Smaato SDK for Windows Phone 8

- Developer's Guide

Changes

Date	Author	Changes	Version
2012/10/25	RLH	Initial version	1.0
2013/02/24	RLH	• Add support for Rich Media • Add Format Property • Add ModifyRM property	1.1
2013/09/02	RLH	Add COPPA support	1.1.4993
2014/05/24	RLH	• Default width and height • Fix Richmedia • Handle HTLM Ads	1.2
2014/06/14	RLH	Add Interstitial Ad Support	1.3

Introduction

The SOMA SDK library for Windows Phone 8 enables a software developer to easily integrate SOMA Ads in their Windows Phone 8 applications. The SDK connects with the SOMA server and downloads the ad in the background. By default, a new ad is downloaded every 60 seconds. The SDK can display the ad and support the click through to see the full page ad in a Browser Window.

The SOMA SDK for Windows Phone 8 consists of two layers plus one class:

1. Presentation Layer, SomaAdViewer, which is the normal way to use the SDK, and,
2. Communication Layer, SomaAd, which gives the developer more control. The communications layer is used by the Presentation Layer to connect with the SOMA Ad Server and download the ads.
3. Interstitial Ads, SomaInterstitialAd, which is a class that allws the developer to retrieve and display interstitial Ads.

Sample Apps

The SDK includes four sample apps that demonstrate the three ways to use the SDK:

1. SOMA Sample - Pres Layer Design Time – in this sample the SomaAdViewer Control is instantiated at design time and the required properties are also set at design time.
2. SOMA Sample – Pres Layer Execution Time – in this sample the SomaAdViewer Control is instantiated at execution time as well as the required properties.
3. SOMA Sample - Communications Layer – in this sample SomaAd is used; its properties and methods are set at execution time to retrieve ads which are then displayed by the developer.
4. SOMA Sample – XNA = Comm Layer – in this sample SomaAd is used; its properties and methods are set at execution time to display ads in an XNA environment.
5. SOMA Sample – Interstitial Ads – in this sample the SomaInterstitialAd class is used to Load and Show an interstitial ad.

Overview of Presentation Layer - SomaAdViewer

The SomaAdViewer Control may be instantiated by the developer at either design time or at execution time. It will then invoke the Communication Layer to connect to the SOMA server and retrieve the ads on a background thread. When an ad has been received, the Presentation Layer will display the ad in a Web Browser Control. When the ad is tapped by the user, a Web Browser Task is launched in a new windows displaying the associated click-through ad. The developer should be sure to perform the normal state save and restore tasks to preserve their application's user interface.

Overview of Communications Layer - SomaAd

The Communication Layer will handle the ad request, the parsing of the XML response, the download of the ads (and beacons) and the handling of errors. The application developer will not be able to bypass this layer. The interface of the Communications Layer will provide access to the following data:

- Type of ad (text or image)
- Ad (the text and/or the image byte stream)
- Click-through-action URI

The Communications Layer runs as a background thread. Its ongoing activity will never block the application from working.

The Communications Layer will allow the setting of the properties identified below. The Communications Layer will require an Ad Request object as a parameter that has as its properties, the ad parameters. All requests done by the SDK will use the phone browser's user agent. The same communications layer can be used whether the application is Silverlight or XNA based.

Overview of Interstitial Ads – SomaInterstitialAd

To have interstitial ads the developer instantiates the Interstitial Ad Class, SomaInterstitialAd. Then its properties including Adspace and Pub sre set. The LoadInterstitial() method retrieves an Interstitial Ad from the SOMA server. When it is ready, the NewAdAvailable event is invoked. The ShowInterstitial() method is then used to display the ad.

Using the SOMA Presentations Layer – SomaAdViewer – with design time instantiation

To get started using the SomaAdViewer you must add a reference to the SOMAWP8.DLL in your project. It is distributed as part of the SDK. Next, while displaying the design canvas for the page where you want ads, right-click in the toolbox and select "Choose Items". Now click the Browse Button in the lower right of the dialog box. Browse to the SOMAWP8.DLL and select it. The Windows Phone Components list will now include SomaAdViewer. Select it by checking the box in front of it and press the OK button. Your Toolbox will now contain the SomaAdViewer Control. Drag it to your design surface. Set the left Margin control to 0.

In the properties window for the SomaAdViewer Control, you must set the pub property with your Publisher ID and the adSpace with your AD Space ID. Pub and adSpace of 0 can be used for testing.

You can optionally set the following demographic properties to enhance the ads retrieved:

Property	Description	Purpose	Type	Example
Age	Age of the user	Targeting demographic	Int	30
Gender	Gender of the user	Targeting demographic	Character	m or f
Kws	Keywords describing the content of the page	Targeting Parameter	Comma separated values	Automotive news, cars
Qs	Search String	Targeting Parameter	Comma separated values	Madonna ringtone, fast, food, football

The following properties can be used to set the location of the user. By default, the SDK sets it to the current location:

Property	Description	Purpose	Example
City	Location of user - city	Targeting Parameter	redwood+city
State	Location of user - state	Targeting Parameter	california
Country	Location of user - country	Targeting Parameter	united+states
Countrycode	Location of user –	Targeting Parameter	us

	ISO-3166-1		
Zip	Location of user – postal code	Targeting Parameter	94539
GPS	Coordinates of the users location, set by the SDK but can be overridden by the developer	Targeting Parameter	37.530676%2C-122.262447

You can use the following properties to control the ads:

- **AdSpaceHeight** – height of ad space, default to 100
- **AdSpaceWidth** – width of ad space **,** default to 480
- **AdInterval** – the number of milliseconds between ads; default is 60000 or 60 seconds;
- **LocationUseOk** – when true, Windows Phone 8 Location Services will be used to determine the current location when requesting Ads from SOMA,
- **BackgroundColor** – can be used to specify a string that identifies the background color of the SomaAdViewer control when displaying the ad; by default the WP8 Theme color will be used; this is the html hex string like #FFFFFFFF
- **ShowErrors** – when true, errors from SOMA Ad Server will be shown in the ad display area;
- **Debug** – When true, an email will be generated whenever the SDK encounters an error exception;
- **Format** – control which format ads are requested
  - **All**: image, text or rich media ads;
  - **Img**: image ads only;
  - **Txt**: text ads only;
  - **Richmedia**: Rich Media ads only
- **ModifyRM** – if true Rich Media ads may contain MRAID compliant ads.
- **PopupAd** – When true, the height of the ad will be set to 0 at instantiation; when an ad arrives, the height will be animated to 100 over ½ second, then the ad will remain visible for the duration specified in PopupAdDuration, then the height will be animated to 0;
- **PopupAdDuration** – The number of milliseconds an ad will be shown if PopupAd is true; this defaults to 10000 (10 seconds) which is its minimum.
- **Status** – This is set to “started” after StartAds is processed and “stopped” after StopAds is processed, this is especially useful in the XNA environment as shown in the Sample for XNA.
- **Coppa** – Set to true indicates that the content should be treated as child directed for COPPA pusposes; set to false indicates that content should be treated as not child directed for COPPA purposes

You can use the following methods to control ad retrieval:

- **StartAds** – if ads are stopped, will start retrieving them;
- **StopAds** – if ads are being retrieved, it will stop the retrieval, StartAds must be used to restart the retrieval of Ads.
- **Dispose** – Disposes of the SomaAdViewer control.

There are three events available:

- “**NewAdAvailable**” is notified whenever a new ad is ready for display.
- “**AdError**” is notified whenever an error is encountered retrieving an Ad; SomaAd.ErrorCodes is an Enum for the Error Codes
- “**AdClick**” is notified whenever an Ad is Clicked by the user.

Using the SOMA Communications Layer for WP8 - SomaAd

In order to use the SOMA Communications Layer for Windows Phone 8, first add a reference for SOMAWP8.DLL to your Windows Phone 8 Silverlight project. Then add a Using to the module in which you want to get ads.

Next instantiate the SomaAd class like this:

```
somaAd = new SomaAd();
```

Then add your PubID and AdSpaceID like this:

```
somaAd.Adspace = 12345678;
somaAd.Pub = 12345678;
```

By default, a new ad will be delivered 60 after the previous one. You can change that. This shows changing it to 90 seconds:

```
somaAd.adInterval = 90000;
```

Next, set whatever demographics you want which can be changed on the fly for each ad:

```
// demographics
somaAd.Age = 30;
somaAd.Gender = "m";
somaAd.City = "Fremont";
somaAd.State = "Ca";
somaAd.Zip = "94539";
somaAd.Country = "United States";
somaAd.Countrycode = "us";
```

Now you need to add an Event to receive notification when a new ad is available:

```
somaAd.NewAdAvailable += new
SomaAd.OnNewAdAvailable(somaAd_NewAdAvailable);
```

When you want to start receiving SOMA ads invoke the GetAd method:

```
somaAd.GetAd();
```

You need a Delegate to receive the ad available notification. In this Event Handler you first check to see if there was an error. If not, you need to see if the ad is an image or text and handle accordingly. This code shows the SOMA Ad image in both a Image Control and a WebBrowser Control so that you can choose which to use.

```
void somaAd_NewAdAvailable(object sender, EventArgs e)
{
  if (somaAd.Status == "error")
  {
    textBoxErrorCode.Text = somaAd.ErrorCode;
    textBoxErrorDescription.Text = DateTime.Now.ToShortTimeString() + " " +
                                   somaAd.ErrorDescription;
  }
  else
  {
```

```
        if (somaAd.AdText != String.Empty)
          textBoxTextAd.Text = somaAd.adText;
        if (somaAd.AdType == "IMG")
        {
          ++adCounter;
          textBoxAdCounter.Text = adCounter.ToString();
          textBoxGifCounter.Text = somaAd.GifImageCount.ToString();
          textBoxImageType.Text = somaAd.ContentType;

          int pixelHeight = somaAd.AdImage.PixelHeight;
          int pixelWidth = somaAd.AdImage.PixelWidth;

          webBrowserAdImage.Source = new Uri(somaAd.AdImageUri);
        }
      }
    }
```

Next, you need to handle a user tap on the ad. This shows using the Mouse Enter Event on the Image control to display the target landing site associated with the Ad. This code uses the WebBrowser Control to display the target. However, you could also use the WebBrowserTask to do this. In the XAML, there is a Grid control for the WebBrowser called wbGrid. The sample code included shows how to do this.

```
WebBrowser wb;
    private void imageAd_MouseEnter(object sender, MouseEventArgs e)
    {
      if (somaAd.uri != "" && somaAd.uri != string.Empty)
      {
        wb.Source = new Uri(somaAd.uri);
        ContentGrid.Visibility = Visibility.Collapsed;
        wbGrid.Visibility = Visibility.Visible;
      }
    }
```

In order to shut down the ads, use the Dispose() method. This BackKeyPress Event Handler illustrates doing this as well as handling the WebBrowser control:

```
private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
    {
      if (wbGrid.Visibility == Visibility.Visible)
      {
        e.Cancel = true;
        wbGrid.Visibility = Visibility.Collapsed;
        ContentGrid.Visibility = Visibility.Visible;
      }
      else
      {
        MessageBox.Show("Application Closing");

        somaAd.Dispose();
      }
    }
```

Methods	Type	Summary
SomaAd	void	Constructor for instantiating a SomaAd object
GetAd	Void	Start the Get Ads from SOMA server process
Dispose	Void	Dispose of the SomaAd object

Event	Type	Summary
GetAdError	OnGetAdError	Notified when an ad is encountered while retrieving an ad from the SOMA Server
NewAdAvailable	OnNewAdAvailable	Notified when a new ad is available from the SOMA Server

Property	Type	Summary
AdImage	BitmapImage	NO LONGER SUPPORTED - since a BitMapImage only supports JPEG and PNG and most ads are GIF, this makes no sense to use The AdImage is in Isolated Storage in the file AdImageFileName ad image if adType is image - set by SDK; read only for developer
AdImageFileName	string	name of file containg the ad image if adType is image - set by SDK; read only for developer
AdImageUri	string	image URI - set by SDK; read only for developer
AdInterval	int	interval in milliseconds between ads; minimum 60000
Adspace	int	AD Space ID assigned by Smaato - ID "0" can be used for unattended testing
AdSpaceHeight	int	Height of ad space - On Mobile the screen height can be used
AdSpaceWidth	int	Width of ad space - On Mobile the screen height can be used
AdText	string	ad text if adType is text - set by SDK; read only for developer
AdType	string	ad type - image or text - set by SDK; read only for developer
Age	int	Targeting parameter - age of the user
City	string	Targeting parameter - location of the user - set by SDK to current location - can be overriden by developer - Example: redwood+city
ContentType	string	Type of content returned from SOMA Server
Coppa	bool	True indicates that content should be treated as child directed for COPPAA purposes
Country	string	Targeting parameter - location of the user - set by SDK to current location

		- can be overriden by developer - Example: united+states
Countrycode	string	Targeting parameter - location of the user - set by SDK to current location - can be overriden by developer - Format according to ISO-3166-1 (2 digit) - Example: us
Debug	bool	Debug True causes email to be sent to Smaato if there is an error
ErrorCode	string	error code – see ErrorCodes enum
ErrorDescription	string	error description
Format	FormatRequested	**Format** – control which format ads are requested • **All**: image, text or rich media ads; • **Img**: image ads only; • **Txt**: text ads only; • **Richmedia**: Rich Media ads only
Gender	string	Targeting parameter - gender of the user - m or f
Gps	string	Targeting parameter - ps coordinates of the users location - set by SDK to current location using cell tower triangulation - can be overriden by developer - latitude and longitude in decimal degrees format - comma (%2C) separated - Example - 37.530676%2C-122.262447
ImageOK	bool	ImageOK is true when the image in AdImageFileName is ready to use; required for XNA SAMPLE
Kws	string	Targeting parameter - keywords describing the content (e.g. automotive news) - comma separated values - example: motorsport, news, cars
LocationUseOK	bool	when true, current location will be used as a parameter for obtaining ads.
ModifyRM	Bool	Determines whether Rich Media ads can contain MRAID compliant ads
Pub	int	Publisher assigned by Smaato - ID "0" can be used for unattended testing
Qs	string	Targeting parameter - search string ((e.g. Madonna ringtone) - comma separated values - example: beyonce, ringtones, fast, food, football
State	string	Targeting parameter - location of the user state - set by SDK to current location - can be overriden by developer
Status	string	status - error means get ad failed

		errorCode contains code for failure errorDescription contains description of reason for failure
Uri	string	click thru action uri; available when ad is available - set by SDK; read only for developer
Zip	string	Targeting parameter - postal code of the current location of the user - set by SDK to current location - can be overriden by developer

Using the SOMA Interstitial Ad Class – SomaInterstitialAd

In order to use the SOMA Interstitial Ad Class for Windows Phone 8, first add a reference for SOMAWP8.DLL to your Windows Phone 8 Silverlight project. Then add a Using to the module in which you want to get ads.

Next instantiate the SomaInterstitialAd class like this:

```
SomaInterstitialAd interstitialAd;
interstitialAd = new SomaInterstitialAd();
```

Then add your PubID and AdSpaceID like this:

```
interstitialAd.Adspace = 0;
interstitialAd.Pub = 0;
```

Next, set whatever demographics you want which can be changed on the fly for each ad:

```
// demographics
somaAd.Age = 30;
somaAd.Gender = "m";
somaAd.City = "Fremont";
somaAd.State = "Ca";
somaAd.Zip = "94539";
somaAd.Country = "United States";
somaAd.Countrycode = "us";
```

Now you need to add an Event to receive notification when a new ad is available:

```
interstitialAd.NewAdAvailable += interstitialAd_NewAdAvailable;
```

When you want to retrieve an interstitial ad invoke the LoadInterstitial() method:

```
interstitialAd.LoadInterstitial();
```

You need a Delegate to receive the ad available notification. This Delegate is triggered when the ad has been retrieved from SOMA and is ready to display. When you are ready to display the ad use the ShowInterstitial() method.

```
void interstitialAd_NewAdAvailable(object sender, EventArgs e)
{
    // display the ad
    interstitialAd.ShowInterstitial();
}
```