Adlantis SDK README For Cocos2d-x

About

This document explains how to show AdLantis ads in your Cocos2d-x application. Currently we support following development tools:

- Cocos2d-x version: 3.0+
- development language: C++
- iOS development IDE: Xcode
- Android development IDE: Eclipse

Run Sample Application

Copy files

- Copy cocos2d direcory to `adlantisSample/`.

  You may use `cocos new` to create a fresh new project and copy its cocos2d directory to `adlantisSample/`.

- Copy `adlantis` directory to `adlantisSample/cocos2d/plugin/plugins/`.

Run on iOS

- Open `adlantisSample/proj.ios/adlantisSample.xcodeproj` in xcode.
- Build and run sample project.
- Xcode7 If you use more , please set the " Build Settings " → " Build Options " → " Enable Bitcode " to No. This is because the current state of the Cocos2d-x does not support Bitcode. Please make this set to Yes because if the future Cocos2d-x has the support of the SDK Adlantis is Acknowledged .

Note: To build for 64-bit, you will need to go to the issue navigator. On both the `cocos2d_libs.xcodeproj` and `PluginProtocol.xcodeproj` projects you will need to click on "Validate project Settings. Update to recommended settings". This will bring up a dialog. Click the "Perform Changes" button for both projects.

Run on Android

- Change dir to `adlantisSample/cocos2d/plugin/tools`.
- Edit `config.sh`, add `adlantis` to `ALL_PLUGINS`

```
#define plugins array
export ALL_PLUGINS=("adlantis")
```

- Run `./publish.sh`, adlantis plugin should be built successfully
- Import project into Eclipse
  - Import cocos2d library
    - Import `cocos2d/cocos/platform/android/java`.
  - Import sample project
    - Import `adlantisSample/proj.android`.
- Set up Variables:

    - C/C++ Environment Variable `NDK_ROOT`:
        - Eclipse->Preferences->C/C++->Build->**Environment**.
        - Click **Add** button and add a new variable `NDK_ROOT` pointing to the root NDK directory
- Link Google Play Service to your project

  AdLantis SDK is using the Google Play services API to get an advertising identifier to track conversion. Please link Google Play services to your Eclipse project as follows:

    - Import `Google Play services` library project to your Eclipse workspace, make sure it can be built with no error.
    - Select your project, right-click and select "Android".
    - Click "Add" and choose the `Google Play services` project.

  Note: When using the Google Play services SDK, Android version should be 2.3 or higher. For full instructions on setting up Google Play services, see http://developer.android.com/google/play-services/setup.html.

- Build and run adlantisSample project

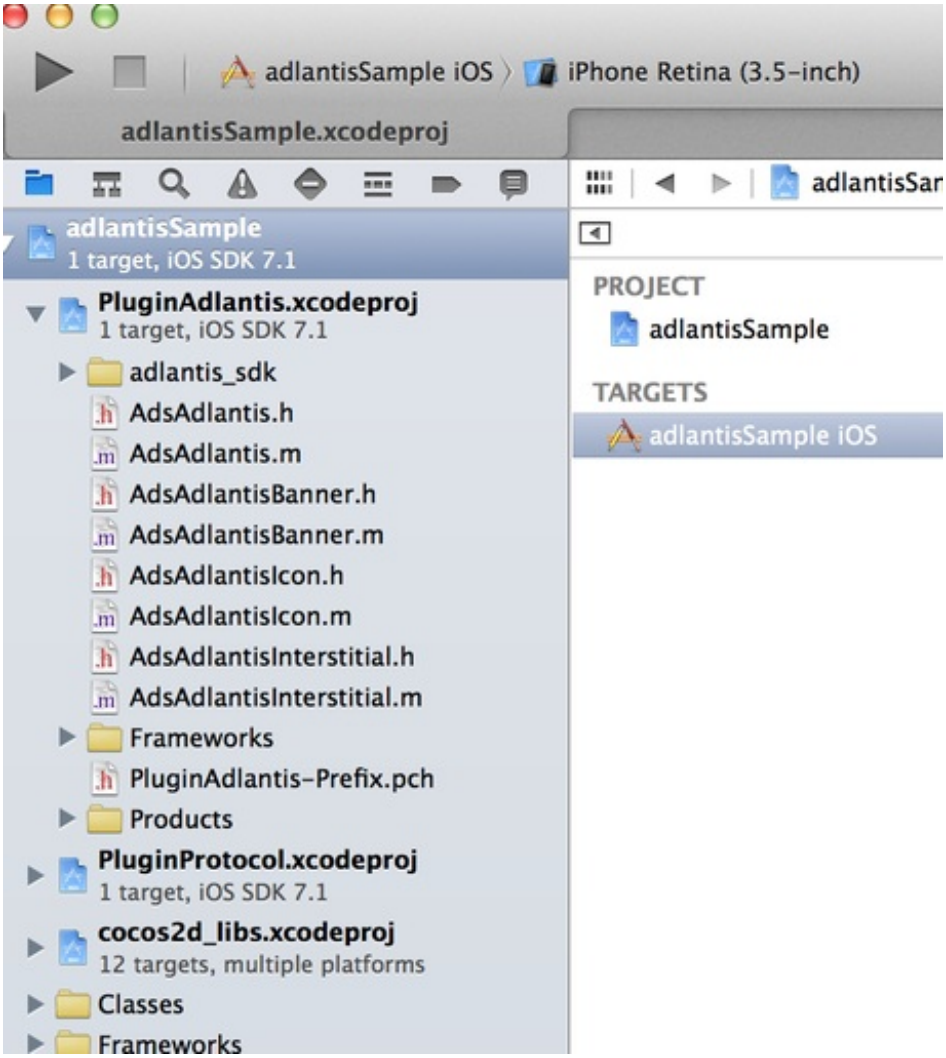
Implement AdLantis SDK to your Cocos2d-x project

Install

- Copy adlantis directory to `[your_project_path]/cocos2d/plugin/plugins/`

iOS

- Add Plugin projects to your Xcode projects
    - Xcode->File->Add Files to "yourProject"
        - Add cocos2d/plugin/protocols/proj.ios/PluginProtocol.xcodeproj
        - Add cocos2d/plugin/plugins/adlantis/proj.ios/PluginAdlantis.xcodeproj

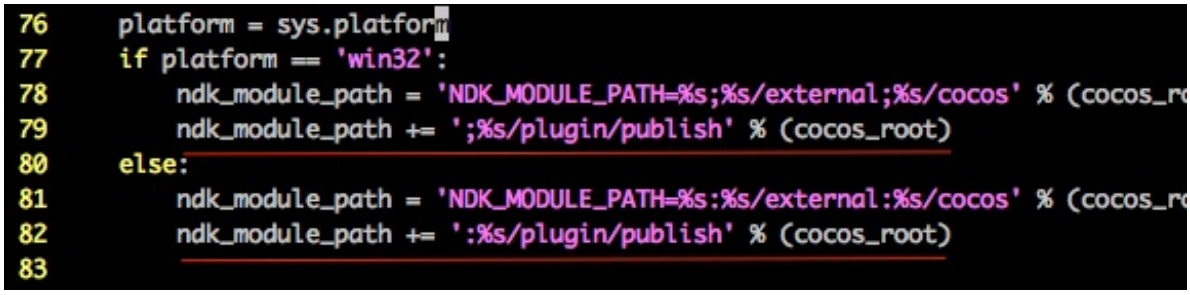


- Build Phases->Target Dependencies
  - add PluginAdlantis
  - add PluginProtocol
- Build Phases->Link Binary With Libraries
  - add libPluginProtocol.a
  - add libPluginAdlantis.a
  - add frameworks
    - `AdSupport.framework`
    - `CoreTelephony.framework`
    - `QuartzCore.framework`
    - `SystemConfiguration.framework`
- Build Settings
  - Header Search Paths
    - $(SRCROOT)/../cocos2d/plugin/protocols/include
  - Linking-Other Linker Flags
    - `-force_load` `$(BUILT_PRODUCTS_DIR)/libPluginAdlantis.a` (or -ObjC)

Android

(Based on Eclipse)

- edit `[your_project_path]/cocos2d/plugin/tools/config.sh`, add adlantis to ALL_PLUGINS
- run ./cocos2d/plugin/tools/publish.sh
- `[your_project_path]/cocos2d/plugin/tools/gameDevGuide.sh`, select adlantis
- edit `[your_project_path]/proj.android/build_native.py`, add `../cocos2d/plugin/publish` to `ndk_module_path`



```
76      platform = sys.platform
77      if platform == 'win32':
78          ndk_module_path = 'NDK_MODULE_PATH=%s;%s/external;%s/cocos' % (cocos_r
79          ndk_module_path += ';%s/plugin/publish' % (cocos_root)
80      else:
81          ndk_module_path = 'NDK_MODULE_PATH=%s:%s/external:%s/cocos' % (cocos_r
82          ndk_module_path += ':%s/plugin/publish' % (cocos_root)
83
```

- initiate Plugin-x on src/org/cocos2dx/cpp/AppActivity.java

```
public Cocos2dxGLSurfaceView onCreateView() {
    Cocos2dxGLSurfaceView glSurfaceView = new Cocos2dxGLSurfaceView(this);
    // TestCpp should create stencil buffer
    glSurfaceView.setEGLConfigChooser(5, 6, 5, 0, 16, 8);
    //initiate plugin-x
    PluginWrapper.init(this);
    PluginWrapper.setGLSurfaceView(glSurfaceView);
    return glSurfaceView;
}
```

- set Java VM for plugin(proj.android/jni/hellocpp/main.cpp)
  - Include header file

    ```
    #include "PluginJniHelper.h"
    ```

  - Set java vm

    ```
    void cocos_android_app_init (JNIEnv* env, jobject thiz) {
        LOGD("cocos_android_app_init");
        AppDelegate *pAppDelegate = new AppDelegate();
        // plugin-x
        JavaVM* vm;
        env->GetJavaVM(&vm);
        PluginJniHelper::setJavaVM(vm);
    }
    ```

- Link Google Play Service to your project

  AdLantis SDK is using the Google Play services API to get an advertising identifier to track conversion. Please link Google Play services to your Eclipse project as follows:

  - Import `Google Play services` library project to your Eclipse workspace, make sure it can be built with no error.
  - Select your project, right-click and select "Android".
  - Click "Add" and choose the `Google Play services` project.
  - Edit `AndroidManifest.xml` and add following tag as a child of the `<application>` element.

  Note: When using the Google Play services SDK, Android version should be 2.3 or higher. For full instructions on setting up Google Play services, see http://developer.android.com/google/play-services/setup.html.

Show Ads

include header files and namespaces

- import header file

```
#include "ProtocolAds.h"
#include "PluginManager.h"
```

- use cocos2d::plugin namespace

```
USING_NS_CC;
using namespace cocos2d::plugin;
```

Show banner ads

show banner as follows:

```
//load AdLantis plugin
ProtocolAds* _adlantisAdsBanner =
    dynamic_cast<ProtocolAds*>(PluginManager::getInstance()-
>loadPlugin("AdsAdlantisBanner"));
//set publisherID for banner
TAdsDeveloperInfo devInfo;
devInfo["AdlantisPublisherID"] = "xxx";//banner
_adlantisAdsBanner->configDeveloperInfo(devInfo);

//we do not use adInfo, but it's needed by plugin-x
TAdsInfo adInfo;
//set posisition for banner
ProtocolAds::AdsPos pos = ProtocolAds::kPosCenter;
//show banner ad
_adlantisAdsBanner->showAds(adInfo, pos);
```

- replace "xxx" with your publisherID.
- set pos to the position you want to show banner ad, following position is acceptable(defined by cocos2dx ProtocolAds).

```
typedef enum {
    kPosCenter = 0,
    kPosTop,
    kPosTopLeft,
    kPosTopRight,
    kPosBottom,
    kPosBottomLeft,
    kPosBottomRight,
} AdsPos;
```

and hide ads with following code

```
_adlantisAdsBanner->hideAds(adInfo);
```

Show interstitial ads

Define interstitial listener

In the header file of your scene which you want to show the interstitial ad, add the listener definition for interstitial

```
class AdlantisAdsListener : public cocos2d::plugin::AdsListener
```

```
{
public:
    virtual void onAdsResult(cocos2d::plugin::AdsResultCode code, const char* msg);
};
```

and declare a variable for it

```
private:
    AdlantisAdsListener* _listener;
```

Show interstitial ads as follows:

```
ProtocolAds* _adlantisAdsInterstitial =
    dynamic_cast<ProtocolAds*>(PluginManager::getInstance()-
>loadPlugin("AdsAdlantisInterstitial"));
//set publisherID for interstitial
TAdsDeveloperInfo devInfo;
devInfo["AdlantisPublisherID"] = "xxx";//interstitial
_adlantisAdsInterstitial->configDeveloperInfo(devInfo);

//set listener
_listener = new AdlantisAdsListener();
_adlantisAdsInterstitial->setAdsListener(_listener);

//we do not use adInfo, but it's needed by plugin-x
TAdsInfo adInfo;
//show interstitial ads
_adlantisAdsInterstitial->showAds(adInfo);
```

- replace "xxx" with your publisherID.
- implement the listener method, and add your logic when the ad displayed and dismissed. For example, you may want to pause your game when the interstitial ad is shown and resume it after the ad is dismissed.

Show icon ads (Android only)

Show icon ads as follows:

```
ProtocolAds* _adlantisAdsIcon =
    dynamic_cast<ProtocolAds*>(PluginManager::getInstance()->loadPlugin("AdsAdlantisIcon"));
//set publisherID for icon
TAdsDeveloperInfo devInfo;
devInfo["AdlantisPublisherID"] = "xxx";
_adlantisAdsIcon->configDeveloperInfo(devInfo);

//AdLantis icon 1
TAdlantisIconInfo iconInfo;
iconInfo["x"] = "10";//position.x
iconInfo["y"] = "10";//position.y
iconInfo["showText"] = "yes";// show text icon or not
iconInfo["textColor"] = "#ffffff";// the text color, default is black
PluginParam mapValue(iconInfo);
_adlantisAdsIcon->callFuncWithParam("add", &mapValue, NULL);

//add icon 2
iconInfo["x"] = "100";//position.x
iconInfo["y"] = "10";//position.y
iconInfo["showText"] = "no";// show text icon or not
PluginParam mapValue2(iconInfo);
```

```
_adlantisAdsIcon->callFuncWithParam("add", &mapValue2, NULL);

//we do not use adInfo, but it's needed by plugin-x
TAdsInfo adInfo;
//show icon ads
_adlantisAdsIcon->showAds(adInfo);
```

and hide ads with the following code:

```
_adlantisAdsIcon->hideAds(adInfo);
```

- replace "xxx" with your publisherID.
- before you call showAds, you can add up to 6 icons.

The coordinates for Icon Ad

Cocos2d-x uses pixel coordinates. The width or height of icons needs to be translated from points into pixels. This can be done with the following code:

```
PluginParam param("");
float contentScaleFactor = _adlantisAdsIcon->callFloatFuncWithParam("getScale", &param,
NULL);
```

The icon width can then be calculated as:

```
const float iconWidth = 57 * contentScaleFactor;
```

The icon height with or without text can be calculated as:

```
float iconHeight = (_textVisibility ? 75 : 57) * contentScaleFactor;
```