MobFox Adobe-Air plugin

1. The **MobFoxPlugin.ane** is the native extension you need to incorporate in your application. Download it, and save it on your computer. You can also use the "**MobFoxDemo**" demo application as an example app to start from.

2. In your "**<Project-name>-app.xml**" file, make sure the following lines are added:

```
<android>
    <colorDepth>16bit</colorDepth>
    <manifestAdditions><![CDATA[
        <manifest android:installLocation="auto">
            <uses-permission android:name="android.permission.INTERNET" />
            <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" />
            <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
            <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
            <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

            <application android:enabled="true">
                <meta-data
                    android:name="com.google.android.gms.version"
                    android:value="@integer/google_play_services_version" />

                <activity
                    android:name="com.mobfox.sdk.interstitialads.InterstitialActivity"
                    >
                </activity>

                <activity
                    android:name="com.mobfox.air.plugin.MyActivity"
                    >
                </activity>

            </application>
        </manifest>

    ]]></manifestAdditions>
</android>

<iPhone>
    <InfoAdditions><![CDATA[
          <key>UIDeviceFamily</key>
          <array>
              <string>1</string>
              <string>2</string>
          </array>
          <key>NSAppTransportSecurity</key>
          <dict>
              <key>NSAllowsArbitraryLoads</key>
              <true/>
          </dict>
          <key>UIRequiresPersistentWiFi</key>
              <string>NO
              </string>
    ]]></InfoAdditions>
    <requestedDisplayResolution>high</requestedDisplayResolution>
</iPhone>

<extensions>
    <extensionID>com.mobfox.air.plugin</extensionID>
</extensions>
```

3. In the view you want to put your ads in, add the following script into the **mxml** file:

```
<fx:Script>
    <![CDATA[
          import com.mobfox.air.plugin.MobFoxPluginController;
          import com.mobfox.air.plugin.events.MobFoxPluginEvent;

          private var controller:MobFoxPluginController = MobFoxPluginController.instance;

          // Put your inventory id here:
          private var myBannerHash:String       = "8769bb5eb962eb39170fc5d8930706a9";
          private var myInterstitialHash:String = "267d72ac3f77a3f447b32cf7ebf20673";
          private var myNativeHash:String       = "80187188f458cfde788d961b6882fd53";

          private var clickUrl:String = "";

          //============================================================

          private function init():void {
              // add listeners for the ad events:
```

```
            controller.addEventListener( MobFoxPluginEvent.BANNER_ERROR, onBannerError );
            controller.addEventListener( MobFoxPluginEvent.BANNER_READY, onBannerReady );
            controller.addEventListener( MobFoxPluginEvent.BANNER_CLOSED, onBannerClosed );
            controller.addEventListener( MobFoxPluginEvent.BANNER_CLICKED, onBannerClicked );
            controller.addEventListener( MobFoxPluginEvent.BANNER_FINISHED, onBannerFinished );

            controller.addEventListener( MobFoxPluginEvent.INTERSTITIAL_READY, onInterstitialReady );
            controller.addEventListener( MobFoxPluginEvent.INTERSTITIAL_ERROR, onInterstitialError );
            controller.addEventListener( MobFoxPluginEvent.INTERSTITIAL_CLOSED, onInterstitialClosed );
            controller.addEventListener( MobFoxPluginEvent.INTERSTITIAL_FINISHED, onInterstitialFinished );
            controller.addEventListener( MobFoxPluginEvent.INTERSTITIAL_CLICKED, onInterstitialClicked );
            controller.addEventListener( MobFoxPluginEvent.INTERSTITIAL_SHOWN, onInterstitialShown );

            controller.addEventListener( MobFoxPluginEvent.NATIVE_READY, onNativeReady );
            controller.addEventListener( MobFoxPluginEvent.NATIVE_ERROR, onNativeError );
    }

    //============================================================
    // listener functions for ads:

    private function onBannerError( event:MobFoxPluginEvent ):void {
            trace( "onBannerError" );
            controller.showToastText("onBannerError");
    }

    private function onBannerReady( event:MobFoxPluginEvent ):void {
            trace( "onBannerReady" );
            controller.showToastText("onBannerReady");
    }

    private function onBannerClosed( event:MobFoxPluginEvent ):void {
            trace( "onBannerClosed" );
            controller.showToastText("onBannerClosed");
    }

    private function onBannerClicked( event:MobFoxPluginEvent ):void {
            trace( "onBannerClicked" );
            controller.showToastText("onBannerClicked");
    }

    private function onBannerFinished( event:MobFoxPluginEvent ):void {
            trace( "onBannerFinished" );
            controller.showToastText("onBannerFinished");
    }

    //------------------------------------------------------------

    private function onInterstitialReady( event:MobFoxPluginEvent ):void {
            trace( "onInterstitialReady" );
            controller.showToastText("onInterstitialReady");

            controller.showInterstitial();
    }

    private function onInterstitialError( event:MobFoxPluginEvent ):void {
            trace( "onInterstitialError" );
            controller.showToastText("onInterstitialError");
    }

    private function onInterstitialClosed( event:MobFoxPluginEvent ):void {
            trace( "onInterstitialClosed" );
            controller.showToastText("onInterstitialClosed");
    }

    private function onInterstitialFinished( event:MobFoxPluginEvent ):void {
            trace( "onInterstitialFinished" );
            controller.showToastText("onInterstitialFinished");
    }

    private function onInterstitialClicked( event:MobFoxPluginEvent ):void {
            trace( "onInterstitialClicked" );
            controller.showToastText("onInterstitialClicked");
    }

    private function onInterstitialShown( event:MobFoxPluginEvent ):void {
            trace( "onInterstitialShown" );
            controller.showToastText("onInterstitialShown");
    }

    //------------------------------------------------------------

    private function onNativeReady( event:MobFoxPluginEvent ):void {
            trace( "onNativeReady" );

            var lines:String = event.eventData;
            var params:Array = lines.split("|");
            var i:uint, tot:uint;
            tot = params.length;
            for (i=0;i<tot;i++)
            {
```

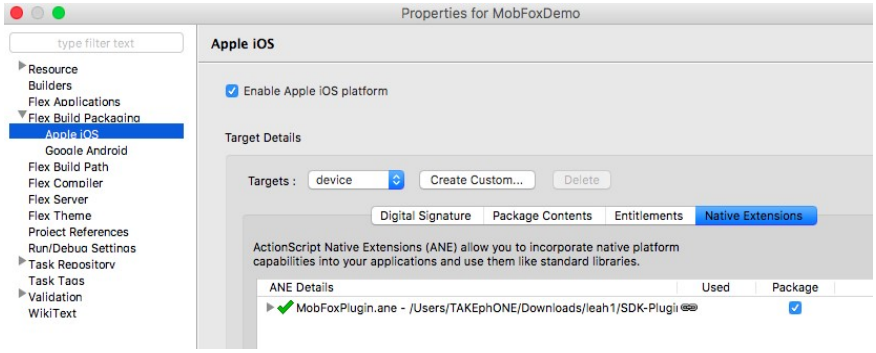
```
                                var line:String = params[i];

                                if (line.substring(0,10)=="<Headline>")
                                {
                                        nativeTitle.text = line.substring(10);
                                }
                                if (line.substring(0,13)=="<Description>")
                                {
                                        nativeBody.text = line.substring(13);
                                }
                                if (line.substring(0,14)=="<IconImageUrl>")
                                {
                                        var iconPictLdr:Loader = new Loader();
                                        iconPictLdr.load(new URLRequest(line.substring(14)));
                                        iconPictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, iconImgLoaded);
                                        function iconImgLoaded(event:Event):void
                                        {
                                                nativeIcon.source = iconPictLdr.content;
                                        }
                                }
                                if (line.substring(0,14)=="<MainImageUrl>")
                                {
                                        var mainPictLdr:Loader = new Loader();
                                        mainPictLdr.load(new URLRequest(line.substring(14)));
                                        mainPictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, mainImgLoaded);
                                        function mainImgLoaded(event:Event):void
                                        {
                                                nativeMain.source = mainPictLdr.content;
                                        }
                                }
                                if (line.substring(0,10)=="<ClickUrl>")
                                {
                                        clickUrl = line.substring(10);
                                }
                        }
                }

                private function onNativeError( event:MobFoxPluginEvent ):void {
                        trace( "onNativeError" );
                        controller.showToastText("onNativeError");
                }

                //============================================================
                // These are example entry points to start the various ad types:

                private function onCheckboxChanged( event:Event ):void {
                        var value:Boolean = useLocationCheckbox.selected;
                        trace( "Set checkbox to:", value );

                        controller.showToastText("Set checkbox to:"+value);

                        controller.setUseLocation(value);
                }

                private function onShowBannerPressed( event:Event ):void {
                        trace( "Show BANNER" );
                        controller.showToastText("onShowBannerPressed");

                        controller.createBanner(myBannerHash, 0, 50, 320, 50);
                }

                private function onShowInterstitialPressed( event:Event ):void {
                        trace( "Show INTERSTITIAL" );
                        controller.showToastText("onShowInterstitialPressed");

                        controller.createInterstitial(myInterstitialHash);
                }

                private function onShowNativePressed( event:Event ):void {
                        trace( "Show NATIVE" );
                        controller.showToastText("onShowNativePressed");

                        controller.createNative(myNativeHash);
                }

                private function onNativePressed( event:Event ):void {
                        trace( "Native ad pressed" );
                        if (clickUrl.length>0)
                        {
                                var urlReq:URLRequest = new URLRequest(clickUrl);
                                navigateToURL(urlReq);
                        }
                }

        ]]>
</fx:Script>
```

4. Right-click your project entry in the left pane in **Adobe Flash Builder 4.7**, and select "**properties**".

3. Select **"Flex Build Path"**, and go to the **"Native Extensions**" tab. If you already see "**MobFoxPlugin.ane**" in the list – select it and "**Remove**" it. Now click "**Add ANE...**" and browse to the location where you saved the "**MobFoxPlugin.ane**" file. Ignore the **"X"** near the added file – it is there because the extension does not support desktop apps – only mobile. Click "**OK**" and get out.

4. Right-click the project in the left pane again, and select "**properties**" again, but this time expand the **"Flex Build Packaging"** line, and repeat the following for both "**Apple iOS**" and "**Google Android**" lines.

5. Click "**Apple iOS**", and go to the "**Native Extensions**" tab. Click on the checkbox in the "Package" column to get to the following state (you may have to click it several times):



6. Click **"Apply",** then "**OK**".

7. Repeat steps 5,6 for "**Google Android**".

8. Clean and build the project – you should be ready to go.