CoreMobile Native Extension

For support, contact support@milkmangames.com

Before you begin:

To View full AS3 documentation, see 'docs/as3docs/index.html'.

This extension requires the AIR 16.0 SDK or higher, which you can get at http://www.adobe.com/devnet/air/air-sdk-download.html ***.***

Review 'example/CoreMobileExample.as' for a sample application.

Note that CoreMobileExample.as is a Document Class. If you're a Flash Professional user and don't know how to use a document class, see the appendix "Using the Example Class in Flash CS6" at the end of this guide.

1. Install the AIR 16.0 (or later) SDK in your IDE

The extension requires the AIR 16.0 SDK or later. You can download the latest AIR SDK from http://www.adobe.com/devnet/air/air-sdk-download.html. If you haven't already installed the AIR 16.0 (or later) SDK for your Flash or Flash Builder IDE, follow the instructions below:

Enabling the AIR 16.0 or later SDK in Flash Professional CS6+:

1. Unzip the AIR 16.0 or later SDK package to a location on your hard drive.
2. Launch Flash Professional.
3. Select Help > Manage AIR SDK...
4. Press the Plus (+) Button and navigate to the location of the unzipped AIR SDK
5. Press OK
6. Select File > Publish Settings
7. Select the latest AIR SDK for iOS from the 'Target' Dropdown menu

Enabling the AIR 16.0 or later SDK in Flash Builder 4.6+- Windows:

1. Unzip the AIR 16.0 or later SDK package to a location on your hard drive.
2. Close Flash Builder.
3. Locate the Flash Builder SDK directory.   On the PC, usually c:\Program Files\Adobe\Adobe Flash Builder 4.6\sdks .
4. Make a copy of the current Flex SDK directory, and give it a descriptive name. For instance, copy the "4.6.0" SDK folder inside /sdks and name the copy "4.6.0_AIR16".
5. Copy and paste the contents of the new AIR SDK on top of the 4.6.0_AIR16 directory. Accept all changes.
6. Edit the flex-sdk-description.xml file inside the new directory, and change the value of the <name> tag to 'Flex 4.6.0 (AIR 16.0)'.
7. Open Flash Builder and choose Project > Properties > Flex Compiler > Configure Flex SDKs.
8. Press 'Add' and navigate to the new folder location.

Enabling the AIR 16.0 or later SDK in Flash Builder 4.6+- Mac:

1. Copy the contents AIR 16.0 or later SDK package to a location on your hard drive.
2. Close Flash Builder.
3. Locate the Flash Builder SDK directory. On the Mac, it is usually /Applications/Adobe Flash Builder 4.6/sdks/. On the PC, c:\Program Files\Adobe\Adobe Flash Builder 4.6\sdks .
4. Create a new folder inside the SDK folder, called AIR16SDK and copy the contents of the SDK package into it.
5. Open the Terminal, and merge the AIR 16.0 or later SDK files into your current SDK directory:

   ```
   sudo cp -Rp /Applications/Adobe\ Flash\ Builder\ 4.6/sdks/AIR35SDK/ /Applications/Adobe\ Flash\
   Builder\ 4.6/sdks/4.6.0/
   ```
6. Edit the flex-sdk-description.xml file inside the new directory, and change the value of the <name> tag to 'Flex 4.6.0 (AIR 16.0)'.
7. Open Flash Builder and choose Project > Properties > Flex Compiler > Configure Flex SDKs.
8. Press 'Add' and navigate to the new folder location.Press 'Add' and navigate to the new folder location.

2. Include the Extension Library

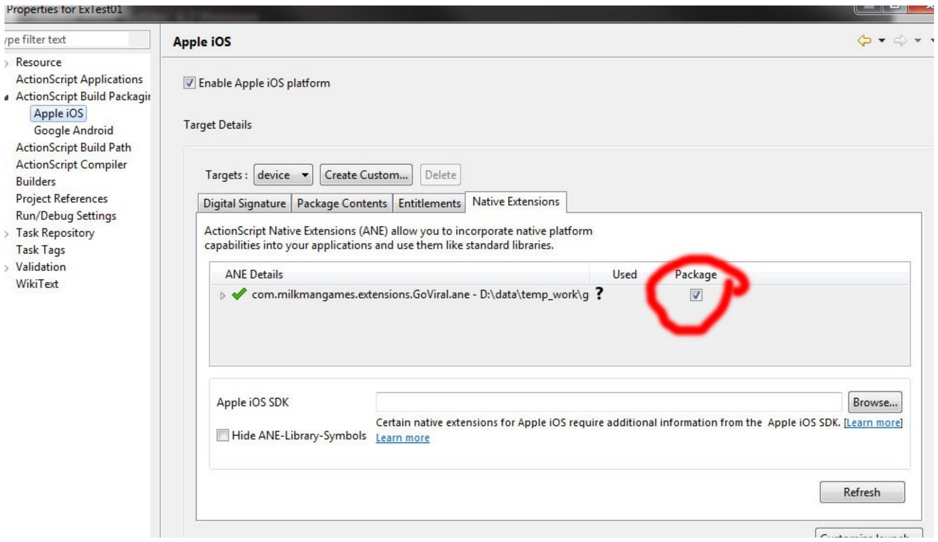
Add the com.milkmangames.extensions.CoreMobile.ane library to your project. This is located in the /extension folder of the extension package zip file.

In Flash Professional CS6 or Higher:

1. Create a new project of the type 'AIR for iOS' (or AIR for Android)
2. Select File > Publish Settings...
3. Select the wrench icon next to 'Script' for 'Actionscript Settings'
4. Select the Library Path tab
5. Press the 'Browse for Native Extension (ANE) File' button and select the com.milkmangames.extensions.CoreMobile.ane file. Press OK.
6. Select the wrench icon next to 'Target' for Player Settings
7. *(For Android Only)* Select the 'Permissions' tab, and enable INTERNET, ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE and WAKE_LOCK permissions.
8. *(For Android Only)* **Check the 'Manually manage permissions and manifest additions for this app' box**.
9. Make sure you're using the AIR 16.0 SDK or higher per the instructions in Section 3.
10. Select File>Publish Settings...
11. For 'Target' Select 'AIR 16.0 (or higher) for iOS' (Or Android)

In Flash Builder 4.6+:

1. Go to *Project Properties*
2. Select *Native Extensions*
3. Choose *Add ANE...* and navigate to the com.milkmangames.extensions.CoreMobile.ane file
4. Make sure the 'package' box is checked for your target, under Actionscript Build Packaging:



[✓] Enable Apple iOS platform

Targets : device

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

ANE Details	Used	Package
✓ com.milkmangames.extensions.GoViral.ane - D:\data\temp_work\g	?	✓

Apple iOS SDK

Certain native extensions for Apple iOS require additional information from the Apple iOS SDK. [Learn more]

[ ] Hide ANE-Library-Symbols Learn more

5. Make sure you're using the AIR 3.5 SDK (or higher) per the instructions in Section 3.

3. Update Your Application Descriptor

You'll need to be using the AIR 16.0 SDK or higher, and include the extension in your Application Descriptor XML. For an example, see **'example/app.xml'**.

Set the AIR Version and Extension ID

1. Set your AIR SDK to 16.0 (or higher) in the app descriptor file:

```
<application xmlns="http://ns.adobe.com/air/application/16.0">
```

2. Include a link to the extension in the descriptor:

```
<extensions>
<extensionID>com.milkmangames.extensions.CoreMobile</extensionID>
</extensions>
```

For Android Only: Update the Android ManifestAdditions

(For Android Only): The extension requires a few android-specific permissions and activities to be defind in your XML file. Edit your application.xml's '<android>' section to look like thie one below:

```
<android>
<manifestAdditions><![CDATA[
<manifest android:installLocation="auto">
<uses-sdk android:minSdkVersion="9" android:targetSdkVersion="19" />

<!-- These permissions are required by CoreMobile (to check internet connection) -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<!-- The WakeLock permission is required to use local notifications. -->
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<!-- The Vibrate permission is required to use the vibration, and/or local notifications. -->
<uses-permission android:name="android.permission.VIBRATE"/>

<!-- this permissions is required by CoreMobile, if you use notifications and want them to
survive a reboot-->
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>

<application>
      <!-- This receiver is required by CoreMobile -->
      <receiver
android:name="com.milkmangames.extensions.android.CMLocalNotificationReceiver"/>

<!-- This receiver is required if you use notifications and want them to survive a reboot →
      <receiver android:name="com.milkmangames.extensions.android.CMBootReceiver">
            <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED"/>
            </intent-filter>
      </receiver>

</application>

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

4. ActionScript Initialization and Basic Setup

The CoreMobile extension can be up and running in a few simple calls. See '**example/CoreMobileExample.as**' for a full example.

1. Import the API Classes:

   ```
   import com.milkmangames.nativeextensions.*;
   import com.milkmangames.nativeextensions.events.*;
   ```

2. First, check to make sure that CoreMobile is running on a supported platform, using the `isSupported()` method. CoreMobile only works on Mobile devices (Android and iOS):

   ```
   if(CoreMobile.isSupported())
   {
         // call this once at app start up, then CoreMobile will be available for use from then on.
         CoreMobile.create();
   }
   else {
         trace("Core Mobile only works on iOS or Android.");
   }
   ```

5. Checking for an Internet Connection

It is often useful to determine whether your device has an active internet connection before performing network functions in your game. CoreMobile allows you to quickly query whether the internet is available, or use an event listener to determine when an internet connection is gained or lost.

Checking if an Internet Connection is Available

You can determine if the device has an active internet connection using the `isNetworkReachable()` function:

```
// returns true if the internet connection is currently working
var networkAvailable:Boolean=CoreMobile.mobile.isNetworkReachable();
```

You can determine if the device is connected via Wifi, Mobile, or neither with the the `getNetworkType()` function:

```
// returns CMNetworkType.NONE, CMNetworkType.MOBILE, or CMNetworkType.WIFI
var networkType:int=CoreMobile.mobile.getNetworkType();

switch(networkType)
{
	case CMNetworkType.NONE:
		trace("no network connectivity");
		break;

	case CMNetworkType.MOBILE:
		trace("on mobile network");
		break;

	case CMNetworkType.WIFI:
		trace("on wifi network");
		break;
}
```

Listening for an Event when the Internet Connection Changes

When the internet connection is lost or reestablilshed, CoreMobile can dispatch a `CMNetworkEvent.NETWORK_REACHABILITY_CHANGED` event. Its `isNetworkReachable` property will tell you the status of the internet connection, and its n`etworkType` property will be one of `CMNetworkType.NONE, CMNetworkType.MOBILE, or CMNetworkType.WIFI`:

```
//get a notificaiton when the internet connection is lost or reestablished...
CoreMobile.mobile.addEventListener(CMNetworkEvent.NETWORK_REACHABILITY_CHANGED, onNetworkChanged);

function onNetworkChanged(e:CMNetworkEvent):void
{
	trace("is internet reachable?: "+e.isNetworkReachable );
	trace("is wifi?: "+*e.networkType==CMNetworkType.WIFI) );
}
```

6. Making the device Vibrate

Vibration can be useful for adding tacticle feedback to a game, or alerting the user to an important event. You can cause the device to vibrate with the `vibrate()` function. If the device does not have a motor, nothing will happen. Optionally, you can pass a duration in seconds for the vibration. (Duration is only acknowledged on the Android platform):

```
// make the phone vibrate for half a second
CoreMobile.mobile.vibrate(0.5);
```

7. Getting Orientation Data from the Gyroscope

CoreMobile provides a simple but powerful interface for reading smooth, accurate orientation data based on the device's gyroscope and other sensors, that produces consistent results on the same scale for both iOS and Android.

CoreMobile monitors the device's built-in gyroscope, in combination with hints from its other available sensors, to compute accurate orientation readings. To conserve battery and system resources, you can enable and disable this monitoring using the `startMonitoringDeviceMotion()` and `stopMonitoringDeviceMotion()` functions.

When monitoring is enabled, you can get the device's current orientation using the `getDeviceMotionData()` method. You can query this method as needed before rendering to update your game's state. This is better than a distinct gyroscope event as it avoids burdening the processor with extra calculations on a separate loop, and avoids adding additional event listeners to performance critical code.

Enabling Motion Listeners

In the example below, we first ensure that monitoring is available (not all devices have gyroscopes), and then start monitoring:

```
if(CoreMobile.mobile.isDeviceMotionAvailable())
{
	CoreMobile.mobile.startMonitoringDeviceMotion();
}
```

Reading Device Orientation

Continuing the above example, the following code demonstrates reading the current orientation (expressed as roll, pitch, and yaw, in radians) during our game's render loop, as well as the quaternion representation of the attitude:

```
function onFrame(e:Event):void
{
	var data:DeviceMotionData=CoreMobile.mobile.getDeviceMotionData();

	// device euler angles
	var roll:Number=data.roll;
	var pitch:Number=data.pitch;
	var yaw:Number=data.yaw;

	// quaternion representation
	var qw:Number=data.qw;
	var qx:Number=data.qx;
	var qy:Number=data.qy;
	var qz:Number=data.qz;

	trace("Orientation angles: "+ roll+", "+ pitch+", "+yaw);
	trace("q=: "+ qw+", "+ qx+", "+qy", "+qz);
}
```

Disabling Motion Listeners

When you are finished with the gyroscope, disable monitoring with the `stopMonitoringDeviceMotion()` method:

```
CoreMobile.mobile.stopMonitoringDeviceMotion();
```

8. Using Native Dialogs

It is often useful to use native modal dialogs to alert the user to an important event, ask a question, collect simple input, or get login information. CoreMobile allows you to implement such dialogs in a consistent, crossplatform way, without having to design and code the UI by hand in ActionScript.

Showing a Confirmation Dialog

You can show a simple modal alert dialog with one line of code using the `showModalConfirmationDialog()` function:

```
CoreMobile.mobile.showModalConfirmationDialog("The Title", "Acknowledge me now!", "OK");
```

Optionally, you can listen for an event to be fired whenever a CoreMobile dialog is dismissed. Each CoreMobile dialog function returns an instance of the `CMDialogDispatcher` object- an ActionScript EventDispatcher that puts out a `CMDialogEvent.MODAL_DIALOG_DISMISSED` event when the dialog closes. This dispatcher has a convenient method called `addDismissListener` that lets you attach the listener function inline, in such a way that it will be weakly linked and automatically removed after its use. The following example expands on the previous one by adding a callback for the dialog's dismissal:

```
CoreMobile.mobile.showModalConfirmationDialog("The Title", "Acknowledge me now!", "OK").
                    addDismissListener( function(e:CMDialogEvent):void

                    {

                          trace("Confirmation dismissed.");

                    });
```

Showing a Question Dialog

You can show a simple modal dialog asking the user to respond to a yes/no question with the `showModalYesNoDialog()` function:

```
CoreMobile.mobile.showModalYesNoDialog("Question..", "Yes or no?", "YES!", "NO!").
                    addDismissListener(function(e:CMDialogEvent):void

                    {

                          if (e.selectedButtonLabel=="YES!") {

                                trace("You said yes.");

                          } else {

                                trace("You said no.");

                          }

                    });
```

Prompting the User for Input

You can show a modal dialog asking the user to enter a line of text with the `showModalInputDialog()` function:

```
CoreMobile.mobile.showModalInputDialog("Your Name", "Who are you?", "OK", "Type name here", "Cancel").
                    addDismissListener(function(e:CMDialogEvent):void

                    {

                          if (e.selectedButtonLabel=="OK") {

                                trace("Your Name: "+e.modalUserInput);

                          } else {
                                trace("You didn't enter a name.");
                        }
                });
```

Prompting the User for Login Credentials

You can show a modal dialog asking the user to enter a username and password with the `showModalCredentialsDialog()` function:

```
CoreMobile.mobile.showModalCredentialsDialog("Login", "Enter name and password:", "OK", "Cancel").
                addDismissListener( function(e:CMDialogEvent):void
                {
                        if (e.selectedButtonLabel=="OK") {
                                trace("Your u/p: "+e.modalUserName+", "+e.modalPassword);
                        } else {
                                trace("You didn't enter credentials.");
                        }
        });
```

Showing a Selection Picker

You can show a Picker dialog asking the user to select an option from a range of choices you provide using the `showPicker()` function:

```
var items:Vector.<String>=new <String>["First", "Second", "Third", "Fourth", "Fifth", "Sixth",
"Seventh", "Eigth", "Ninth", "Tenth", "Eleventh"];

// 'items' is the list of items, defined above, to present in the picker
// '3' is the default index of the item to pre-select
// "Done" and "Cancel" allow you to specify the button labels to use on Android.  On iOS it will use the
// system default buttons for these functions.
//
// On iOS iPads, the picker is presented in a popover bubble.
// 320, 320 is the location in device POINTS (not pixels!) where the popover will originate from.
CoreMobile.mobile.showPicker(items, 3, "Done", "Cancel", 320, 320).
        addDismissListener( function(e:CMPickerEvent):void
        {
                if (e.wasCanceled)  {
                        trace("The picker was canceled!");
                } else {
                        trace("Selected picker item: "+e.selectedItemIndex));
                        trace(" (value is "+(e.items[e.selectedItemIndex]));
                }
        });
```

Canceling a Selection Picker

You can cancel a current picker with the `cancelPicker()` function. You may wish to cancel a picker if the user selects a UI item elsewhere on the screen, depending on your implementation:

```
CoreMobile.mobile.cancelPicker();
```

9. Using Local Notifications

While Push Notifications can be used for sending new information, sales, updates, messages, and multiplayer game events from a server to your users, Local Notifications are useful for sending a message to the current user only after a given delay.

For instance, a game in which the user's energy slowly recharges after one hour, could use a local notification to remind the user when their energy is refilled; or if the user has not loaded your app in 3 days, a Local Notification could prompt them to come back and play some more.

Registering for Notifications (iOS)

If you're targeting iOS, you should “register” for notifications near the very beginning of your application. If the app does not yet have permission to show notifications, it will present the user with a prompt asking for their permissions to show them:

```
CoreMobile.mobile.registerForNotifications();
```

You can check at any time whether notifications have been previously allowed:

```
var isRegistered:Boolean=CoreMobile.mobile.isRegisteredForNotifications();
```

Scheduling a Local Notification

The following example schedules a local notification to be triggered one day from the current time. You assign each local notification a unique integer ID (the first parameter), which you can use to refer to the notification later.

The second parameter is the date the notification should appear, expressed in seconds. Here we are using the `CoreMobile.futureTimeSeconds` convenience function to easily calculate the timestamp for one day from now.

The next three parameters set the title, message, and action label of the notification. (Action labels will appear on iOS devices in certain situations, such as when the notificaiton is visible from the lock screen.)

The last parameter lets you set an arbitrary collection of key value pairs unique to the notificaiton. This information can be retrieved later when the notification is selected, allowing you to send information about the current game state to a future load.

```
var ONE_DAY_NOTIFICATION_ID:int=100;

CoreMobile.mobile.scheduleLocalNotification(
                    ONE_DAY_NOTIFICATION_ID,
                    CoreMobile.futureTimeSeconds(CoreMobile.DAY),
                    "Come back!",
                    "Hey, you should come play some more.",
                    "Play now",
                    {coins:50, name:"Bob",type:"ONE_DAY"});
```

Scheduling a Repeating Local Notification

You may also schedule notifications that will repeat on a regular basis, using the `scheduleRepeatingLocalNotification` method. The format is the same as `scheduleLocalNotification`, but with an additional third parameter, `thenEverySeconds`. You should set this value to the interval at which you want notifications to repeat, using one of the CoreMobile time constants. The following example schedules a repeating local notification, that will fire about one week from now, then every week after that until canceled:

```
var REPEATING_NOTIFICATION_ID:int=200;

CoreMobile.mobile.scheduleRepeatingLocalNotification(
                    REPEATING_NOTIFICATION_ID,
                    CoreMobile.futureTimeSeconds(CoreMobile.WEEK),
                    CoreMobile.WEEK,
                    "Another week!",
                    "One more week has gone by...",
                    "Play now",
                    {coins:30, name:"Alice",type:"REPEATING"});
```

Listening for Local Notification Events

There are two scenarios for receiving local notifications: Either the app was open, and a notification was received (`CMLocalNotificationEvent.LOCAL_NOTIFICATION_RECEIVED`), or the app was closed or in the background, and the user clicked the notification to open the app (`CMLocalNotificationEvent.RESUMED_FROM_LOCAL_NOTIFICATION`.) In either case, you can use the event's `extra` property to read back the special data you sent with the original notification:

```
// this event is dispatched when a local notification occurs while the app is running
CoreMobile.mobile.addEventListener(CMLocalNotificationEvent.LOCAL_NOTIFICATION_RECEIVED,
onNotification);
// this event is dispatched when the app is resumed or started because of a local notification
clickCoreMobile.mobile.addEventListener(CMLocalNotificationEvent.RESUMED_FROM_LOCAL_NOTIFICATION,
onNotification);

function onNotification(e:CMLocalNotificationEvent):void
{
       trace("notified: "+e.title+"/ "+e.message);
       // we can retrieve the extra daya we passed in the scheduleLocalNotificaiton example above.
       var extra:Object=e.extra;
       trace("coins: "+extra.coins);
       trace("name: "+extra.name);
       trace("type: "+extra.type);
}
```

Canceling a Local Notification

In some cases, you may wish to cancel a pending local notification that is no longer valid. In our example above, we set a notification to go off in one day to remind the user to come back and play our game some more. In this case, we would not want the notification to occur if the player starts a new session before the day is over- so we might cancel it, using its ID, as shown below, when the game starts up. (Of course, to keep the process running, you'd need to schedule a new local notification for the next day.)

```
// cancel the pending notification with the id ONE_DAY_NOTIFICATION_ID
CoreMobile.mobile.cancelLocalNotification(ONE_DAY_NOTIFICATION_ID);
```

Canceling All Local Notifications

If you wish to cancel *all* pending local notifications scheduled by your app, use the `cancelAllLocalNotifications` method:

```
// cancel ALL pending local notifications from this app
CoreMobile.mobile.cancelAllLocalNotifications();
```

10. Troubleshooting Common Problems

“Why is my app not working properly on Android?”

- *Carefully reread the 'Update Your Application Descriptor' instructions and make sure you've updated the descriptor with the correct android data. These changes are required for Android.*
- *If you're using Flash Professional version CS6, there is a bug in the IDE can cause your XML changes to be overriden. To avoid this, make sure you publish your app by Selecting File>Publish or Debug>Debug Movie, instead of File > Publish Settings... > Publish to avoid this.*

“What iOS/Android versions does CoreMobile support?”

- *CoreMobile will work on any Android device running Android 2.3 or higher, and any iOS Device running iOS 5 or higher.*

“How do I use the CoreMobileExample.as file in Flash Professional?”

- First, setup the ANE and descriptor in your IDE, as shown in Sections 1-3 above .
- Copy and paste CoreMobileExample.as into the same folder as your .fla. Do not copy and paste its contents on to the timeline. That will not work.
- In Flash properties, under 'Document Class', type 'CoreMobileExample' (no quotes) and press OK.
- CoreMobileExample.as is a complete application. It draws a user interface that lets you test everything CoreMobile can do using buttons. It is not meant to be used “on top of” an existing Flash app (unless you want your game to be covered in gray example buttons!)
- Build and install the application.

“Why doesn't the extension work when I run my swf on my Mac / Windows Computer?”

- *The extension uses features that are built into the iOS and Android operating systems. The extension will only work when you run it on a phone or tablet.*

“I still need help! How can I get technical support?”

- Your purchase comes with free technical support by email – just send us a message at support@milkmangames.com . We're here to help! Please remember that...:
- We're open during United States business hours, excluding U.S. Holidays, Monday-Friday, Pacific Standard Time. We strive to answer all support email within 24 hours (not including weekends and holidays) but usually do so much faster. Remember that we may not be in the same time zone.
- Please remember to mention: which extension you're having a problem with, what IDE you're using (such as 'Flash CS6' or 'Flash Builder 4.6'), and what device you're targeting. If you're experiencing an error message, please specify what that message is.
- ***We don't provide tech support through blog comments, Facebook, or Twitter. Please email us and we'll be happy to help you out!***