MYAPPSNIPPET

BORN TO MOBILIZE THE FUTURE

Thanks for your interest in our extensions. Please follow us on Twitter and FaceBook to get the latest news and also you can keep yourself updated from http://myappsnippet.com/update-manager/

You will find the following topics in this document:

The AS3 API

MyAR ANE V3.0 is based on MetaioSDK V6.0.2 and you can have full access to all the features that MetaioSDK supports through its AREL solution. All you have to do to open the AR camera in your AS3 projects is as follow. This extension supports Android, Android 64-bit, iOS 32-bit and iOS 64-bit.

```
import com.doitflash.air.extensions.AR.MyAR;
import com.doitflash.air.extensions.AR.MyAREvent;

var _ex:MyAR = new MyAR();
_ex.stage = stage;
_ex.fixAirSDKOrientsBug(stage.autoOrients);
_ex.addEventListener(MyAREvent.COMMUNICATION, onCommunication); // A listener to listen to messages coming from AREL side
_ex.addEventListener(MyAREvent.STATUS, onStatus); // A listener to know about the AR status: MyAR.AR_STARTED / MyAR.AR_FINISHED

var isSupported:Boolean = _ex.isSupported(); // isSupported() method MUST be always called after initializing the extension
trace("is Supported = ", isSupported);

_ex.startAR("/METAIO_AR_ANE_demo/AREL_FILES/index.xml", true); // returns false if the file is not found
// _ex.finishAR(); // call this method to close the AR camera and return back to flash

function onCommunication(e:MyAREvent):void
{
   // send a message from AREL to flash like: arel.Media.openWebsite("flash://MY_MESSAGE_FROM_JS")
   trace("message from AREL = " + e.param);

   // on the AREL side, you may wish to open a URL in device's browser. For this reason, call:
   // arel.Media.openWebsite("http://www.myappsnippet.com/")

   // you can also call functions on the AREL side from flash, like this:
   _ex.toAREL("javascript:myjavascriptfunc('param 1 from Flash!', 'param 2 from Flash!')"); // myjavascriptfunc is a function you may create on the AREL side
}

function onStatus(e:MyAREvent):void
{
   trace("AR Status = " + e.param);
}
```

The initialization code for your AR experience to begin is very simple as you can see above but let me point out to a few details.

1) The method "fixAirSDKOrientsBug" is a workaround solution we came up with to hide AirSDK 16 and 17 rotation bug in iOS 64-bit UIViewControler. We have reported this problem to adobe and they are aware of it and hopefully, when they fix the problem, all you have to do is to remove this method but until then, call this method as you see and your app won't mess up when returning from AR camera on iOS devices.

2) The "MyAREvent.COMMUNICATION" listener is your best friend when trying to communicate between your AREL code and flash. You will be able to send string messages from AREL to flash and on the other hand, you will be able to call JavaScript functions from flash directly. What you have to remember is that those JS functions you call from flash cannot return any values like other regular functions. Instead you must send a message back to flash. Read comments above.

3) Start the AR camera by calling "_ex.startAR("/METAIO_AR_ANE_demo/AREL_FILES/index.xml", true);". What you do here is to tell the extension where it can find the starter xml file for the AREL to be initialized. Your AREL contents must be available in sdcard "File.documentsDirectory" prior to calling this method. Our job is not to teach you how AREL works, you can learn about AREL

on Metaio’s website directly. **https://dev.metaio.com/arel/overview/** but you can use our sample projects or AREL templates to learn how things work around easier and faster. In simple words, you can use AREL to load your AR content and fully engage with it. **You will not need to use any flash 3D engines at all and you will not mess around with flash Stage3D in anyway**. You actually don’t have to know 3D and openGL programming! AREL is here to take care of everything through a simple JavaScript API.

Air manifest .xml file

Before being able to run your project, you need to carefully setup your air manifest .xml file. (Check out Metaio website for more details if you need to) on Android side, you may need the following permissions, based on your AR content. For example if you are not using GPS locators in your AREL project, you won't need "ACCESS_FINE_LOCATION" permission. But here, I will list them all.

```
  <uses-permission android:name="android.permission.CAMERA" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />

  <!-- This is only required for Cloud or Visual Search applications -->
  <uses-permission android:name="android.permission.INTERNET" />

  <!-- These permissions are only needed for debugging -->
  <uses-permission android:name="android.permission.SET_DEBUG_APP" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

  <!-- Required OpenGLES 2.x -->
  <uses-feature android:glEsVersion="0x00020000" android:required="true" />

  <!-- Camera is always required -->
  <uses-feature android:name="android.hardware.camera.any" android:required="true" />
  <uses-feature android:name="android.hardware.camera.autofocus" android:required="false" />

  <!-- Only required for location based applications -->
  <uses-feature android:name="android.hardware.location" android:required="false" />

  <!-- Only required by tracking types such as SLAM, GPSCompass etc.. -->
  <uses-feature android:name="android.hardware.sensor.accelerometer" android:required="false" />
  <uses-feature android:name="android.hardware.sensor.compass" android:required="false" />
  <uses-feature android:name="android.hardware.sensor.gyroscope" android:required="false" />

<application android:hardwareAccelerated="true" android:allowBackup="true">
		<activity android:hardwareAccelerated="false">
			<intent-filter>
				<action android:name="android.intent.action.MAIN" />
				<category android:name="android.intent.category.LAUNCHER" />
			</intent-filter>
			<intent-filter>
				<action android:name="android.intent.action.VIEW" />
				<category android:name="android.intent.category.BROWSABLE" />
				<category android:name="android.intent.category.DEFAULT" />
			</intent-filter>
		</activity>

		<activity android:name="com.doitflash.ar.ARELViewActivity"
android:configChanges="fontScale|keyboard|keyboardHidden|locale|mnc|mcc|navigation|orientation|screenLayout|screenSize|smallestScreenSize|uiMode|touchscreen"
android:theme="@style/Theme.FullScreen" />

		<meta-data android:name="MetaioLicenseString" android:value="d2Zd0H/H5juW9CQ+DYUJXpCWPXzm3MqHhEparJCJkQo=" />
</application>
```

The above settings also includes "**<meta-data android:name="MetaioLicenseString" android:value="d2Zd0H/H5juW9CQ+DYUJXpCWPXzm3MqHhEparJCJkQo=" />**" which is actually your Metaio license key. You must register in Metaio as a developer http://my.metaio.com and get your own license so the Metaio watermark will be removed from your app.

Now, for the iOS part of your manifest, you should set it up like this:

```
<InfoAdditions>
        <![CDATA[<key>MinimumOSVersion</key>
        <string>6.1</string>
        <key>UIStatusBarStyle</key>
        <string>UIStatusBarStyleBlackOpaque</string>
        <key>UIRequiresPersistentWiFi</key>
        <string>NO</string>

        <key>MetaioLicenseString</key>
        <string>MhYNCvoRL6E/PBjhl7Q8+PslosNzVH0FOrJEqyJfbAw=</string>

        <key>UIPrerenderedIcon</key>
        <true />
        <key>UIDeviceFamily</key>
        <array>
                <!-- iPhone support -->
                <string>1</string>
                <!-- iPad support -->
                <string>2</string>
        </array>]]>
</InfoAdditions>
```

It's important to set the "MinimumOSVersion" to at least "6.1" and again, you need to set your Metaio license key as shown above in red.

IMPORTANT: from V3.2 and above MyAR .ane is using google "**android-support-v13.jar**" so you have to download our commonDependencies.ane too and add it to your project. https://github.com/myflashlab/common-dependencies-ANE

You might think that everything is now ready for you to compile your project... sorry to disappoint you but there are still a few more tasks to be done before you can compile your .apk or .ipa! Nothing complicated though. Read the next section and you'll be ready.

Setup the Android, iOS compilation

To be able to compile for android, as an Air developer you are always dependent on Flex/Air SDK but unfortunately Adobe has not yet updated all its android dependencies to the latest versions of Android SDK and we must do this job ourselves. It's just a matter of copying a file from Android SDK to overwrite the current one available in Air SDK (Air SDK 17 built 124 is the latest SDK we have right now maybe in future Air SDK versions, adobe will have updated this, but make sure to do these steps if you face compile errors.)

For Android:
1) Install Android SDK if you don't have it already. https://developer.android.com/sdk/installing/index.html?pkg=tools
2) Update the SDK to make sure you have the latest android build tools. We are using android build tools V19.0.1
3) When done, close your android SDK manager and go to the folder where you have the Android SDK installed.
4) Go to folder "\build-tools\19.0.1\lib" and copy the "dx.jar" file.
5) Now open your Flex-Air SDK and go to folder "\lib\android\bin" in which there is an older version of dx.jar
6) Save a backup from the old dx.jar if you wish and then paste the new dx.jar you copied from Android there.
It's done; get back to your project and compile your .apk it should build your .apk with no problem now.

For iOS:
1) Download Metaio SDK for iOS and install it. You can see download links after you signup as a dev http://my.metaio.com/
2) You need a Mac to be able to compile your .ipa (there's a known bug from Adobe when trying to compile .ipa using third-party frameworks as an extension. Yet, we have found a solution to be able to do it on a windows, but we need to be sure about this approach first before inserting it into this documents, for now, just use a Mac and you'll be fine. If compiling on a windows is your only solution, please contact our support and we'll help you).
3) After installing the iOS SDK, browse its folder and find the framework named "metaioSDK.framework" and copy it.
4) You need to add this framework to your Air SDK where you have the default frameworks so go to your Air SDK in folder "\lib\aot\stub" in which you will find all the default iOS frameworks. All you have to do is to add the "metaioSDK.framework" to this folder.
It's done; now, you can compile your .ipa with no problem.

Privacy and Policy

Considering that 'MyAR' extension is built on MetaioSDK, all the rules with the usage of this 'technology' remains with Metaio. You may read more about their rules here: http://www.metaio.com/metaio-store/ and http://www.metaio.com/privacy-policy/

On the other hand, developing and maintaining the Air Native Extension of this technology is provided by MyFlashLab team and the usage rules of the 'MyAR extension' remains with the MyFlashLab team. You may read more about our usage rules here: http://myappsnippet.com/augmented-reality-adobe-air-native-extension/ and http://www.myappsnippet.com/AR/privacy-policy/

Regards,
MyFlashLab Team