ANE For Mobage Native SDK Programming Guide

Introduction

This guide explains how to use the ANE For Mobage Native SDK to develop social application for the Mobage social platform (Currently only for Android).

Revision History

Ver.	Summary of Changes	Last Update
1.0	Initial Release	2013/09/10
1.1	Updated changes for NDK 1.4.5	2013/11/25

Contents

1. Outline of ANE For Mobage Native SDK

ANE For Mobage Native SDK enables you to develop applications for the Mobage Platform using Adobe’ s AIR technology. It makes it easy for you to integrate the feature-rich Mobage social functions into your AIR applications by combining Mobage’ s user-friendly social APIs with the simple yet powerful workflow provided by the AIR platform.

Setup requirements

The following are required to develop with the ANE For Mobage Native SDK. Please make sure that your development environment satisfies them.

- [OS] Mac OS X 10.8 Mountain Lion, Windows XP or above
- [Adobe Flash Builder] Adobe Flash Builder 4.7 or above (FLEX and AIR SDK 3.8 or above)
- [Target Device OS] Android 2.2 or later / iOS 5 or later

Environment used in this guide

- [OS] Mac OS X Mountain Lion (10.8.3)
- [Adobe Flash Builder] Adobe Flash Builder 4.7
- [Target Device OS] Android 4.0.4
- [Region] Japan

Known Issues

There are some known issues found in the current release. Please refer to the README for more details.

2. File hierarchy of the ANE For Mobage Native SDK

ANE For Mobage Native SDK consists of the following files and folders.

Files/Folders	Description
as/	ActionScript interface to android/ and ios/
android/	Eclipse project for Android (you will find project for jp and kr)
ios/	Xcode project for iOS
docs/	Documents regarding the ANE For Mobage Native SDK. The following are included: • ASDOCS/*: ANE For Mobage Native SDK API reference • ProgrammingGuide_AIR_en.pdf: This programming guide

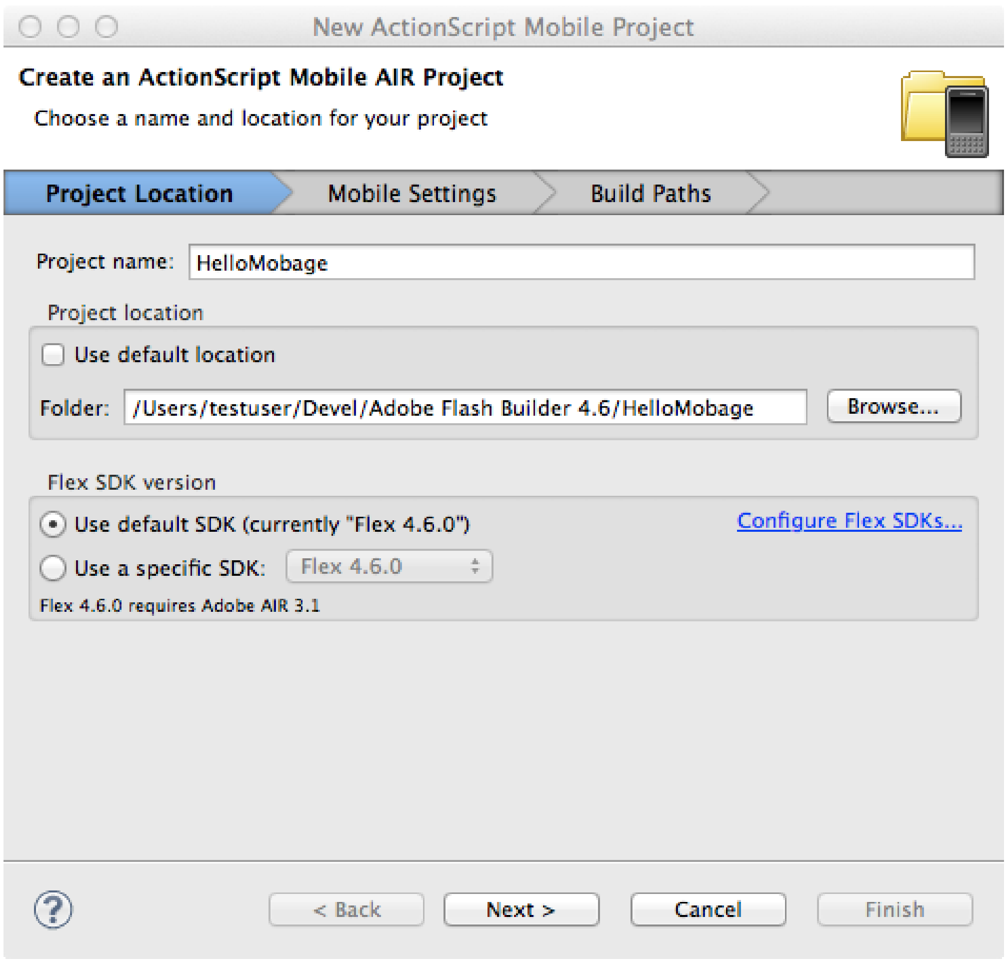
samples/	A simple Hello World sample to get you started. Section 4: “Adding ANE For Mobage Native SDK Social Functions” guide you through it. + UICompSample project which will provide you most of the Mobage Native SDK methods to test.
README.txt	Contains important information about the current release and change history. Please be sure read it first.

3. Setting up the ANE For Mobage Native SDK

This chapter presents a step-by-step guide to actually use the ANE For Mobage Native SDK to develop a simple application for Android (iOS is also mentioned in this chapter). You should be able to integrate ANE For Mobage Native SDK into your existing project after this. Before proceeding to the following section, please make sure that you have Flash Builder installed (please see Setup Requirements for the required version). Here we will use Flash Builder 4.7.

3-1. Create an ActionScript Mobile project in Flash Builder

First, launch Flash Builder and create a new ActionScript Mobile project. From the “File” menu, choose “New”, then “ActionScript Mobile Project”. A dialog named “New ActionScript Mobile Project” should appear. Fill in “HelloMobage” as the project name and click “Next >”.



Create an ActionScript Mobile AIR Project

Choose a name and location for your project

Project name: HelloMobage

Project location

Use default location [ ]

Folder: /Users/testuser/Devel/Adobe Flash Builder 4.6/HelloMobage

Flex SDK version

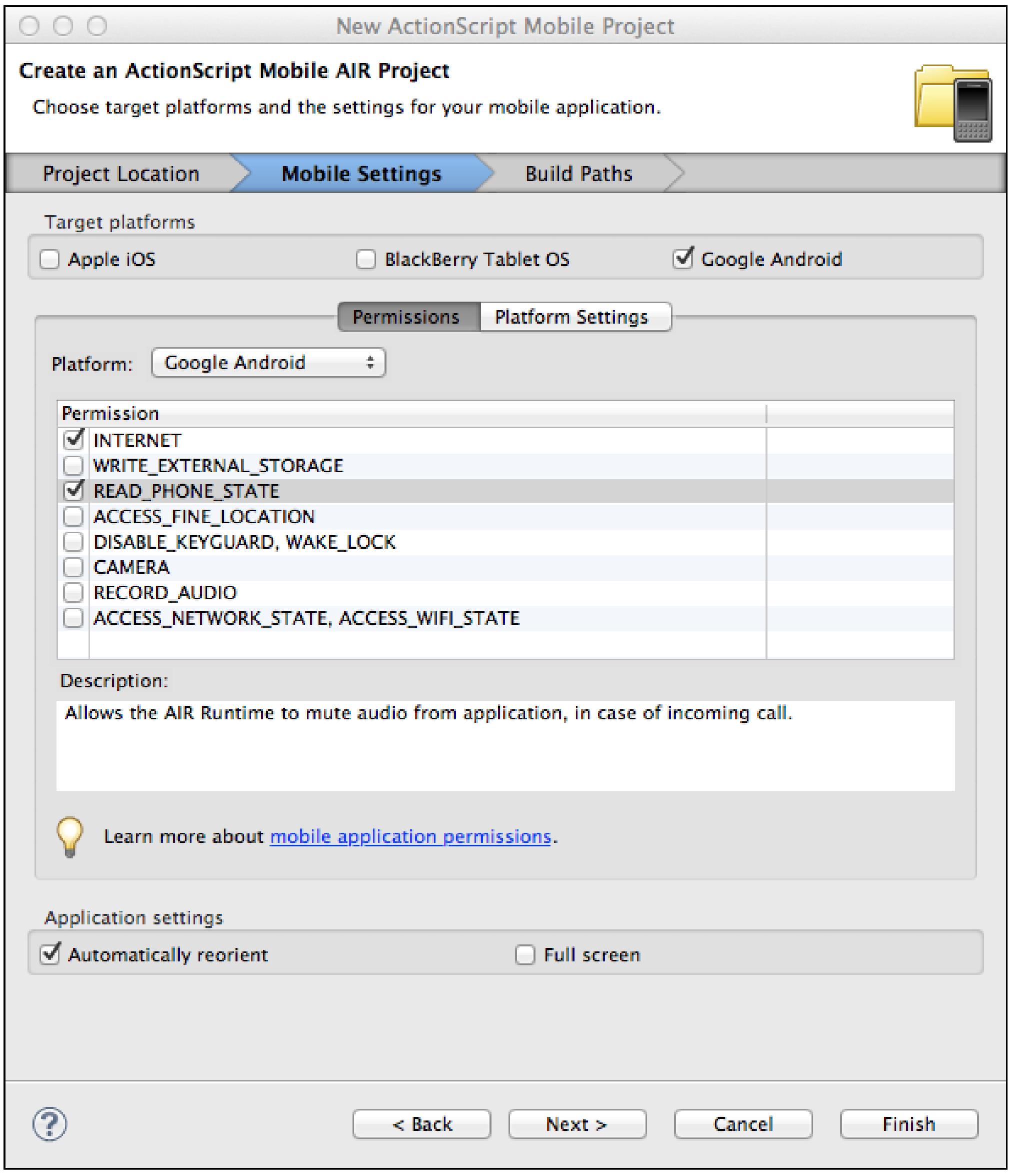
[✓ Use default SDK (currently "Flex 4.6.0")]

Use a specific SDK: Flex 4.6.0

Flex 4.6.0 requires Adobe AIR 3.1

In the next screen, please uncheck “Apple iOS” and “BlackBerry Tablet OS” since we currently only support “Google Android”. Click “Next >”.

At the last stage, we import the ANE For Mobage Native SDK module into the project. Please select the “Native Extensions” tab. Then, click the “Add ANE …” button and browse to where you saved “MobageSDK4AIR.ane”. Press “OK” to add it into your project. The same thing can be done via a project’s “Properties” page, if you ever need to add it to your existing projects. The same screen can be accessed from “Properties” > “ActionScript Build Path” > “Native Extensions” tab.


New ActionScript Mobile Project

Create an ActionScript Mobile AIR Project

Choose target platforms and the settings for your mobile application.

Project Location / Mobile Settings / Build Paths

Target platforms

- [ ] Apple iOS
- [ ] BlackBerry Tablet OS
- [x] Google Android

Permissions | Platform Settings

Platform: Google Android

Permission	
✓ INTERNET	
☐ WRITE_EXTERNAL_STORAGE	
✓ READ_PHONE_STATE	
☐ ACCESS_FINE_LOCATION	
☐ DISABLE_KEYGUARD, WAKE_LOCK	
☐ CAMERA	
☐ RECORD_AUDIO	
☐ ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE	

Description:

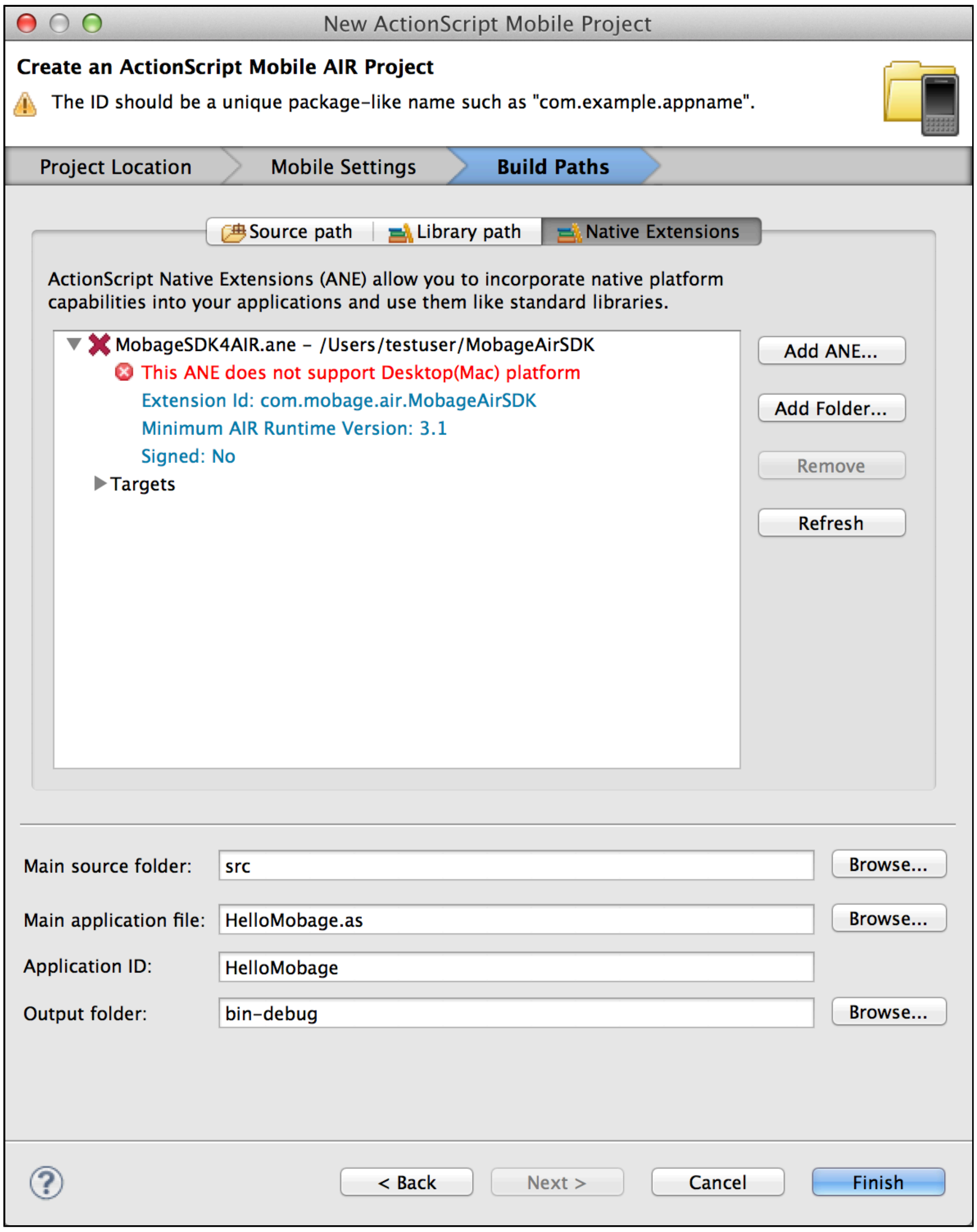
Allows the AIR Runtime to mute audio from application, in case of incoming call.

Learn more about mobile application permissions.

Application settings

- [x] Automatically reorient
- [ ] Full screen

Press “Finish” here to create the project.


New ActionScript Mobile Project

Create an ActionScript Mobile AIR Project

The ID should be a unique package-like name such as "com.example.appname".

ActionScript Native Extensions (ANE) allow you to incorporate native platform capabilities into your applications and use them like standard libraries.

```
▼ MobageSDK4AIR.ane - /Users/testuser/MobageAirSDK
    This ANE does not support Desktop(Mac) platform
    Extension Id: com.mobage.air.MobageAirSDK
    Minimum AIR Runtime Version: 3.1
    Signed: No
  ▶ Targets
```

Main source folder: src

Main application file: HelloMobage.as

Application ID: HelloMobage

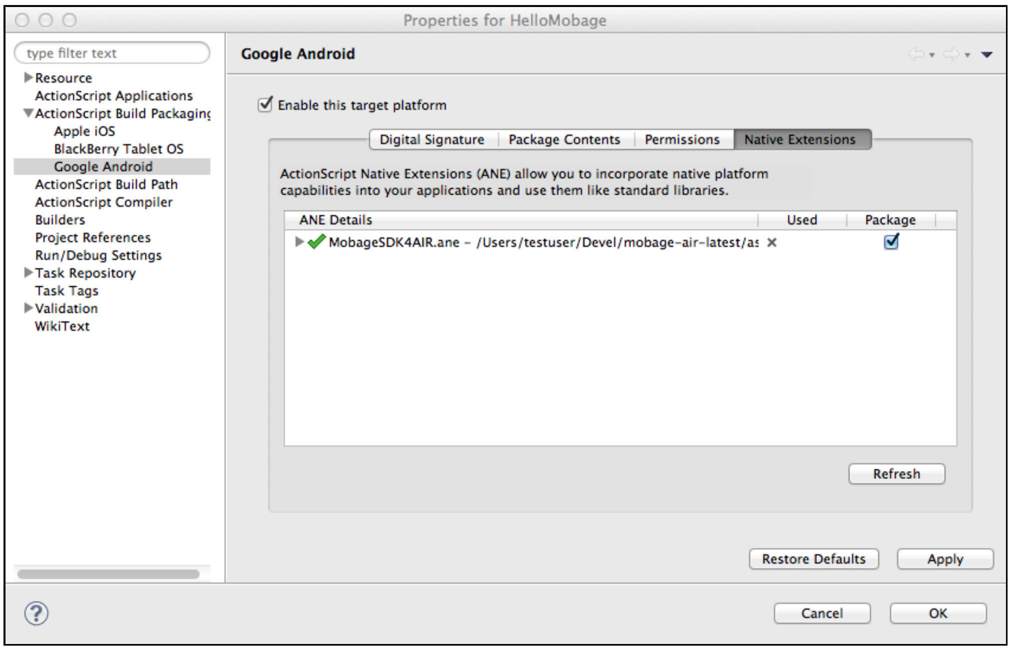
Output folder: bin-debug

Notice that you will see “This ANE does not support Desktop(Mac or Windows) platform” error when importing ANE file to your project since the ANE is not compatible for Desktop (Mac or Windows). You can ignore this error since it is not relevant.

Next, you would need to make sure that the imported ANE file is actually packaged along with your final output application.

You can do this by going to the project’s “Properties” page, expand the “ActionScript Build Packaging” node, and select “Google Android”. Switch to the “Native Extensions” tab, and make sure the check box beside “MobageSDK4AIR.ane” is checked.

You might get a warning but please go ahead and click “Yes”. Finally, click “Apply” then “OK”.



☑ Enable this target platform

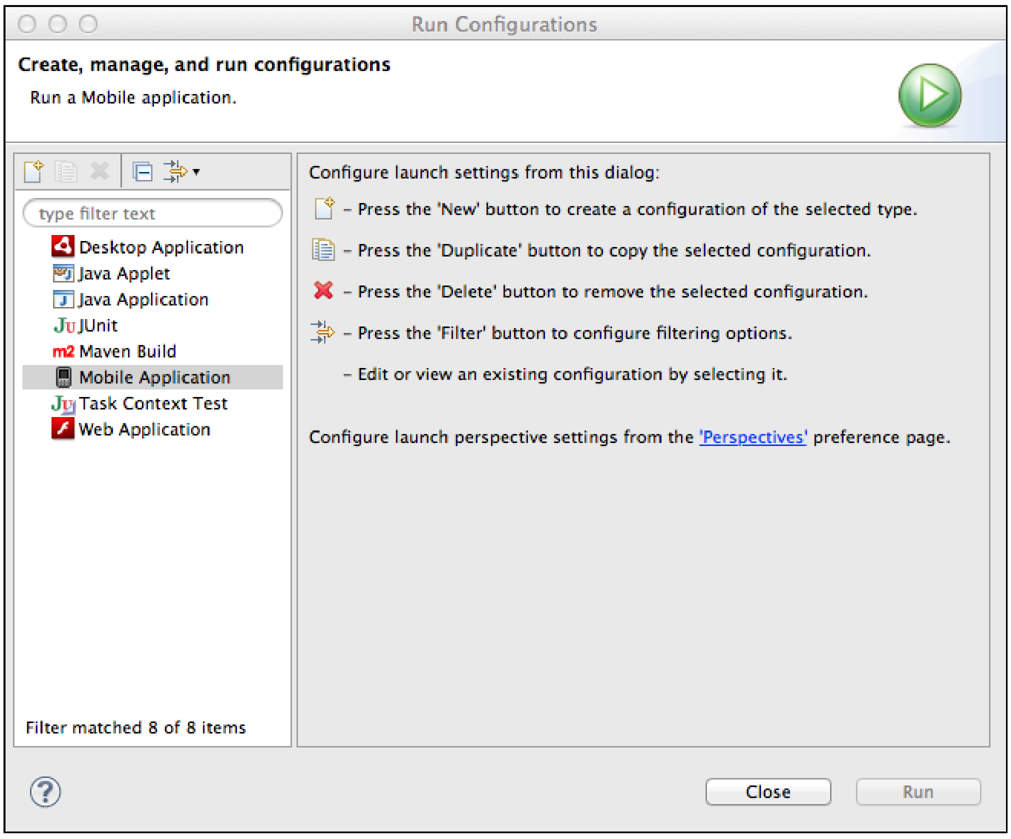
ActionScript Native Extensions (ANE) allow you to incorporate native platform capabilities into your applications and use them like standard libraries.

ANE Details	Used	Package
MobageSDK4AIR.ane – /Users/testuser/Devel/mobage-air-latest/as	✗	☑

3-2. HelloMobage Sample

By default, Flash Builder generates a “HelloMobage.as” file template, which we will later extend to include social functions provided by ANE For Mobage Native SDK.

Now, we should test run it to make sure that everything works fine before going on. From Flash Builder’s “Run” menu, click on “Run Configurations …” and the following dialog should appear.



Create, manage, and run configurations

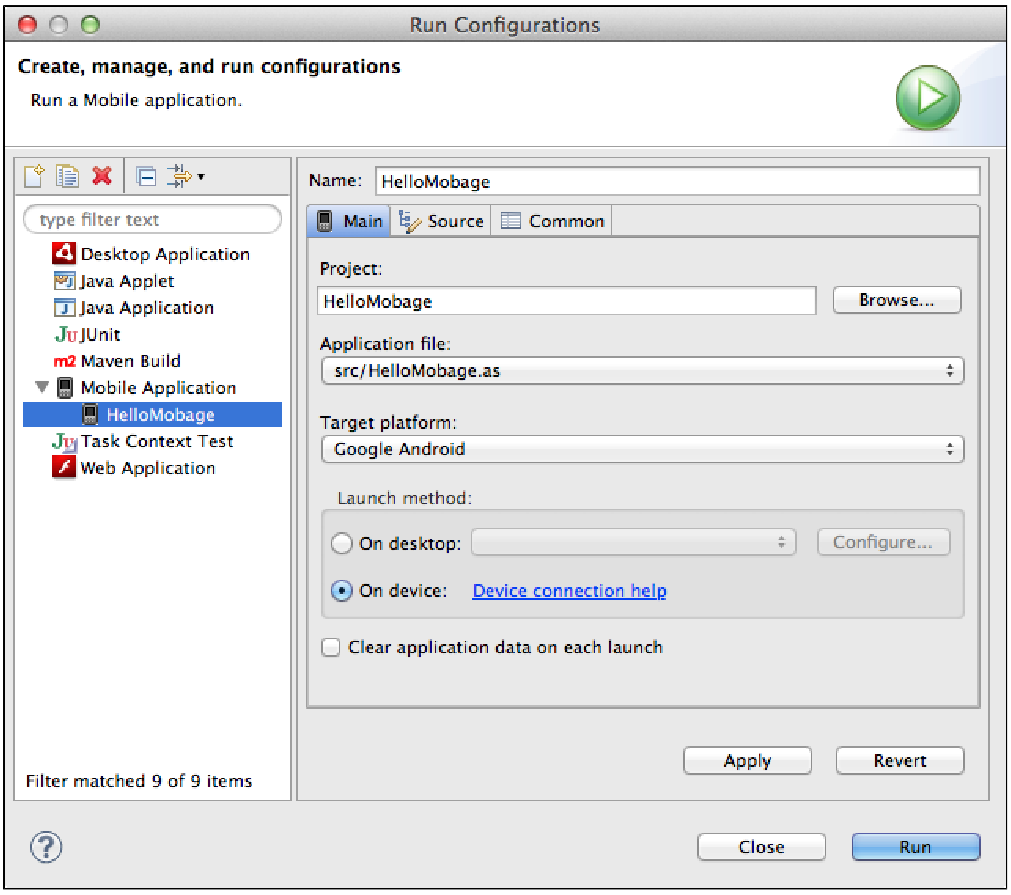
Run a Mobile application.

Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

Select “Mobile Application” from the list on the left, and click the “New launch configuration” button located at the top of the left pane. The right part of the dialog changes as shown in the next screen shot.


Run Configurations

Create, manage, and run configurations

Run a Mobile application.

Name: HelloMobage

Project: HelloMobage

Application file: src/HelloMobage.as

Target platform: Google Android

Launch method:

( ) On desktop:

[✓ On device:] Device connection help

[ ] Clear application data on each launch

Make sure to select "HelloMobage", or whatever you named your project, under "Project". Also, select "On device" so that we can run the sample on a hardware phone (Currently only hardware phone is supported). Click "Apply" and then "Run" to actually run the sample on the phone. You will see a blank screen with white background.

[Note] Click on "Device connection help" to get information on supported Android devices and how to configure your phone. Details on setting up your phone with Flash Builder are out of the scope of this document.

To Run you project on iOS you can follow the same steps above, though selecting the Target platform as Apple iOS.
Notice that you will need to have setup your Digital Signature for iOS to run the application on your device. Please see Flash Builder documents for details.

4. Adding ANE For Mobage Native SDK Social Functions

First of all, please replace the existing content of " HelloMobage.as " with the content of "samples/HelloMobage/src/HelloMobage.as". We will go through the file part-by-part here.

4-1. Importing Modules

First, import the appropriate modules provided by ANE For Mobage Native SDK. There are more to those listed here so please refer to the API reference and import them when necessary.

```
package

{

 import com.mobage.air.ErrorCode;

 import com.mobage.air.Mobage;

 import com.mobage.air.PlatformListener;

 import com.mobage.air.Region;

 import com.mobage.air.ServerMode;

 import com.mobage.air.social.User;

 import com.mobage.air.social.common.People;

  import com.mobage.air.social.common.RemoteNotificationPayload;

  // other necessary imports …
```

4-2. The “ HelloMobage ” Class

The “HelloMobage” class should implement the “PlatformListener” interface so that it can get callbacks from the Mobage platform when necessary.

```
public class HelloMobage extends Sprite implements PlatformListener
```

4-3. Initialization and Login Handling

In the constructor of the class, we call several Mobage ’ s APIs to initialize the Mobage library. First, Mobage.initialize() initializes the connection with the server by specifying the region, the server mode (sandbox or production), the consumer key, the consumer secret and the application ID.

Then, we add a platform listener to handle various login events by registering the “HelloMobage” class in Mobage.addPlatformListener() .

Next, Mobage.checkLoginStatus() checks whether the user needs to log in, if necessary PlatformListener#onLoginRequired() callback function (described later) will be called.

```
public function HelloMobage()
 {
super();


  Mobage.initialize(
   Region.JP,                // region
   ServerMode.SANDBOX,       // serverMode
"sdk_app_id:your_app_id",  // consumerKey
"your_consumer_secret",    // consumerSecret
"your_app_id",             // appId
this);


trace("MobageSDK version: " + Mobage.getSdkVersion());
  Mobage.registerTick(); // To handle events for iOS
  RemoteNotification.setListener(); // Will set listener to receive remote notification
  Mobage.addPlatformListener(this);
  Mobage.checkLoginStatus();


// other initialization code …
 }
```

The “appId”, “consumerKey” and “consumerSecret” in the above code are issued when you register your application at the developer site.

What follow are several callback functions that need to be implemented to handle various login events emitted at different stage on Mobage’s initialization. The function names are rather self-explanatory.

```
public function onLoginError(error :ErrorCode) :void {
 // ...
  }
 public function onLoginCancel() :void {
 // ...
  }
 public function onLoginRequired() :void {
 // ...
  }
 public function onLoginComplete(userId :String) :void {
 // ...
  }
 public function onSplashComplete() :void {
 // ...
  }
public function handleRecieve(payload :RemoteNotificationPayload) :void {
 // ...
  }
```

4-4. Asynchronous event handling (For iOS)

Basically, the ANE For Mobage Native SDK processes events asynchronously, and receives event-handling results through an invoked thread. Therefore, to be able to receive results, we need to call Mobage.register Tick() to register the [MBGPlatform registerTick] method to a timer.

});

```
public function checkOnComplete() :void {
 if(_loginCompleted && _splashCompleted) {
   Mobage.registerTick();
 // ...
  }
 }
```

💡 Above is not needed for Android devices.

4-5. Remote Notification Listener

In order to receive Remote Notification you will need to call RemoteNotification.setListener() in your main function where you called Mobage.addPlatformListener().

4-6. Sample API Call: Obtain User Information

To test that the social functions are working, we call the People.getCurrentUser() API. The People.getCurrentUser() method takes three arguments, i.e. an array of fields to include in the callback, the callback function on success and the callback function on error. This should give you information about the current logged in user.

```
 public function getUserExample() :void {
 var fields :Array = [ User.ID, User.NICKNAME, User.HAS_APP ];
 People.getCurrentUser(fields,
 function(user :User) :void {
 trace("getCurrentUser() succeeded: " + user.nickname);
  },
 function(error :ErrorCode) :void {
 trace("getCurrentUser() failed: " + error);
  });
}
```

[Note] Please be careful when you use anonymous functions like the above callbacks, the “this” pointer might not work inside anonymous functions.

4-7 Editing HelloMobage-app.xml

Finally, in order for your app to communicate with the Mobage platform correctly, you need to modify ${YourProjectName}-app.xml file inside your Flash Builder project. Open it and switch to the "Source" tab. Then, edit or add the following information.

The application ID:

```
<id>jp.mbga.a${YourApplicationId}</id>
```

⚠This application ID is for Japan Region.

For example, if your application ID is 1200xxxx, use " jp.mbga.a1200xxxx " here. Application ID is issued at Mobage Developer Portal.

The version number:

```
<versionNumber>x.y.z</versionNumber>
```

The version number must not be left as 0.0.0, which is the default. Please change it to something like 1.0.0 according to your versioning scheme.

The aspect ratio:

```
<initialWindow>
   <!-- Other settings ... -->
   <aspectRatio>portrait</aspectRatio>
</initialWindow>
```

The Android specific settings:

Modify ${YourAppID} with your application ID.

```
<android>

  <manifestAdditions><![CDATA[
  <manifest android:installLocation="auto">
   <uses-permission android:name="android.permission.INTERNET"/>
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
   <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
   <permission android:name="jp.mbga.a${YourAppID}.permission.C2D_MESSAGE"
android:protectionLevel="signature" />
   <uses-permission android:name="jp.mbga.a${YourAppID}.permission.C2D_MESSAGE" />
   <!-- This app has permission to register and receive message -->
   <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />

    <application>
    <activity android:name="com.mobage.android.activity.MobageDashboardActivity"
                android:windowSoftInputMode="adjustPan"
                android:screenOrientation="portrait">
    </activity>

     <activity android:name="com.mobage.android.activity.MobageProxyActivity"
android:launchMode="singleTask">
          <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="mobage-jp-${YourAppID}" />
          </intent-filter>
    </activity>

     <receiver android:name="com.mobage.android.C2DMBaseReceiver"
android:permission="com.google.android.c2dm.permission.SEND">
          <!-- Receive the actual message -->
          <intent-filter>
             <action android:name="com.google.android.c2dm.intent.RECEIVE" />
             <category android:name="jp.mbga.a${YourAppID}" />
          </intent-filter>
          <!-- Receive the registration id -->
          <intent-filter>
             <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
             <category android:name="jp.mbga.a${YourAppID}" />
          </intent-filter>
    </receiver>

     <service android:name="com.mobage.android.iab.MobageBillingService" />


    <receiver android:name="com.mobage.android.iab.BillingReceiver">
            <intent-filter>
            <action android:name="com.android.vending.billing.IN_APP_NOTIFY" />
            <action android:name="com.android.vending.billing.RESPONSE_CODE" />
            <action
             android:name="com.android.vending.billing.PURCHASE_STATE_CHANGED" />
            </intent-filter>
    </receiver>
   </application>

   </manifest>


   ]]></manifestAdditions>
</android>
```

⭐ See details of configuration for AndroidManifest.xml on Docs Mobage

The iOS specific settings:

For iOS you need to setup Key Chain Group for iOS Entitlements.

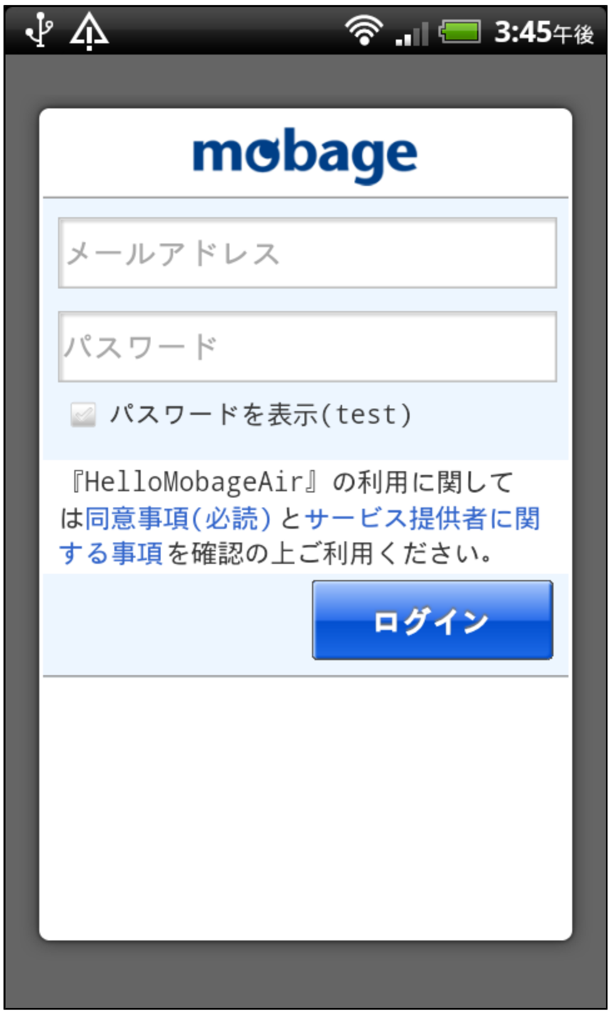
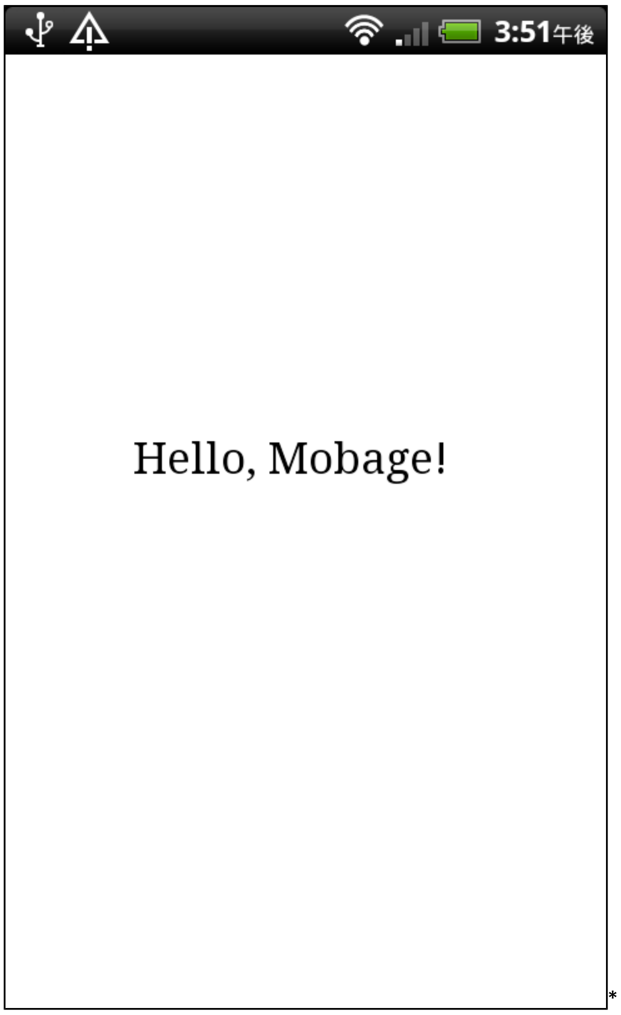
```
 <Entitlements>
<![CDATA[
<key>keychain-access-groups</key>
<array>
<string>$(AppIdentifierPrefix).com.mobage.shared</string>
</array>
]]>
</Entitlements>
```

Replace the $(AppIdentifierPrefix) with your App Identifier Prefix.

4-7 Running the Sample

Now you can run the sample again to see Mobage in action.

💡Notice that the following screen could change depending on the Native SDK version your using.

	
The login screen. This might differ depending on the region of your application. You can log in here with your test account.	The welcome screen.

mobage

メールアドレス

パスワード

パスワードを表示(test)

『HelloMobageAir』の利用に関しては同意事項(必読)とサービス提供者に関する事項を確認の上ご利用ください。

ログイン

Hello, Mobage!

Appendix: ANE For Mobage Native SDK Programming Q&A

Q1. How do I show the balance of virtual currency for a user?

A. You can call the balance button with "com.mobage.air.social.common. getBalanceButton()" method and remove it with "com.mobage.air.social.common. removeBalanceButton() ".
Please notice that there are some difference how the balance button shows on iOS and Android.

Q2. Why the connection is interrupted when I use tools like adb logcat or ddms from the Android SDK?

A. It is found that if the version of adb bundled in Flash Builder and the version of adb from an Android SDK install are different, you cannot use them at the same time. You can create a symbolic link that points to the adb from Android SDK in Flash Builder ’ s installation directory (On a Mac, this will be “ /Applicati ons/Adobe Flash Builder 4.6/sdks/4.6.0/lib/android/bin/adb ”) to solve this problem.

Q3. How to display the Mobage Community Button?

A. From Mobage Native SDK 1.4.3 there is a specific API to display Mobage Community Button.

See "com.mobage.air.social.jp.Service.showCommunityButton()" section in the API reference.

⭐From Mobage Native SDK 1.4.5 this API has become deprecated and you should use "com.mobage.air.social.common.Service.showCommunityUI()" instead to show Mobage Portal.

Q4. How to display the User Profile screen?

A. Please call the API to launch the Mobage Portal Application. For details, please refer to " com.mobage.air.social.common.Service.openUserProfile()" section in the API reference.

Q5. How to display “the Act on Specified Commercial Transactions” and inquiry form?

A. Mobage Native SDK provides an API to display them by using WebView. Please refer to com.mobage.air.social.jp.Service.openDocument() section in the API reference.

Q6. How to use PlatformListener#onLoginCancel()?

A. From Mobage Native SDK 1.1 , the API PlatformListener#onLoginCancel() is added into the SDK. This callback is invoked when the user presses the Back Key (in Android) while the application displays the login Dialog. If you want to terminate the application when the user presses the Back Key in the login screen, please invoke some methods, for example, finish(), in the callback.

Q7. How to setup correct Android package name when exporting for release build?

A. Adobe AIR will automatically add “air.” on front of your package name. This is a known issue and our platform will fix this automatically through Mobage Developers when you upload the apk file.
Although if you want to test Remote Notification on Android before uploading the apk, you will need to remove this "air." from package name or you will not be able to receive notification.

To do this you can follow the following procedure.

1. Build your APK as release build.

2. Use “apktool” to un-package your APK.

```
$ apktool d YourApp.apk
```

3. Open the “AndroidManifest.xml” from the un-packaged APK and modify the following.

```
 <?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="0" android:versionName="0.0.0" android:installLocation="auto"
package="jp.mbga.a${YourAppID}"
xmlns:android="http://schemas.android.com/apk/res/android">

<application>
...
<activity android:theme="@style/Theme.NoShadow" android:label="@string/app_name"
android:name="air.jp.mbga.a${YourAppID}.AppEntry" android:launchMode="singleTask"
android:screenOrientation="user" android:configChanges="keyboardHidden|orientation|screenSize"
android:windowSoftInputMode="stateHidden|adjustResize">
...
</application>
```

- Remove "air." from package.
- Add full path to ".AppEntry" Activity name.

4. Build your modified APK with apktool.

```
$ apktool b YourAppFolder YourApp_unsigned.apk
```

5. Sign your APK with jarsigner.

```
$ jarsigner -verbose -keystore yourkeystore.keystore YourApp_unsigned.apk keyname
```

6. Optimize your APK file with zipalign.

```
$ zipalign -v 4 YourApp_unsigned.apk YourApp.apk
```

Your application should now be able to transmit the device token to Mobage platform for remote notification and you will be able to upload for submission.