclipse project

How to merge manifest with another android plugin

If the plugin has conflict with the IOS / WP8 / Android (which does not overrides main activity) Please contact support team.

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. Avoiding conflicts

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 Asset Store version, you may get duplicated or conflicted files.

Warning: Check the Version Notes before updating. If the version notes contains

Code Refactor: version section. It means that plugin structure is changed. Some files can be removed or function names changed. If files was removed, you should remove it by your self from the project. Example or Remove notes:

Removed:
Assets/Extensions/AndroidNative/Other/Twitter
All Scene under xExample/Scenes Moved to corresponding folder
only Preview Scene should be under xExample/Scenes

Note: If you own another plugins with also have `GooglePlayCommon` folder (this folder is shared between few plugins in order to supply compatibility of android plugins) I also recommend update those plugins too. To avoid conflicts

3. Saving Plugins settings

Plugin setting that was specified in editor GUI earlier will be overridden. So just backup your settings data with stored in files:

```
Assets/Extensions/AndroidNative/Resources/AndroidNativeSettings
Assets/Extensions/GooglePlayCommon/Resources/SocialSettings
```

And replace plugin files with your backup after update. Or uncheck this files when you installing the update

4. Update Android Native files and libs

All native android files and libraries are stored Under `Assets/Plugins/StansAssets/*`

Advantages

- When you updating the plugin you will not override your own changes, for example if you did changes in AndroidManifest.xml, or in ids.xml. Or if you compiled your own version of androidnative.jar
- You can simply remove all plugin files before update. Because with this approach all plugin files located in the `Assets/Plugins/StansAssets` and `Assets/Extensions/AndroidNative`

Disadvantages

- After update you should manually copy updated files from `Assets/Plugins/StansAssets` to your `Assets/Plugins` folder.

In-App Purchases

Setup

Before you can integrate in-app purchases for your game you should complete your app setup for platforms you going to use.

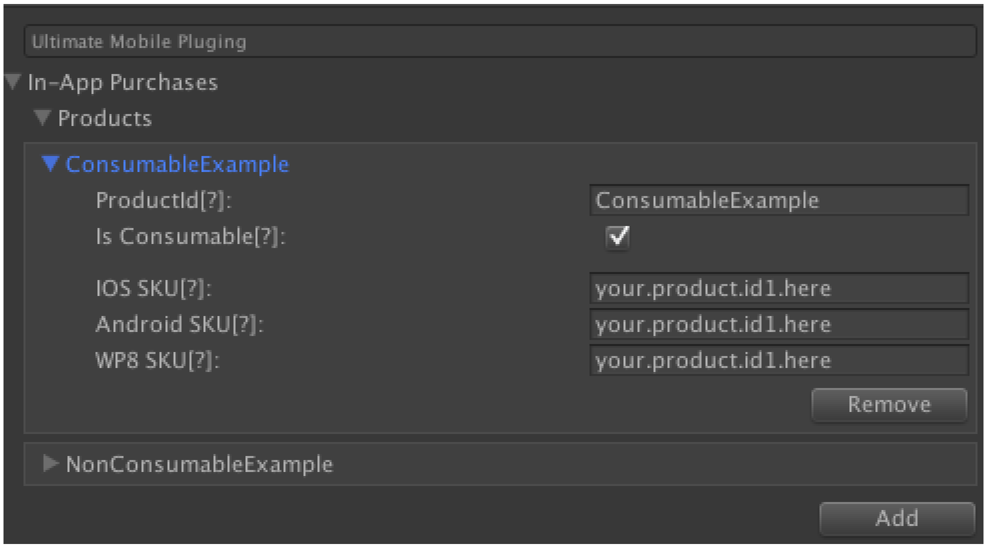
IOS In-App purchases Setup Guide

WP8 In-App purchases Setup Guide

Android In-App purchases Setup Guide

After you done with platforms set up, fill product info int the plugin settings.

Windows → Ultimate Mobile → Edit Settings



Ultimate Mobile Pluging

▼ In-App Purchases

▼ Products

▼ ConsumableExample

ProductId[?]: ConsumableExample

Is Consumable[?]: ☑

IOS SKU[?]: your.product.id1.here

Android SKU[?]: your.product.id1.here

WP8 SKU[?]: your.product.id1.here

▶ NonConsumableExample

Coding Guidelines

First of all you need to init in-app manager. Best practise is to do this on your app start.

```
UM_InAppPurchaseManager.instance.Init();
```

As soon a UM_InAppPurchaseManager will be initialized you will ge ON_BILLING_CONNECT_FINISHED event.

To purchase product use:

```
UM_InAppPurchaseManager.instance.Purchase(YOUR_PRODUCT_ID);
```

As soon a product will be purchased you will ge ON_PURCHASE_FLOW_FINISHED event.

To check if product was already purchased use:

```
UM_InAppPurchaseManager.instance.IsProductPurchased(YOUR_PRODUCT_ID);
```

```
For restoring transactions of IOS you may use:
IOSInAppPurchaseManager.instance.restorePurchases();
```

More example can be found in the **UM_BillingExample.cs** script.

API References

UM_InAppPurchaseManager : SA_Singleton<UM_AdManager>

init in-app manager

public void Init()

purchase products by id (with you specified in plugin in-apps editor settings)

public void Purchase(string productId)

check if product was already purchased by id

```
public bool IsProductPurchased(string id)
```

```
check if product was already purchased
public bool IsProductPurchased(UM_InAppProduct product)
```

GET / SET

```
true if in-app was inited
public bool IsInited
```

UM_InAppProduct

GET / SET

```
tue if product is comsumable
public bool IsConsumable;
```

```
product id
public string id = "new_product";
```

```
IOS product id
public string IOSId;
```

```
Android product id
public string AndroidId;
```

```
WP8 product id
public string WP8Id;
```

```
WP8 product template
public WP8ProductTemplate WP8Template
```

```
IOS product template
public ProductTemplate IOSTemplate
```

```
Android product template
public GoogleProductTemplate AndroidTemplate
```

Game Services

Plugin is using Play Service API on Android platform and Game Center API on IOS platform.

Setup

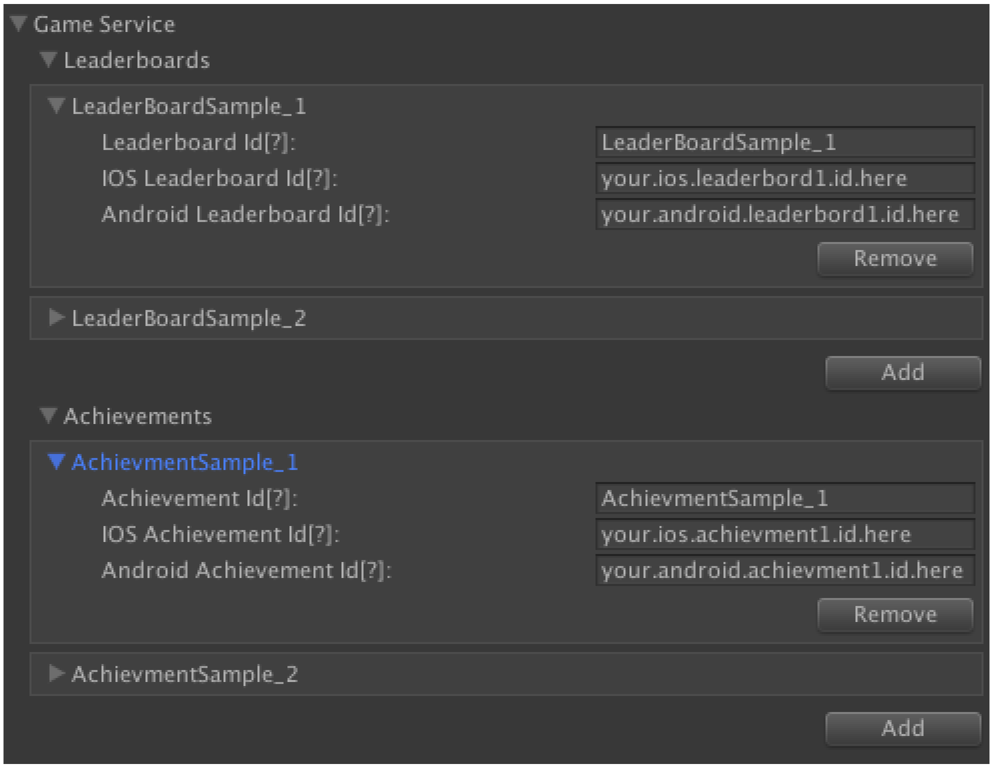
Before you can use unified API, complete setup steps for service you going to use

Play Service Setup - Android

Game Center Setup - IOS

After you done with platforms set up, fill in leaderboards and achievements info in the plugin settings.

Windows → Ultimate Mobile → Edit Settings



Game Service

Leaderboards

LeaderBoardSample_1

Leaderboard Id[?]: LeaderBoardSample_1

IOS Leaderboard Id[?]: your.ios.leaderbord1.id.here

Android Leaderboard Id[?]: your.android.leaderbord1.id.here

LeaderBoardSample_2

Achievements

AchievmentSample_1

Achievement Id[?]: AchievmentSample_1

IOS Achievement Id[?]: your.ios.achievment1.id.here

Android Achievement Id[?]: your.android.achievment1.id.here

AchievmentSample_2

Coding Guidelines

Coming soon.

API References

UM_GameServiceManager : SA_Singleton<..>

```
init in-app manager
public  void Connect()
public void Disconnect()
public void ShowAchivmentsUI()
public void RevealAchievement(string id)
public void RevealAchievement(UM_Achievement achievement)
public void ReportAchievement(string id)
public void ReportAchievement(UM_Achievement achievement)
public void IncrementAchievement(string id,  float percentages)
public void IncrementAchievement(UM_Achievement achievement, float percentages)
public void ResetAchievements()
public float GetAchievementProgress(string id)
public float GetAchievementProgress(UM_Achievement achievement)


public void ShowLeaderBoardsUI()
public void ShowLeaderBoardUI(string id)
public void ShowLeaderBoardUI(UM_Leaderboard leaderboard)
public void SubmitScore(string LeaderboardId, int score)
public void SubmitScore(UM_Leaderboard leaderboard, int score)
public int GetCurrentPlayerScore(string leaderBoardId)
public int GetCurrentPlayerScore(UM_Leaderboard leaderboard)
public int GetCurrentPlayerRank(string leaderBoardId)
public int GetCurrentPlayerRank(UM_Leaderboard leaderboard)
```

GET / SET

public UM_ConnectionState ConnectionSate

public GameServicePlayerTemplate player

UM_Leaderboard

GET / SET

public string id

public string IOSId

public string AndroidId

UM_Achievement

GET / SET

public string id

public bool IsIncremental

public string IOSId

public string AndroidId

GameServicePlayerTemplate

GET / SET

public string PlayerId

public string Name

public Texture2D Avatar

public GameCenterPlayerTemplate GameCenterPlayer

public GooglePlayerTemplate GooglePlayPlayer

Advertisement

Setup

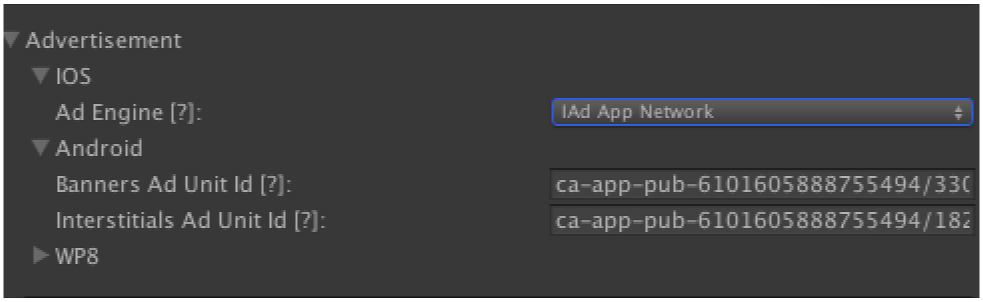
Before you can use unified API, complete setup steps for service you going to use

iAd App network - not set up action required

[Google Mobile Ad - IOS, Android, WP8]()

After you done with platforms set up, fill in ad settings info in the plugin settings.

Windows → Ultimate Mobile → Edit Settings



▼ Advertisement

▼ IOS

Ad Engine [?]: IAd App Network

▼ Android

Banners Ad Unit Id [?]: ca-app-pub-6101605888755494/330

Interstitials Ad Unit Id [?]: ca-app-pub-6101605888755494/182

▶ WP8

Coding Guidelines

First of all you need to init ad manager. Best practise is to do this on your app start.

```
UM_AdManager.instance.Init();
```

Star interstitial ad is simple, all you have to do is to call:

```
UM_AdManager.instance.StartInterstitialAd();
```

it will initialize interstitial request and interstitial will show up as soon as it’s loaded

To create banner use:

```
int bannerId1 = UM_AdManager.instance.CreateAdBanner(TextAnchor.UpperLeft);
```

Make sure you sotre created banner id. You need this is to control the banner by usign following functions:

```
UM_AdManager.instance.HideBanner(bannerId1);

UM_AdManager.instance.ShowBanner(bannerId1);

UM_AdManager.instance.RefreshBanner(bannerId1);

UM_AdManager.instance.DestroyBanner(bannerId1);
```

or checking banner states:

```
UM_AdManager.instance.IsBannerLoaded(bannerId1)

UM_AdManager.instance.IsBannerOnScreen(bannerId1)
```

Use example can be founded at **UM_AdExample.cs** script.

API References

UM_AdManager : SA_Singleton<UM_AdManager>

init ad manager

public void Init()

create banner with anchor

public int CreateAdBanner(TextAnchor anchor)

show banner by banner id

public void ShowBanner(int id)

hide banner by banner id

public void HideBanner(int id)

refresh banner by banner id

```
public void RefreshBanner(int id)
```

check if banner was loaded by banner id

```
public bool IsBannerLoaded(int id)
```

check if banner is on screen by banner id

```
public bool IsBannerOnScreen(int id)
```

destroy banner by banner id

```
public void DestroyBanner(int id)
```

GET / SET

true if UM_AdManager was inited

```
public bool IsInited
```

Social Sharing

Plugin contains Cross platform native platform sharing API implementation.

Advantages:

- You do not need to create any social app on Facebook or Twitter website
- You do not need to prompt user to authenticate with your app with requested permissions before you can use API
- User will not leave your app for Twitter authentication.
- Posting looks natively to platform

Disadvantages

- You can only prompt user to post messages / images to twitter.
- You will not know or be able to respond if the user cancels their post from the native app.

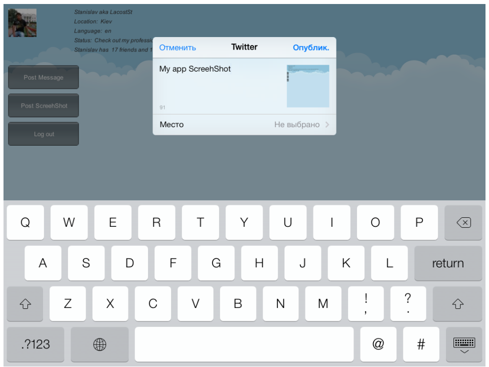
If you need more API of social platform, please have a look at Mobile Social Plugin

Coding Guidelines

Posting to Twitter

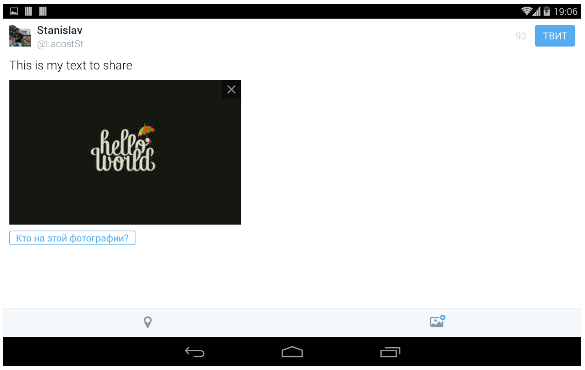
```
UM_ShareUtility.TwitterShare("This is my text to share");
UM_ShareUtility.TwitterShare("This is my text to share", ImageToShare);
```

Result of this API calls would be similar to screenshot below:



Stanislav aka LacostSt
Location: Kiev
Language: en
Status: Check out my professi
Stanislav has 17 friends and 1

My app ScreehShot



Stanislav
@LacostSt

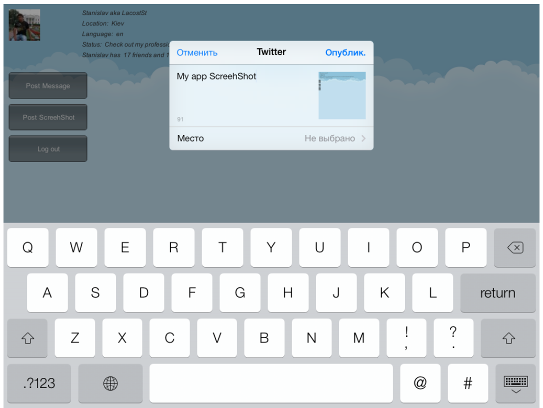
This is my text to share

Posting to Facebook

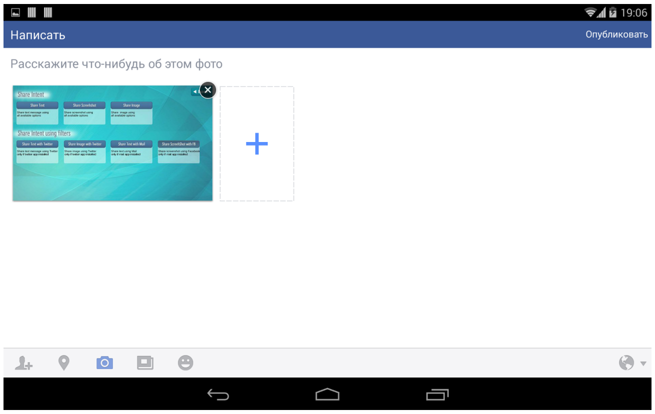
```
UM_ShareUtility.FacebookShare("This is my text to share");

UM_ShareUtility.FacebookShare("This is my text to share", ImageToShare);
```

Result of this API calls would be similar to screenshot below:



Stanislav aka LacostSt
Location: Kiev
Language: en
Status: Check out my professi
Stanislav has 17 friends and 1

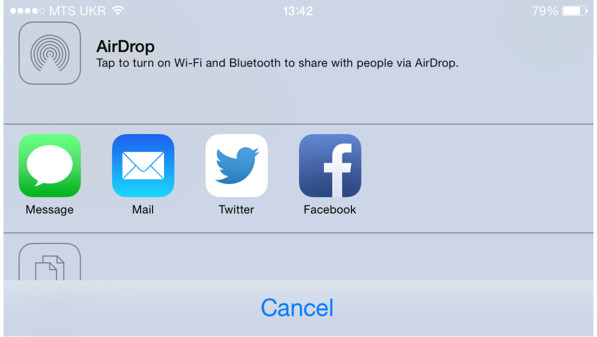


Sharing Media

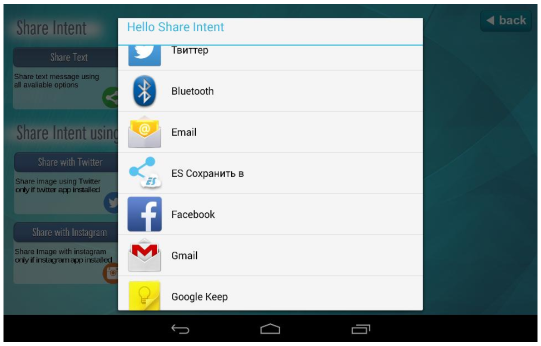
```
UM_ShareUtility.ShareMedia("This is my text to share");

UM_ShareUtility.ShareMedia("This is my text to share", ImageToShare);
```

Result of this API calls would be similar to screenshot below:



AirDrop
Tap to turn on Wi-Fi and Bluetooth to share with people via AirDrop.



Share text message using all avaliable options

Share image using Twitter only if twitter app installed

Share Image with instagram only if instagram app installed

Analytics Tracking

Analytics tracking documentation can be found here.

Native Pop Ups and Preloaders

Full native pop ups and preloaders usage described in the Mobile Native Pop-ups document