Blackberry mOcean SDK

Developer Getting Started Guide

For Blackberry SDK Version 3.0

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 JAR 5

User Interface / Layout (Design) 6

Simple Ad Integration 7

Interstitial Ad Integration 9

MASTAdView Customization 11

Customize view appearance 11

Customize ad network properties 11

Location detection 11

Content Updates 12

Detecting Updates and Failures 12

Troubleshooting 12

Ad content loading issues 12

Next Steps 13

# 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 2.0 SDK. Prior to installing the 3.0 SDK remove the 2.0 SDK references.

# System Requirements

* Blackberry development environment (Eclipse used to develop SDK)
* Blackberry SDK 5.0 or higher

# Prerequisites

This guide does not cover Blackberry development techniques or instructions for using Eclipse/JDE to develop applications for Blackberry. Blackberry developer documentation is available from RIM at <http://developer.blackberry.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-blackberry/>.

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

# Feature List

* **HTML/ JS ads**

SDK supports displaying web ads using the BrowserField component.

* **Image/Text ads**

SDK supports displaying image and text ads with non-BrowserField native components.

* **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.out.

# Installing the Ad SDK

## SDK Integration Methods

The SDK can be linked in as a JAR or copying in direct source. Whichever method is best can be determined by the integrating developer/team.

## Building and adding the JAR

The JAR 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. Because the MASTAdView can have a handler and may needed to be accessed later it’s good practice to create a member variable for the instance (vs. adding it to a manager and leaving it be).

Basic code to extend an existing controller:

|  |
| --- |
| private MASTAdView adView = null;  private void onUiEngineAttached(Boolean attached) {  if (attached)  {  if (adView == null)  {  // Create the MASTAdView adView instance.  adView = new MASTAdView();   // Insert the ad view at the top of the screen.  insert(adView, 0);   // Setup normal field properties like autorotation masks, background  // colors, etc..   // NOTE: Developers will need to get their own site and zone  // configuration and should never use these test values in production  // releases.  adView.setSite(19829);  adView.setZone(88260);  }    // Update the ad to get content from the ad server and configure to  // update every minute. Force the update since the screen was just  // attached.  adView.update(60, true);  }  else  {  if (adView != null)  {  // Since the screen is going away stop the ad view from updating.  adView.reset();  }  } } |

MASTAdView instance creation, configuration, and cleanup

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

* **Publisher site**: this is setup through the Mocean Mobile UI when you setup ad feeds to display content in your application. Typically a “site” will be used to identify one of your applications and distinguish it from another of your applications. The site is required in order to request an ad.
* **Ad zone**: this is used to identify one specific ad placement in your application. In this example we have created one placement so far, the banner ad to be displayed at the top of the screen. If we choose to display ads in another part of this application, a different placement will be used for that location. Zones are created through the Mocean Mobile UI and target content to ad placements in your application. A given zone falls under one site. 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.

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 need to be added to a screen or manager. Note however that an interstitial instance can’t be used as a banner and vice versa.

Basic code to add an interstitial MASTAdView instance:

|  |
| --- |
| private MASTAdView interstitialAdView = null;  private void onUiEngineAttached(Boolean attached) {  if (attached)  {  if (interstitialAdView == null)  {  // Create the MASTAdView interstitial instance.  interstitialAdView = new MASTAdView(true);   // NOTE: Developers will need to get their own site and zone  // configuration and should never use these test values in production  // releases. There will usually be different site/zone combinations  // for different ad placements and types.  interstitialAdView.setSite(19829);  interstitialAdView.setZone(88260);  }   // Every time the view appears, display the interstitial.  interstitialAdView.update();  interstitialAdView.showInterstitial();  }  else  {  if (interstitialAdView != null)  {  // Reset will stop any internal timers and close the interstitial  // if open.  interstitialAdView.reset();  }  } } |

MASTAdView interstitial reference creation, configuration, cleanup and presentation

Note that interstitial ad views 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, orientation/sizing masks, etc. can be used on the MASTAdView instance. Note that the MASTAdView instance itself a Manager.

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 setAdServerURL method. 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 Blackberry implementation of the javax.microedition.location package. This feature is disabled by default and can be enabled with the setLocationDetectionEnabled and enableLocationDetection methods.

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. Calling the update method. Use this after initializing and during display of the owning Screen.
2. Calling the update(int, boolean) method. This method will cause the SDK to update every interval seconds and optionally allow forcing the update. Interacting with the current ad suspends the timer. This can be due to a user clicking and viewing publisher content with the internal web browser or if the user’s action leaves the application.

Call the reset method to stop ad loading and cancel any timers.

# 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 an optional *MASTAdViewHandler* interface which applications can implement to receive notifications when download related ad events occur. This protocol includes the following methods that relate to ad download status in addition to others not described in this section:

* *onAdReceived* which is invoked after the ad content has been downloaded successfully.
* *onDownloadError* 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 handler and debug any ad download failure errors.
3. Enable simple test banners by enabling test mode with the setTest method.

# 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-blackberry/>.

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