Vserv Windows Phone 8.0 VMAX SDK v 3

Contents

Introduction

The Vserv VMAX SDK is designed to help you integrate Ads in your application and fast track your way to monetizing your WP8.0 applications.

Prerequisites

Before you begin, we need to make sure of a few things:

1. You have a Vserv.mobi Developer Account. If you have one, go to Step 2. If you do not, sign up at our Developer Portal.
2. Note down your **ZoneID**. This will be required later on when you use the VMAX SDK to request Ads. The default ZoneID will be available in your welcome email or you could visit http://mp.vserv.com/ad-spot-list.html to get a list of all your zones.

Download the SDK

The first step for integrating VMAX SDK in your WP 8.0 application is to download our SDK package. The SDK package is available from here.

Alternatively the SDK can also be downloaded as NuGet package using the following command in the Nuget Package Manager.

```
Install-Package com.vserv.windows.ads.wp8
```

We suggest that you unzip the SDK Zip file that you just downloaded into a temporary folder. Assuming that the folder is **[VMAX_SDK_FOLDER]**, you will see the following contents:

VMAXSDKWP8/packages	This folder contains the VMAX SDK and its dependencies.
VMAXSampleAppWP8/	Sample WP8 project source code that demonstrates VMAX SDK integration.

Integrating the VMAX SDK

Before you start you need to make sure that your project satisfies the following requirements:

1. You need to copy the '**packages'** folder into your project's folder structure and then add the **com.vserv.windows.ads.wp8.dll** to your project [Project -> Add Reference -> Browse] which will be present in the **'packages'** folder itself.
2. You also need to make sure that your project has the following capabilities:-

ID_CAP_NETWORKING
ID_CAP_MEDIALIB_AUDIO
ID_CAP_MEDIALIB_PLAYBACK
ID_CAP_SENSORS
ID_CAP_WEBBROWSERCOMPONENT
ID_CAP_MEDIALIB_PHOTO
ID_CAP_PHONEDIALER
ID_CAP_IDENTITY_DEVICE
ID_CAP_LOCATION
ID_CAP_IDENTITY_USER

You can then reference the Vserv VMAX namespace in your code by using the following 'using directives'

Through code-behind in C#

```
using com.vserv.windows.ads.wp8;
using com.vserv.windows.ads;
```

Through XAML-

To integrate the **VservAdView** through XAML you have to first import the namespace in which is declared. To do that, copy this statement to the starting of your XAML page where other XAML namespaces are declared .

```
xmlns:ad="clr-namespace:com.vserv.windows.ads.wp8;assembly=com.vserv.windows.ads.wp8"
```

Requesting Ads

The VMAX SDK provides you with various mechanisms to request for Ads in your WP8 application. The SDK provides both simple fully-managed Ad request methods and also the Cache & Show Ad methods to enable prefetching of ads and showing them later.

Requesting Interstitials

Managed Interstitials using LoadAd()

To request an interstitial you need to call the LoadAd() method.
You need to use this function after the page has loaded. Here is an example -

```
VservAdView  adView=new VservAdView ();
adView.UX=VservAdUX.Interstitial; // To specify explicitly that you want to display Interstitial ads
adView.ZoneId="<zone id>"; // This will set the zone from where ads will be fetched.
```

adView.TimeOut = 20; // To set timeout if Ad failed to fetch.

To set Demographic parameters(optional),

adView.UserInfo.Age =<age> ; // To set age of user
adView.UserInfo.City = <city>; // To set city of user
adView.UserInfo.Country = <country>; // To set country of user
adView.UserInfo.DOB = <dob>; // To set date of birth of user
adView.UserInfo.Email = <email>; // To set email of user
adView.UserInfo.Gender = <gender>; // To set gender of user

adView.LoadAd(); // This will load the Ad and on success provides a callBack named **DidLoadAd**. You need to handle this to perform any action on this event in your application.

Interstitial Caching

To Cache Interstitial Ads and showing them later will be possible through below api:

adView.CacheAd(); // This will cache Ad and send the respective callback . If cached successfully it will give the callBack **DidCacheAd**. You need to handle this to perform any action on this event in your application.

Showing Cached Ads:
adView.ShowAd(); // This will show Cached Ad

Cancel Ads:
adView.CancelAd(); // This will cancel Cache Ad and free up all resources of adview.

Requesting Banner Ads

Managed Banner Ads using LoadAd()

You can request for Banner Ads which auto refresh automatically in a specific container using LoadAd() method. The signature for the method is shown below:

VservAdView adView=new VservAdView (); // This will initialize the VservAdView
// Now you can set the properties and configure the adView
adView.ZoneId = "<your zone id>"; // enter your <zone id> here
adView.UX= VservAdUX.Inline; // for inline ads

adView.TimeOut = 20; // To set timeout if Ad failed to fetch.
adView.RefreshRate = refreshRate; // To set refreshrate for banner Ads
adView.Refresh =true; // To enable refresh of banners true/ false

```
(optional)
adView.ResumeRefresh(); // To resume refresh
adView.StopRefresh();  // To stop refresh
adView.PauseRefresh(); // To pause refresh

To set Demographic parameters(optional):

adView.UserInfo.Age =<age> ; // To set age of user
adView.UserInfo.City = <city>; // To set city of user
adView.UserInfo.Country = <country>; // To set country of user
adView.UserInfo.DOB = <dob>; // To set date of birth of user
adView.UserInfo.Email = <email>; // To set email of user
adView.UserInfo.Gender = <gender>; // To set gender of user
```

Optional Callbacks

The VservAdView gives callbacks respectively on successful and unsuccessful fetching attempts of Ads as well as based on interactions with the ads.

FailedToLoadAd Callback

At times we might not be able to serve an Ad to you, in which case the VMAX SDK provides the following callBack.This will call handler attached to it.

```
private void AdFailed(object sender,   com.vserv.windows.ads.wp8.VservAdView
.AdFailedEventArgs e)
{
   // ..handler code goes here
}
```

FailedToCacheAd Callback

At times we might not be able to cache the Ad for you, in which case the VMAX SDK provides the following callBack.This will call handler attached to it.

```
private void DidFailed_CacheAd(object sender, EventArgs e)
      {
         // … handler code goes here
      }
```

DidLoadAd Callback

This Callback method is invoked when fetched Ad is successfully rendered.

```
private void AdReceived(object sender, EventArgs e)
{
   // ..handler code goes here
}
```

DidCacheAd Callback

This Callback method is invoked when Ad is cached successfully.

```
private void AdCached(object sender, EventArgs e)
{
   // ..handler code goes here
}
```

DidInteractWithAd Callback

This Callback method is invoked when Ad is clicked or interaction happens with Ad.

```
private void InteractedWithAd(object sender, EventArgs e)
{
   // ..handler code goes here
}
```

WillDismissOverlay Callback

This Callback method is invoked when an Interstitial Ad/Overlay is dismissed.

```
private void AdCollapsed(object sender, EventArgs e)
{
   // ..handler code goes here
}
```

WillLeaveApp Callback

This Callback method is invoked When Ad is going to leave current App and open any outside App.

```
private void LeavingApplication(object sender, EventArgs e)
{
   // ..handler code goes here
```

```
}
```

WillPresentOverlay Callback

This Callback method is invoked when Interstitial Ad/Overlay is going to be shown.

```
private void AdExpanded(object sender, EventArgs e)
{
   // ..handler code goes here
}
```

Testing Integration

The Vserv SDK Integration can be tested by either setting your integration to test mode for certain devices or by using the Test Zones provided by Vserv.

Adding Test Devices

The Vserv VMAX SDK allows you to set a number of Devices as Test Devices by passing their DeviceID to the SDK. You can add test devices by using the **TestDevice** property which takes a list of id(string) as its parameters.

```
adView.TestDevice = new List<string>() { [your device id/ids here] };
```

Each Device's DeviceID can be fetched by using the following piece of code in your application.

```
try
{
        var value = (byte[])DeviceExtendedProperties.GetValue("DeviceUniqueId");
        return Convert.ToBase64String(value);
}
catch { return string.Empty; }
```

Using the TestZones

You can also test your integration across our solutions by using the following ZoneIDs

Billboard Test ZoneID: **8063**
Banner Test ZoneID: **20846**

Note: Please make sure your production App does not have these Zones as you would not be earning revenue using these Zones.

Mediation

To Support Mediation with partners:

1.) The **com.vserv.windows.ads.wp8.dll** in your project consists of all supported Client to Server integration Partners like **Admob, Microsoft, AdDuplex**.
2.) Fill required details on Vserv web interface to start mediation for that zone.

Queries

If you have any Queries or issues please email us at developers@vserv.mobi.