BlackBerry mOcean Java SDK

Developer Guide

V 1.6

Table of Contents

[Overview 3](#_Toc284945351)

[System requirements: 3](#_Toc284945352)

[SDK contents: 3](#_Toc284945353)

[Library version: 3](#_Toc284945354)

[Preverification information: 3](#_Toc284945355)

[Feature List: 3](#_Toc284945356)

[Creating sample project with Eclipse IDE: 4](#_Toc284945357)

[Creating sample project with Netbeans IDE: 11](#_Toc284945358)

[Creating sample project with BlackBerry IDE: 15](#_Toc284945359)

[AdServer Constructor: 18](#_Toc284945360)

[Usage examples: 20](#_Toc284945361)

[Text and images Advert: 20](#_Toc284945362)

[Parameters auto detection: 20](#_Toc284945363)

[Default image: 20](#_Toc284945364)

[Callback notifications: 21](#_Toc284945365)

## Overview

This document provides a description of the following:

* System requirements
* Installation Instructions
* SDK contents
* SDK API reference
* Sample of usage

### System requirements:

* Microsoft Windows XP or later.
* IDE, supports BlackBerry environment:
  + Eclipse (<http://us.blackberry.com/developers/javaappdev/javaplugin.jsp>)
  + Netbeans (<http://netbeans.org/downloads/index.html>) with Blackberry plugin (<http://plugins.netbeans.org/PluginPortal/faces/PluginDetailPage.jsp?pluginid=11194> )
  + BlackBerry JDE (<http://us.blackberry.com/developers/javaappdev/javadevenv.jsp> )

### SDK contents:

Javadoc - SDK documentation

Release – SDK libraries for appropriate BlackBerry OS

Sources - SDK sources for appropriate BlackBerry OS

Sample - Sample usage application

### Library version:

For Blackberry OS’es 4x use Adserver.jar located in \Releases\4.x folder.

For Blackberry OS’es 5 and higher use Adserver.jar from \Releases\5.x folder

### Preverification information:

Do not try to preverify compiled library (Adserver.jar) as it already preverified. If you want to manually compile library from sources – use preferify.exe from bin folder of your Rim JDE.

### Feature List:

* Supported ads type - Simple ad only
* Open links in internal or external web browser
* Phone user-agent auto detection or defined value
* Predefined ip address or ip auto detection
* Predefined country code country code auto detection
* Phone carrier auto detection
* Predefined user location of auto detection
* Landscape or portrait phone orientation
* Callback notification

## Creating sample project with Eclipse IDE:

1. Open eclipse with your workspace and go to **New => BlackBerry Project**
2. Type project name (ex. AdserverTest):
3. Import mOcean SDK into **lib** folder
4. Add mOcean SDK **jar** into project build path
5. Open sample project properties and open Java Build Path.
6. Open “*Order and Export*” tab and select *Adserver.jar*
7. Create package “mypackage” and add into MyApp.java and MyScreen.java into them
8. Run

**Example content of MyApp.java:**

**package** mypackage;

**import** net.rim.device.api.ui.UiApplication;

/\*\*

\* This class extends the UiApplication class, providing a

\* graphical user interface.

\*/

**public** **class** MyApp **extends** UiApplication

{

/\*\*

\* Entry point for application

\* **@param** args Command line arguments (not used)

\*/

**public** **static** **void** main(String[] args)

{

MyApp theApp = **new** MyApp();

theApp.enterEventDispatcher();

}

/\*\*

\* Creates a new MyApp object

\*/

**public** MyApp()

{

pushScreen(**new** MyScreen());

}

}

**Example content of MyScreen.java:**

**package** mypackage;

**import** net.rim.device.api.system.Display;

**import** net.rim.device.api.ui.component.Dialog;

**import** net.rim.device.api.ui.container.MainScreen;

**import** com.adserver.core.AdClickListener;

**import** com.adserver.core.Adserver;

**import** com.adserver.core.AdserverBase;

**import** com.adserver.core.AdserverRequest;

**import** com.adserver.core.WebView;

**import** com.adserver.utils.EventListener;

/\*\*

\* A class extending the MainScreen class, which provides default standard

\* behavior for BlackBerry GUI applications.

\*/

**public** **final** **class** MyScreen **extends** MainScreen **implements** EventListener, AdClickListener

{

**private** Adserver field;

/\*\*

\* Creates a new MyScreen object

\*/

**public** MyScreen()

{

setTitle("Adserver Test");

field = **new** Adserver("571", "345",

AdserverRequest.*ADS\_TYPE\_IMAGES\_ONLY*, **null**, **null**, **null**, **null**,

**null**, **new** Integer(0), **null**, **null**, **null**, **null**, **null**, **null**, **null**,

**null**, **null**, **null**, **null**, **null**, **null**, **null**, **null**, **null**, **null**,

**null**, "test", "defaultImage", Boolean.*FALSE*, **this**, **null**);

field.addListener(**this**);

}

**public** **void** onLoaded(WebView webView) {

webView.setSize(Display.*getWidth*(), Display.*getHeight*());

add(webView);

}

**public** **void** onError(**final** String msg) {

Dialog.*alert*(msg);

System.*exit*(-1);

}

**public** **synchronized** **void** didAdClicked(AdserverBase inst) {

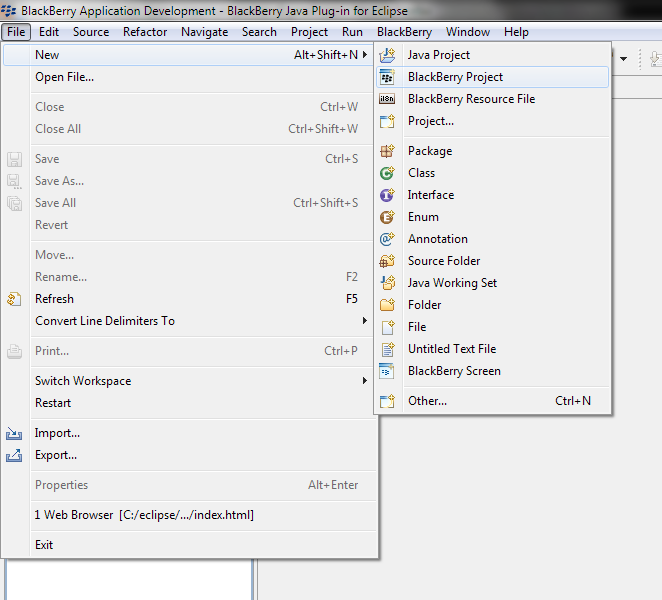
**if** (field.equals(field)) {

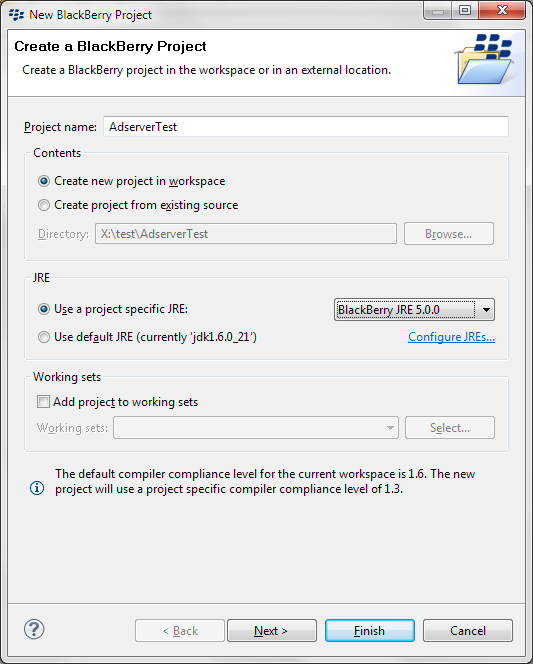
// ads on field clicked

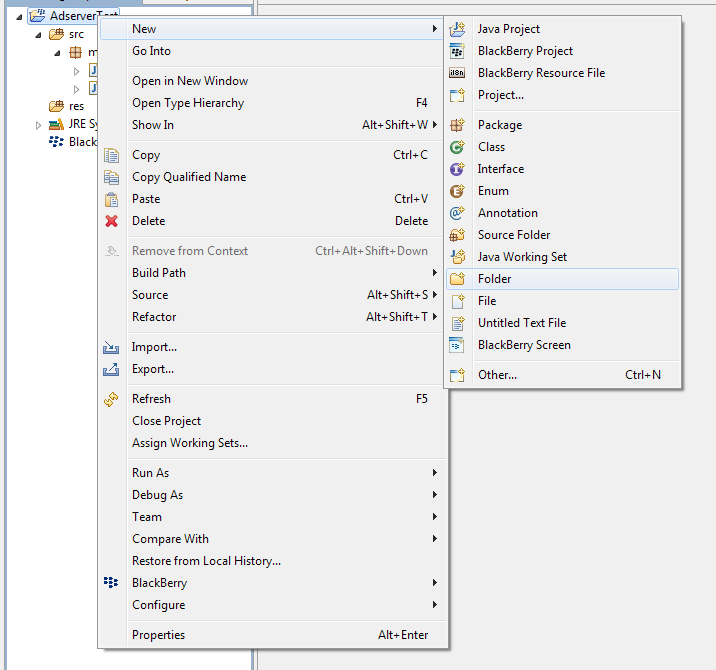
}

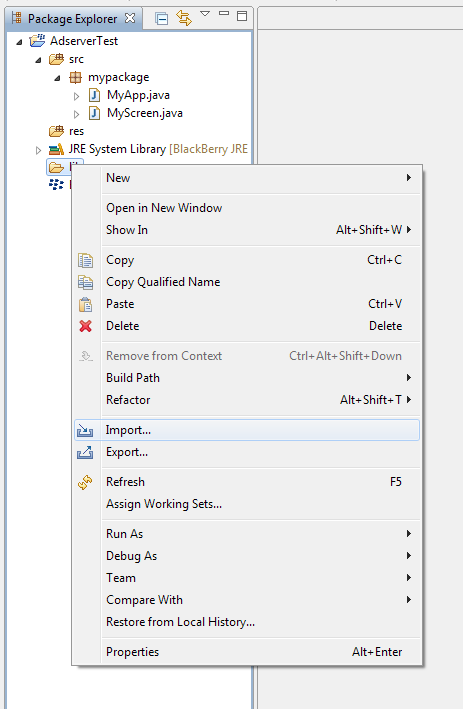
}

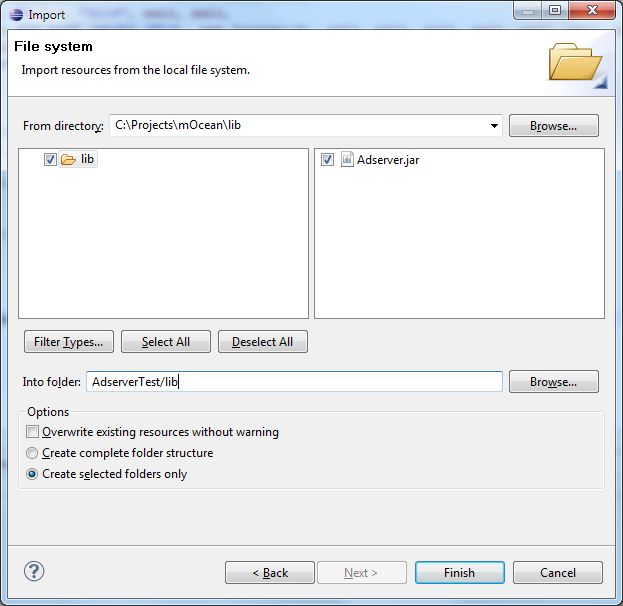
}

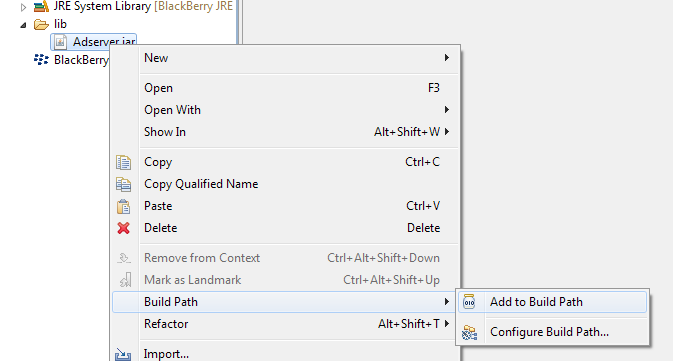


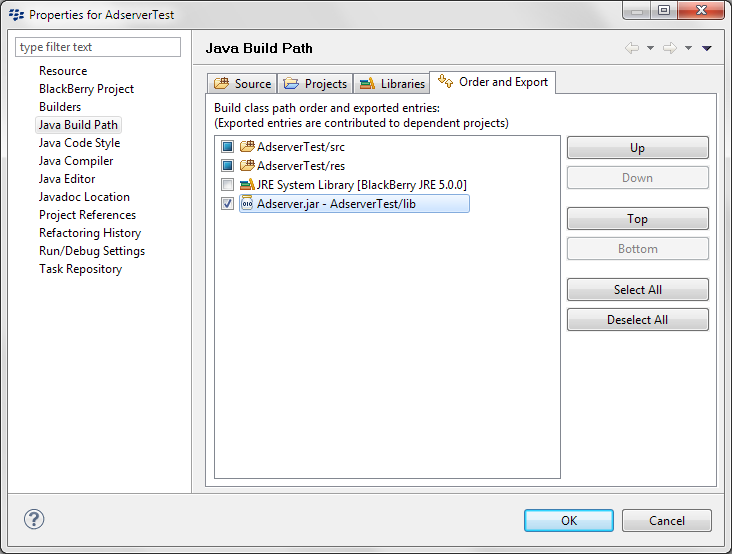


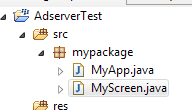


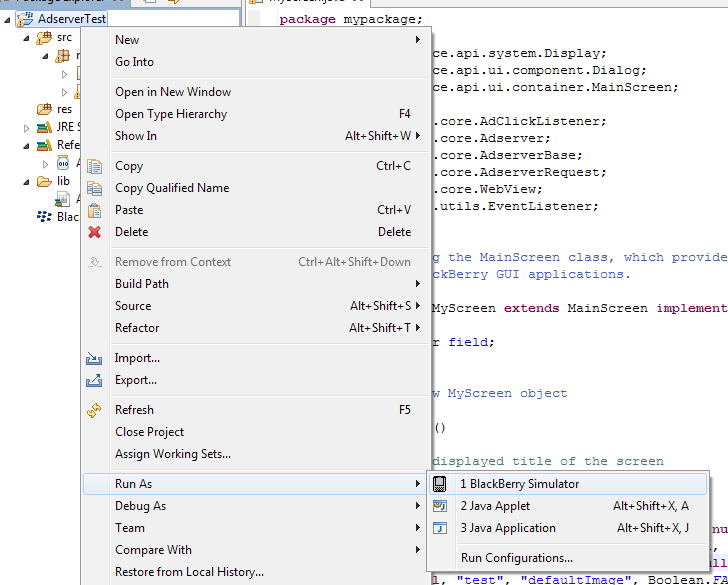


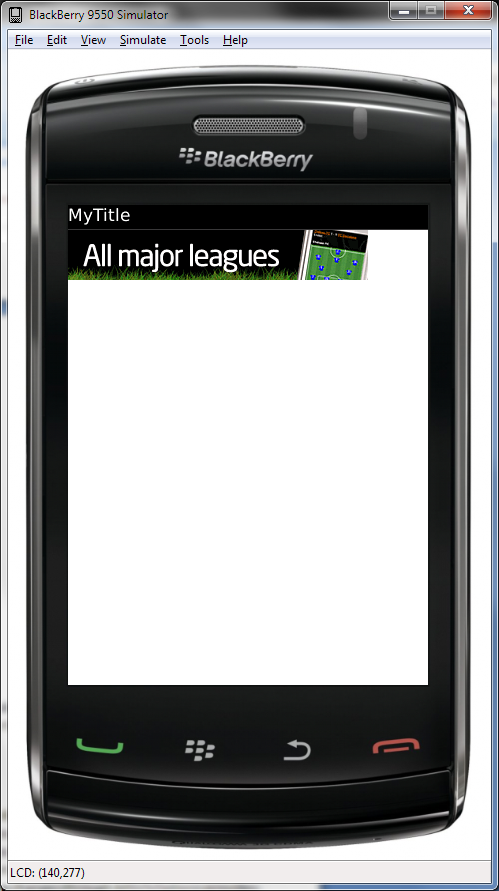












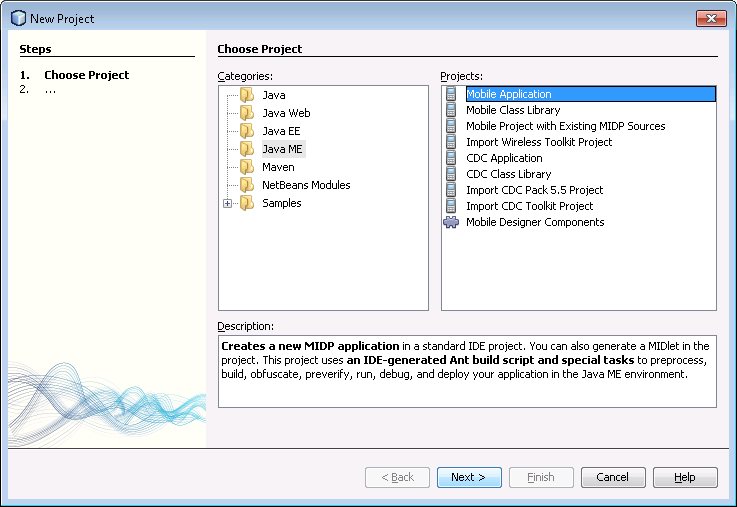
## Creating sample project with Netbeans IDE:

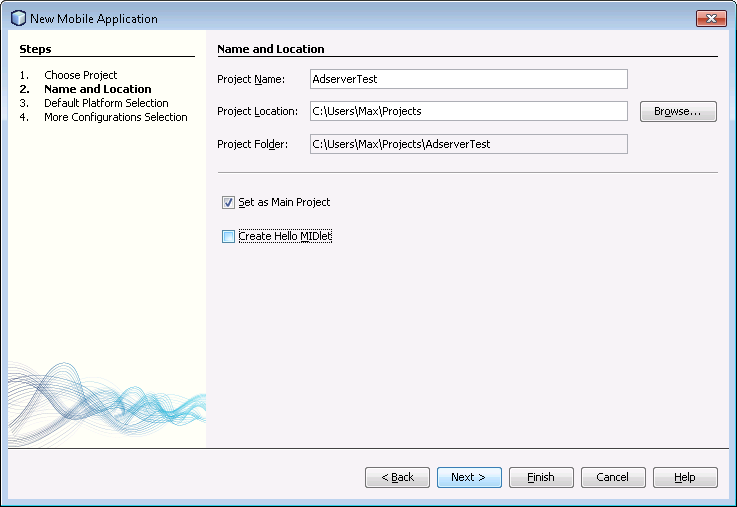
Preparing IDE:

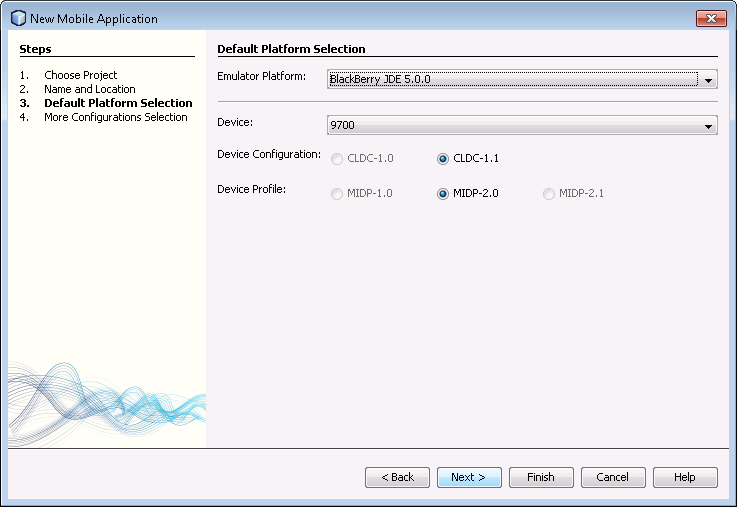
1. Run installed Netbeans IDE
2. Go to Tools => Plugins => Downloaded and push “Add Plugin” button
3. Select downloaded netbeans blackberry plugin, press OK, and then push “Install” button
4. Restart netbeans
5. Go to Tools => Java Platforms. Push “Add Platform” button and select “Java ME MIDP Platform Emulator”
6. System will search installed jdk’s, then select BlackBerry JDK in result list

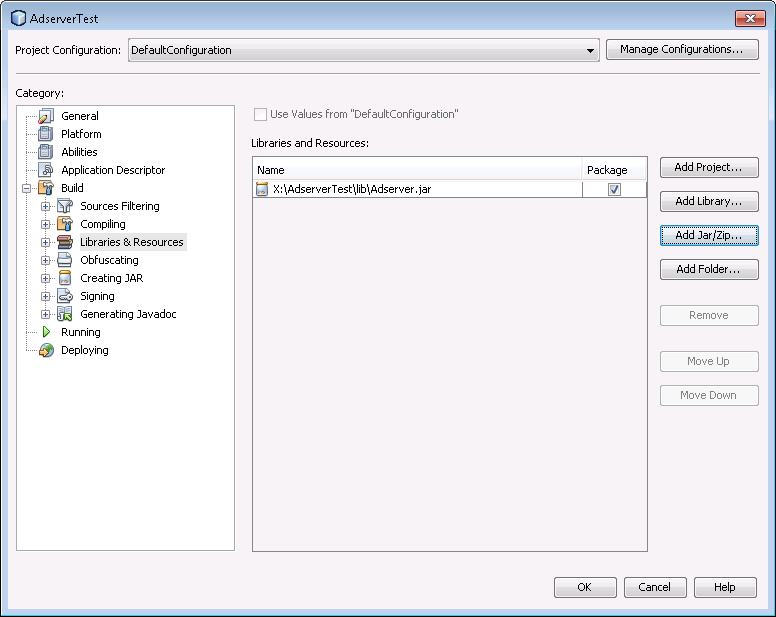
Creating sample project:

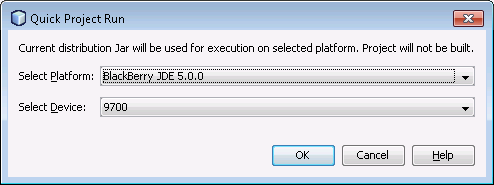
1. File => New Project
2. Select Java ME => Mobile Application
3. Type project name (ex. AdserverTest); Uncheck “Create Hello Midlet”
4. Select “Emulator platform” (ex. Blackberry JDE 5.0.0)
5. Open create project folder in any file manager and create folder **lib.**
6. Copy mOcean SDK jar into **lib** folder, copy build.xml and bb-build.xml(provided with that doc) into project folder (replace original build.xml)
7. Open in netbeans AdserverTest properties
8. Go to Build => Libraries & Resources
9. Push “Add Jar/Zip” and select mOcean SDK jar from our **lib** folder. Check “package” option.
10. Create package “mypackage” and place into them two files MyApp.java, MyScreen.java provided in “Eclipse IDE” example.
11. To alter project title, version, icon and vendor manually edit build.xml
12. Go to Run => Run Main Project







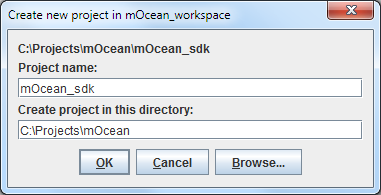


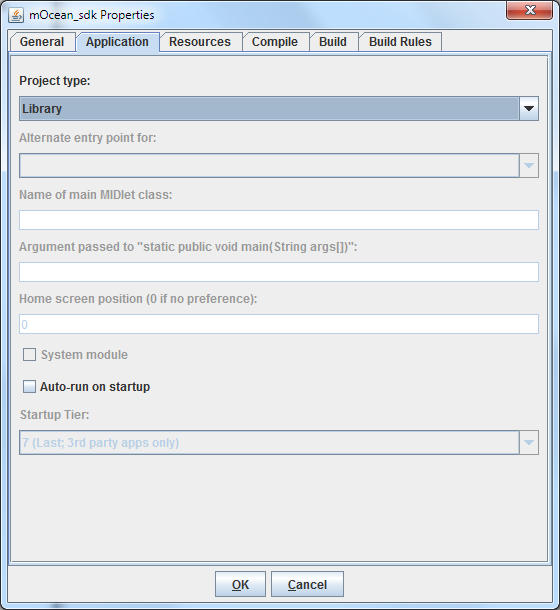


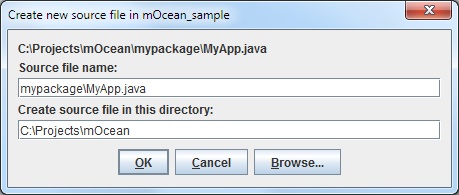


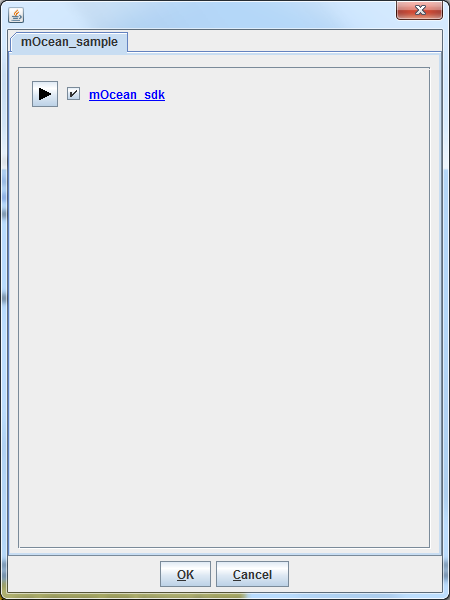
## Creating sample project with BlackBerry IDE:

1. Open BlackBerry JDE, create sample workspace
2. Create new project “mOcean\_sdk” that will contain our mOcean SDK library
3. Copy Adserver.jar inside project directory
4. Go to “mOcean\_sdk => Add File to Project
5. Select mOcean SDK library file (File Type set “All files”)
6. Go to mOcean\_sdk => Properties
7. Click the Application tab. From the Project Type drop-down list, select Library.
8. Right-click mOcean\_sdk project and click **Build Project**.
9. Create new project “mOcean\_sample” that will contain our mOcean SDK sample
10. Create new source files MyApp.java and MyScreen.java (mOcean\_sample => Create new File in Project); Defining package name in Project name field (ex. mypackage\MyApp.java)
11. Place code from “eclipse IDE” instruction
12. Right-click mOcean\_sample project and click Project Dependencies
13. Check mOcean\_sdk field
14. Go to Build => Build and Run mOcean\_sample project











## AdServer Constructor:

public Adserver(java.lang.String campaign,

java.lang.String appID,

int mode,

java.lang.String site,

java.lang.String zone,

java.lang.String ip,

java.lang.String keywords,

java.lang.Integer adsType,

java.lang.Integer over18,

java.lang.String latitude,

java.lang.String longitude,

java.lang.String ua,

java.lang.Integer premium,

java.lang.Integer key,

java.lang.Boolean testMode,

java.lang.Integer count,

java.lang.String country,

java.lang.String region,

java.lang.Boolean textBordersEnabled,

java.lang.String paramBorder,

java.lang.String paramBG,

java.lang.String paramLINK,

java.lang.String carrier,

java.lang.Integer imageSize,

java.lang.String target,

java.lang.String url,

java.lang.Boolean pixelModeEnabled,

java.lang.String hashId,

java.lang.String defaultImage,

java.lang.Boolean adInternalBrowserEnabled,

AdClickListener clickListener,

java.lang.Integer reloadPeriod)

Parameters:

appID - Identification of application

mode - Working mode:

MODE\_COUNTER\_ONLY - no ads, just counter,

MODE\_ADS\_ONLY = only ads, no counter,

MODE\_COUNTER\_AND\_ADS = ads and counter.

site - The id of the publisher site (default: DEFAULT\_SITE).

zone - The id of the zone of publisher site (default: DEFAULT\_ZONE).

ip - The IP address of the carrier gateway over which the device is connecting (default: DEFAULT\_IP).

keywords - Keywords to search ad delimited by commas.

adsType - Type of advertisement (ADS\_TYPE\_TEXT\_ONLY - text only, ADS\_TYPE\_IMAGES\_ONLY - image only, ADS\_TYPE\_TEXT\_AND\_IMAGES - image and text, ADS\_TYPE\_SMS - SMS ad, default - DEFAULT\_ADS\_TYPE).

SMS will be returned in XML.

over18 - Filter by ad over 18 content:

OVER\_18\_TYPE\_DENY - deny over 18 content,

OVER\_18\_TYPE\_ONLY - only over 18 content,

OVER\_18\_TYPE\_ALL - allow all ads including over 18 content). default: DEFAULT\_OVER\_18

latitude -

longitude -

ua - The browser user agent of the device making the request.

premium - Filter by premium (PREMIUM\_STATUS\_NON\_PREMIUM - non-premium, PREMIUM\_STATUS\_PREMIUM - premium only, PREMIUM\_STATUS\_BOTH - both).

Can be used only by premium publishers.

key - Output format: OUTPUT\_FORMAT\_NORMAL - Normal format uses key

OUTPUT\_FORMAT\_XML - XML output

OUTPUT\_FORMAT\_JSON - JSON output.

testMode - Default setting is test mode where, if the ad code is properly installed, the ad response is "Test MODE" (default: DEFAULT\_TEST\_MODE).

count - Quantity of ads, returned by a server (Maximum: 5; Default: DEFAULT\_COUNT).

country - Country of visitor. Will override country detected by IP. (http://www.mojiva.com/docs/iso3166.csv)

region - Region of visitor. Codes for US and Canada - http://www.mojiva.com/docs/iso3166\_2.csv, others - http://www.mojiva.com/docs/fips10\_4.csv.

textBordersEnabled - Show borders around text ads (Boolean.FALSE - non-borders, Boolean.TRUE - show borders, default: DEFAULT\_TEXT\_BORDER).

paramBorder - Borders color (#XXXXXX)

paramBG - Background color (#XXXXXX)

paramLINK - Link color (#XXXXXX)

carrier - Carrier name

imageSize - Override size detection for banners (IMAGE\_SIZE\_SMALLEST - the smallest, IMAGE\_SIZE\_LARGEST - the largest).

target - Target attribute for:

TARGET\_BLANK - open the linked document in a new window,

TARGET\_SELF - open the linked document in the same frame,

TARGET\_PARENT - open the linked document in the parent frameset,

TARGET\_TOP - open the linked document in the full body of the window)

url - URL of site for which it is necessary to receive advertising.

pixelModeEnabled - Redirect to image for ad (or tracking pixel 1x1) directly (default: DEFAULT\_PIXEL\_MODE).

hashId - Unique id of Adserver instance (used for cache)

defaultImage - Image resource name, that placed on screen when no network and cache

adInternalBrowserEnabled - Open ad links in external or internal browser (default: DEFAULT\_AD\_BROWSER\_MODE)

reloadPeriod - Ad reload period

## Usage examples:

### Text and images Advert:

Sample code, provided above will work on any simulator and real device even without internet connection. Default banner will be shown on application launch.

If you want test library on real world example you must alter AdserverScreen.java source code and add “site” and “zone” to Adserver constructor. For Default mOcean Ad use this values -

* site – 6121
* zone – 9216

Constructor will look like:

**new** Adserver("571", "345",

AdserverBase.*MODE\_ADS\_ONLY*, "6121", "9216", **null**, **null**,

**null**, **new** Integer(0), **null**, **null**, **null**, **null**, **null**, **null**, **null**,

**null**, **null**, **null**, **null**, **null**, **null**, **null**, **null**, **null**, **null**,

**null**, "test", "defaultImage", Boolean.*FALSE*, **this**, **null**);

Run sample application with suggested **site**/**zone** parameters and after default ad you will see downloaded advertisement.

Note: **adsType** (seventh parameter in constructor) set to null – so ads type will be text based. If you want to see image – based ads – you can set **adsType** to ADS\_TYPE\_IMAGES\_ONLY. Constructor will look like:

**new** Adserver("571", "345",

AdserverBase.*MODE\_ADS\_ONLY*, "6121", "9216", **null**, **null**,

**new** Integer(AdserverRequest.*ADS\_TYPE\_IMAGES\_ONLY*), **new** Integer(0), **null**, **null**, **null**, **null**, **null**, **null**, **null**,

**null**, **null**, **null**, **null**, **null**, **null**, **null**, **null**, **null**, **null**,

**null**, "test", "defaultImage", Boolean.*FALSE*, **this**, **null**);

### Parameters auto detection:

If you leave **ip**, **latitude**, **longitude**, **ua** (user agent) , **country** and **region** parameters as null, they will auto detect, you can manually set values as well

### Default image:

You can change default Ad by alter defaultImage parameter – add new image to application resource folder and point it with defaultImage.

### Callback notifications:

* onError - notified when SDK can’t load advertisement: no internet connection or internal problems.
* didAdClicked – notified when user clicked on advert.