Windows Phone mOcean SDK

Developer Getting Started Guide

For Windows Phone SDK Version 8

Table of Contents

Table of Contents 2

What’s New in 3.0 3

Implementation Changes 3

System Requirements 3

Prerequisites 3

Feature List 4

Installing the Ad SDK 5

SDK Integration Methods 5

Building and adding the SDK assemblies 5

User Interface / Layout (Design) 6

Simple Ad Integration 7

Interstitial Ad Integration 8

MASTAdView Customization 9

Customize view appearance 9

Customize ad network properties 9

Location detection 9

Content Updates 10

Detecting Updates and Failures 10

Troubleshooting 10

Ad content loading issues 10

Next Steps 11

# What’s New in 3.0

* New codebase
* The SDK has been redesigned and simplified
* Image and text ads now render in native fields

# Implementation Changes

* The 3.0 SDK is not compatible with the 2.x SDK. Prior to installing the 3.0 SDK remove the 2.x SDK references.

# System Requirements

* Windows Phone 8 development environment (Visual Studio 2012)
* Windows Phone SDK 8
* Applications can target Windows Phone 7.1 or 8

# Prerequisites

This guide does not cover Blackberry development techniques or instructions for using Visual Stuido to develop applications for Windows Phone. Windows Phone developer documentation is available from Microsoft at <http://developer.windowsphone.com/> .

More thorough, complex examples and additional use cases in the sample application distributed with the SDK. Both the sample app and the SDK itself are available in source code form from our Google Code project site at <http://code.google.com/p/mocean-sdk-windows-phone/>.

Additional documentation, information, and other supported platforms on the developer wiki at: <http://developer.moceanmobile.com/Main_Page>.

# Feature List

* **Rich Media ads**

SDK supports displaying MRAID 2 compatible ads.

* **Image/Text ads**

SDK supports displaying image and text ads with native controls.

* **Location auto detect**

SDK can automatically detect user location.

* **User-Agent auto detect**

SDK automatically detects device User-Agent.

* **Internal browser**

SDK contains built-in browser for displaying ads in application.

* **Ad visibility tracking**

SDK automatically detects ads visibility for controlling updates.

* **Logging**

SDK supports logging through delegate callbacks and System.Debug.

# Installing the Ad SDK

## SDK Integration Methods

The SDK can be linked in as a collection of assembly references or including the MASTAdView project and referencing that. Whichever method is best can be determined by the integrating developer/team.

## Building and adding the SDK assemblies

The assemblies (DLL files) can be built from source or obtained through a released archive available from the Google Code page.

# User Interface / Layout (Design)

The first step is deciding where you want to incorporate ads in your application.

The simplest approach is to integrate a small horizontal banner ad into the user-interface (UI). A typical form factor is a 50-pixel tall, full width rectangle which does not crowd the existing UI elements or break the appearance and flow.

# Simple Ad Integration

Implementation can vary depending on existing code style and layout. Windows Phone XAML layouts are a good place to start. It is also possible to place MASTAdView in backing code as with any other control.

XAML fragment from default VS2012 phone application page:

|  |
| --- |
| <!--ContentPanel - place additional content here--> <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">   <StackPanel>  <mast:MASTAdView x:Name="adView" Background="Gray" Zone="88269"   UpdateInterval=”60” HorizontalAlignment="Stretch" Height="50"/>  </StackPanel> </Grid> |

MASTAdView instance creation, configuration, and cleanup

This example shows a few properties of the ad view being set including:

* **Ad zone**: this is used to identify one specific ad placement in your application. Zones are created through the Mocean Mobile UI and target content to ad placements in your application. The zone is required in order to request an ad.
* **Ad update interval**: This configured the time period (in seconds) after which the ad view will retrieve a new ad from the back-end.
* **Horizontal alignment**: Like other Windows Phone controls, this attribute instructs the rendering of the MASTAdView to stretch the full width of the parent. The resulting width is then used as the size\_x parameter sent to the ad server.
* **Height**: Another base control property is used to specify a fixed height for the ad view. This value is used as the size\_y parameter sent to the ad server.

See Also:

* For more code samples examine the Samples application.

# Interstitial Ad Integration

Interstitial ads work much like inline/banner ads except that they are directly displayed and do not have any inline appearance/view. Note however that an interstitial instance can’t be used as a banner and vice versa.

To allow seamless flow with XAML development, the MASTAdViewInterstitial component is available. This component is a superclass to the MASTAdView class and automatically presents the interstitial ad it after it is downloaded.

XAML fragment from default VS2012 phone application page:

|  |
| --- |
| <!--ContentPanel - place additional content here--> <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">  <StackPanel>  <!-- Note that if placing interstitials this way they need  to be collapsed since they won't function as inlinebanner ads. -->  <mast:MASTAdViewInterstitial x:Name="adView" Zone="88269"  Visibility="Collapsed"/>  </StackPanel> </Grid> |

MASTAdView interstitial reference creation, configuration, cleanup and presentation

Note that interstitial ad views created in backing code vs. XAML still require update to be called and can be customized like inline/banner ads.

See Also:

* For more code samples examine the Samples application.
* To automatically close the interstitial after a specified amount of time pass in a delay amount to showInterstitial(int, int).

# MASTAdView Customization

## Customize view appearance

Ad links are opened in the system browser by default. To enable the internal browser set the UseInteralBrowser property to true.

Default Field customization such as animation, background color, stretching masks, etc. can be used on the MASTAdView instance. Note that the MASTAdView instance itself a Canvas.

The MASTAdView instance allows direct access to the ad content container fields. These fields can be customized but should not have properties adjusted that would affect their behavior in the MASTAdView view.

## Customize ad network properties

By default the Mocean ad network is used. To use a different network specify the URL of the desired network with the AdServerURL property. The network is expected to follow the same interface and implementation as the Mocean ad network.

To supply additional parameters or override SDK defaults set ad network parameters using the setAdServerParameter method. All parameter keys and values must be NSString objects. The ad request parameters can be found here: <http://developer.moceanmobile.com/Mocean_Ad_Request_API>

## Location detection

The SDK can automatically determine the user’s location using the Windows Phone System.Device.Location implementation. This feature is disabled by default and can be enabled with the LocationDetectionEnabled property or the EnableLocationDetection method. The later allows more customized control over how the location is acquired.

Developers that wish to reuse existing application location information can do so by setting location parameters for the ad network. See the section above for setting custom ad request parameters.

# Content Updates

MASTAdView updates content only by the following methods:

1. Creating the MASTAdView instance in XAML and supplying a valid Zone property. When the instance is loaded it will automatically update once, unless the XAML also has the UpdateInterval property set.
2. Calling the Update method. Use this after loading the phone page that will display the instance.

Call the Reset method to stop ad loading and cancel any timers. Note that any backing code will work for XAML or code originated instances. As with other XAML components specifying an instance name with the x:Name property is recommended to make code access easier.

# Detecting Updates and Failures

Sometimes a developer might want to take a special action if no ad is available that satisfies the current constraints sent to the mobile ad server. This might occur if a particular ad type or minimum size was requested, and no matching ad is available. This could also happen if all ads scheduled for the requested zone have reach the maximum daily or monthly cap. Developers can also take advantage of a successful ad update to redisplay a hidden banner or to show interstitials after the ad is downloaded.

The SDK includes various events on the MASTAdView class which applications can implement to receive notifications when download related ad events occur. These events include the following that relate to ad download status:

* *AdReceived* which is triggered after the ad content has been downloaded successfully.
* *AdFailed* which is invoked if downloading ad content fails for any reason.

# Troubleshooting

## Ad content loading issues

1. Verify the specified content zone has ad content.
2. Implement the ad instance’s error and logging events to debug any ad download failure errors.
3. Enable simple test banners by enabling test mode with the Test property. Note that the zone configuration on the ad server would need test banners configured.

# Next Steps

More thorough, complex examples and additional use cases in the sample application distributed with the SDK. Both the sample app and the SDK itself are available in source code form from: <http://code.google.com/p/mocean-sdk-windows-phone/>.

Additional documentation, information, and other supported platforms on our developer wiki at: <http://developer.moceanmobile.com/Main_Page>.