Windows 10

Hands-on lab

Lab: Integrating Ads

October 2015

Microsoft

CONTENTS

Overview

The Universal Ad Client SDK is easy to integrate and gives you the ability to promote and monetize your app in markets around the world. Microsoft Advertising and Windows ad mediation have now been merged into a single SDK that supports Windows 10, Windows Phone 8.x, and Windows 8.x.

In this lab, you will install the Microsoft Universal Ad Client SDK, which includes the Microsoft Advertising SDK for XAML. The Microsoft Advertising libraries for XAML/JavaScript can be referenced separately from the Microsoft Advertising Universal SDK in the Visual Studio Reference Manager. For more information, visit https://msdn.microsoft.com/en-us/library/mt313199(v=msads.30).aspx.

We will use the Advertising SDK to implement interstitial and inline ads and leverage the demo ads made available by Microsoft.

Objectives

This lab will show you how to:

- Install the Microsoft Universal Ad Client SDK
- Introduce an interstitial ad to the app
- Add an inline ad to the app content

System requirements

You must have the following to complete this lab:

- Microsoft Windows 10
- Microsoft Visual Studio 2015

Setup

You must perform the following steps to prepare your computer for this lab:

1. Install Microsoft Windows 10.

2. Install Microsoft Visual Studio 2015.

3. Install the Microsoft Universal Ad Client SDK.

Instructions for installing the Microsoft Universal Ad Client SDK are in Exercise 1: Task 2.

Exercises

This Hands-on lab includes the following exercises:

1. Introducing Ads to the App

Estimated time to complete this lab: **15 to 30 minutes**.

Exercise 1: Introducing Ads to the App

In this exercise, you will install the Microsoft Universal Ad Client SDK and use it to add interstitial and inline ads to your app.

Task 1 – Create a blank Universal Windows app

We will begin by creating a project from the Blank App template.

1. In a new instance of Visual Studio 2015, choose **File > New> Project** to open the New Project dialog. Navigate to **Installed > Templates > Visual C# > Windows > Universal** and select the **Blank App (Universal Windows)** template.

2. Name your project **Advertising** and select the file system location where you will save your Hands-on Lab solutions. We have created a folder in our **C:** directory called **HOL** that you will see referenced in screenshots throughout the labs.

   Leave the options selected to **Create new solution** and **Create directory for solution**. You may deselect both **Add to source control** and **Show telemetry in the Windows Dev Center** if you don't wish to version your work or use Application Insights. Click **OK** to create the project.

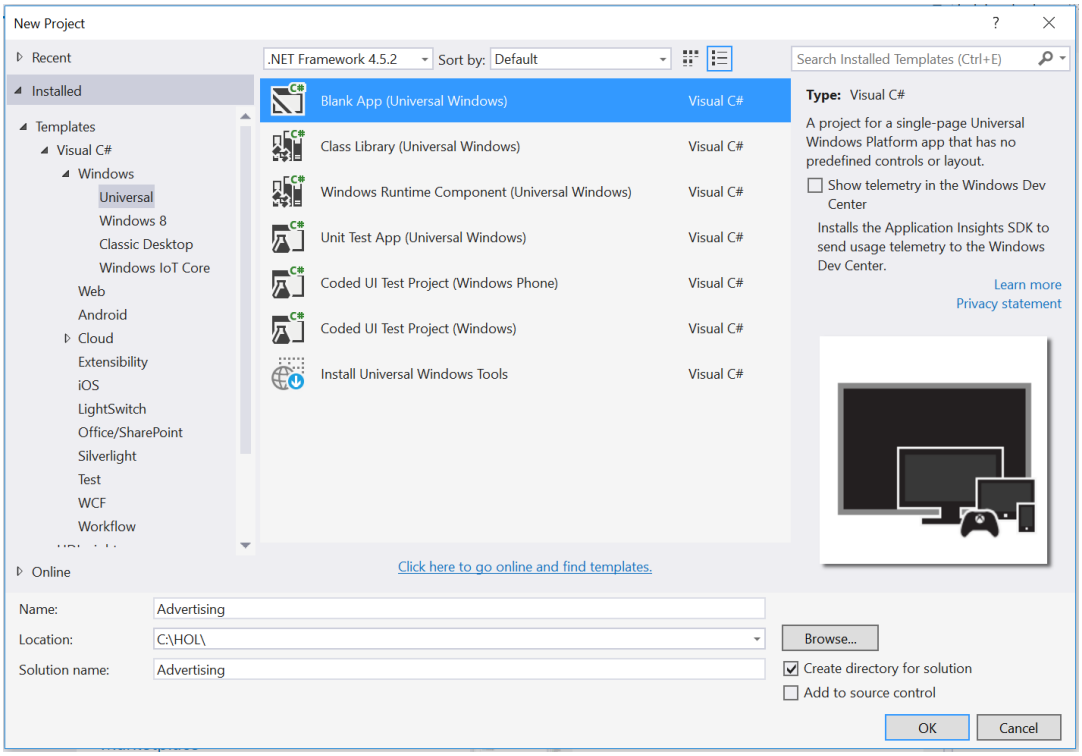

Figure 1

Create a new Blank App project in Visual Studio 2015.

3. Set your Solution Configuration to **Debug** and your Solution Platform to **x86**. Select **Local Machine** from the Debug Target dropdown menu.

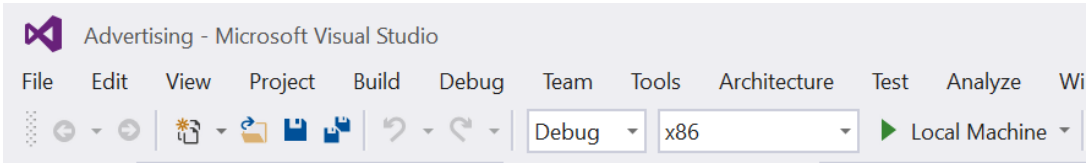

Figure 2

Configure your app to run on the Local Machine.

4. Build and run your app. You will see a blank app window with the frame rate counter enabled by default for debugging.

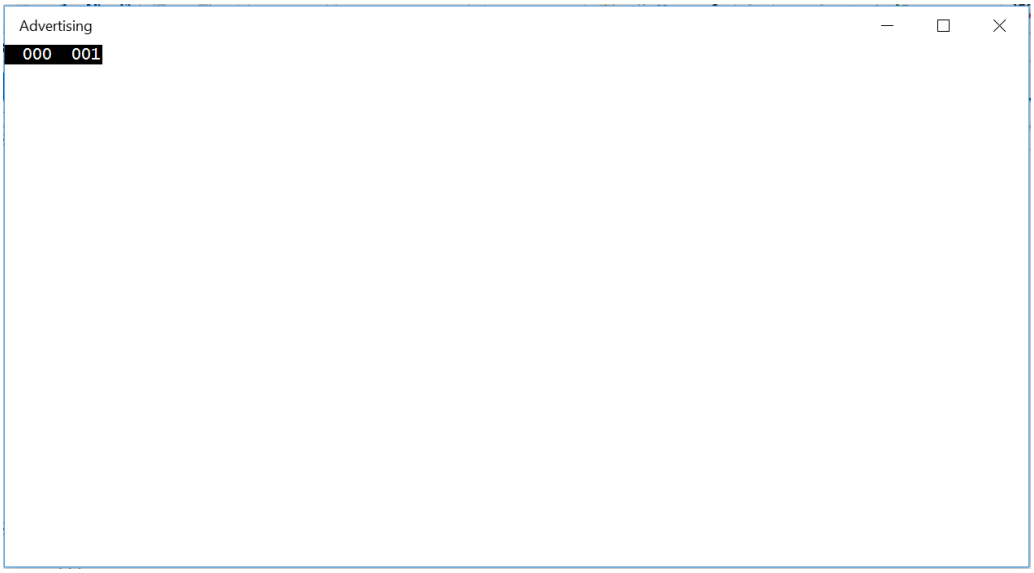

Figure 3

The blank universal app running in Desktop mode.

Note: The frame rate counter is a debug tool that helps to monitor the performance of your app. It is useful for apps that require intensive graphics processing but unnecessary for the simple apps you will be creating in the Hands-on Labs.

In the Blank App template, the preprocessor directive to enable or disable the frame rate counter is in **App.xaml.cs**. The frame rate counter may overlap or hide your app content if you leave it on. For the purposes of the Hands-on Labs, you may turn it off by setting **this.DebugSettings.EnableFrameRateCounter** to **false**.

5. Return to Visual Studio and stop debugging.

Task 2 – Install the Microsoft Universal Ad Client SDK

Before adding ads to your app, you will need to install the Universal Ad SDK.

Note: The Microsoft Universal Ad Client SDK includes the Microsoft Advertising SDK for XAML.

1. In Visual Studio, open the **Tools > Extensions and Updates** dialog.
2. Browse to the **Online** section in the menu and use the search box to search for **universal ad sdk**.

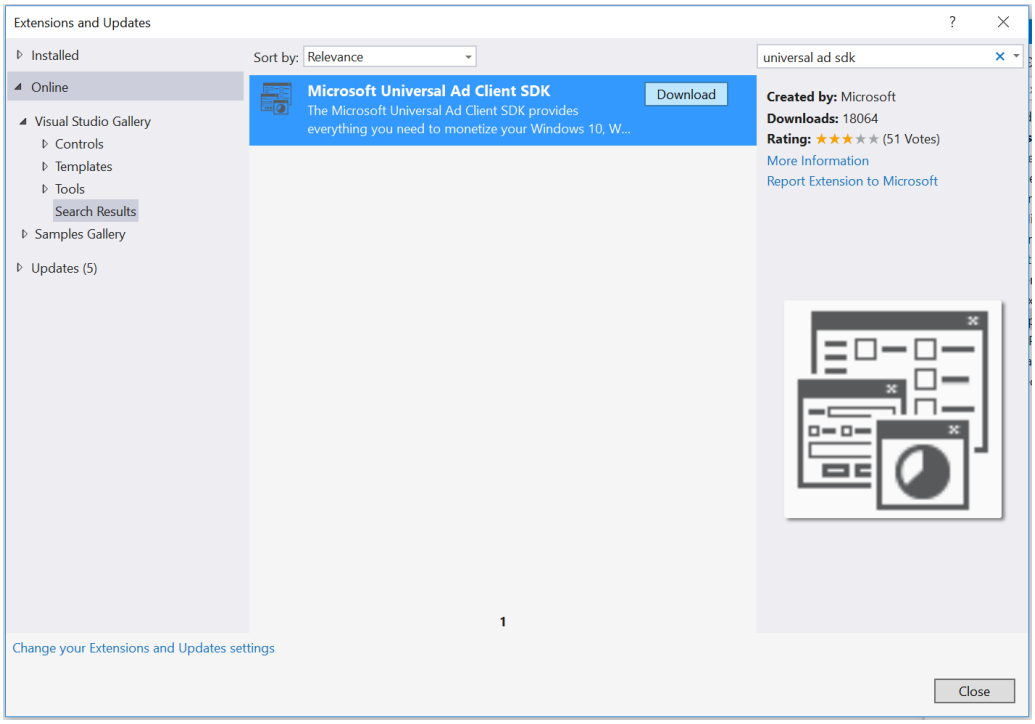

Figure 4

The Windows ad mediation extension in the Extensions and Updates dialog.

3. Select the **Microsoft Universal Ad Client SDK** extension and choose **Download**. Your browser will launch to download the installation file.

4. Close all instances of Visual Studio.

5. Run the **msi** installation file when prompted by the browser. When the Universal Ad SDK setup launches, install it using the default options. If prompted by the User Account Control, select **Yes** to allow the app to install software on your PC. When the installation is complete, use the **Finish** button to exit the setup wizard.

Figure 5

The Universal Ad Client SDK Setup Wizard.

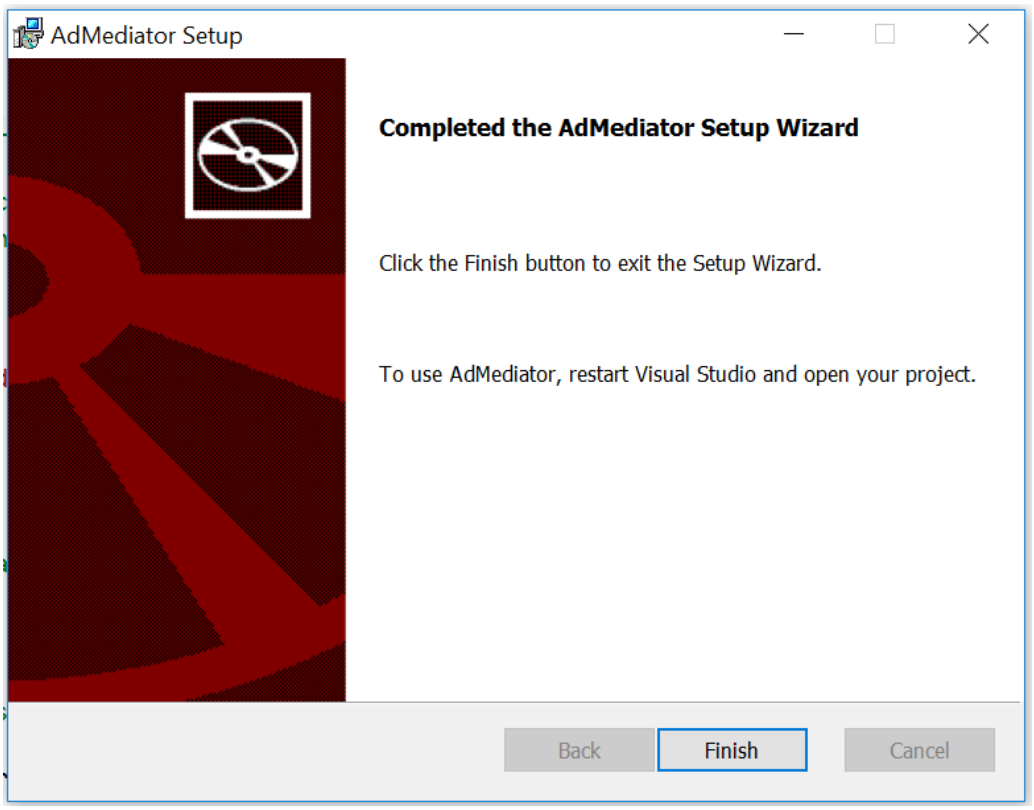

Figure 6

Complete the Universal Ad Client SDK installation.

6. Open Visual Studio and load the Advertising project you created in Task 1. When the project is open, right-click References in the Solutions Explorer and choose **Add Reference**.

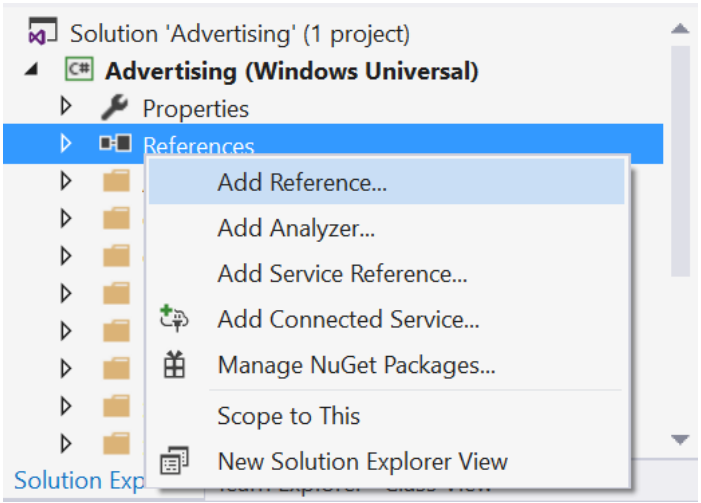

Figure 7

Navigate to the Add Reference dialog.

7. Expand the **Universal Windows** section and select **Extensions**. You will see a list of SDKs applicable to your project. Check the box next to the **Microsoft Advertising SDK for XAML** to select it and click OK to add it to the project as a reference. Be careful to select the correct SDK!

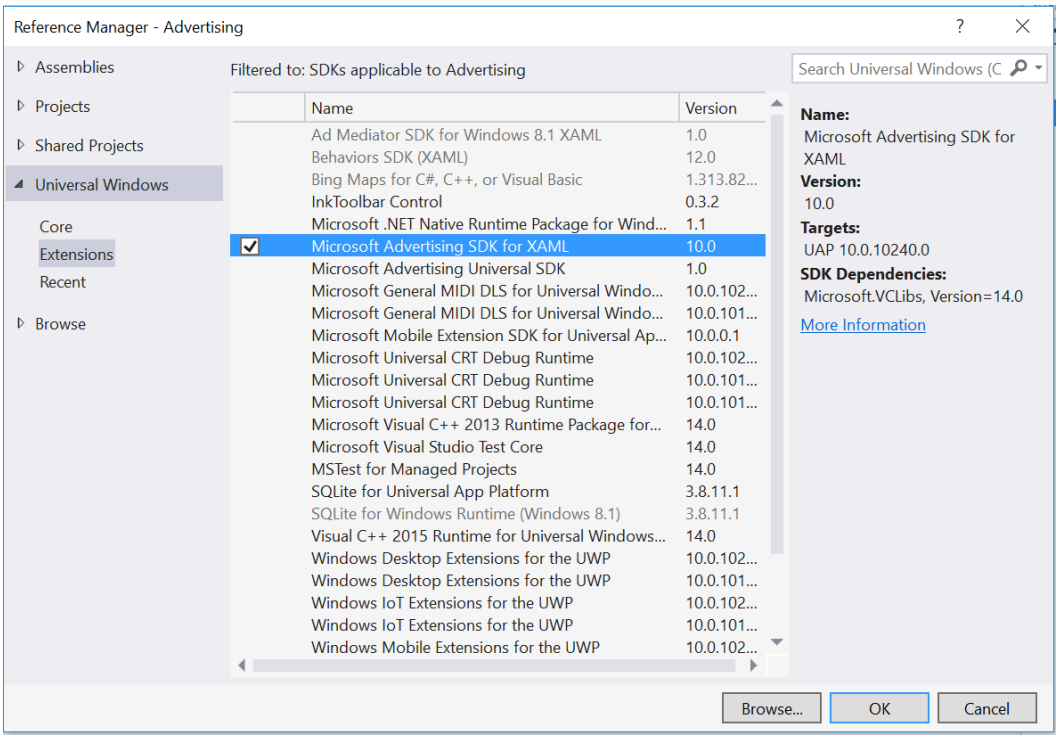

Figure 8

Add the Microsoft Advertising SDK for XAML as a project reference.

8. When the Add Reference dialog closes, you will see the Microsoft Advertising SDK appear in the list of project references.

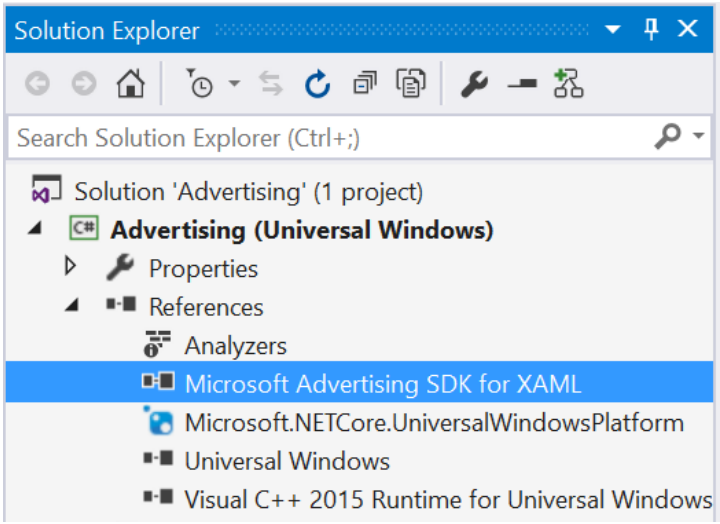

Figure 9

Add the Microsoft Advertising SDK for XAML as a project reference.

Task 3 – Add an interstitial ad

Now that you have referenced the Microsoft Advertising SDK in the Advertising project, you can begin to integrate ads into your app. In this task, you will create a new class called DemoAds, which will use test AppIds and AdUnits provided by Microsoft to display an interstitial ad in your app.

1. Right-click on the project name in the Solution Explorer and **Add > New Folder**. Name the folder **Models**.

2. To create the new **DemoAds** class, right-click on the **Models** folder and choose **Add > New Item**. When the **Add New Item** dialog appears, select Visual C# Class as your new item type (Figure 17). Give the class the name **DemoAds.cs**.

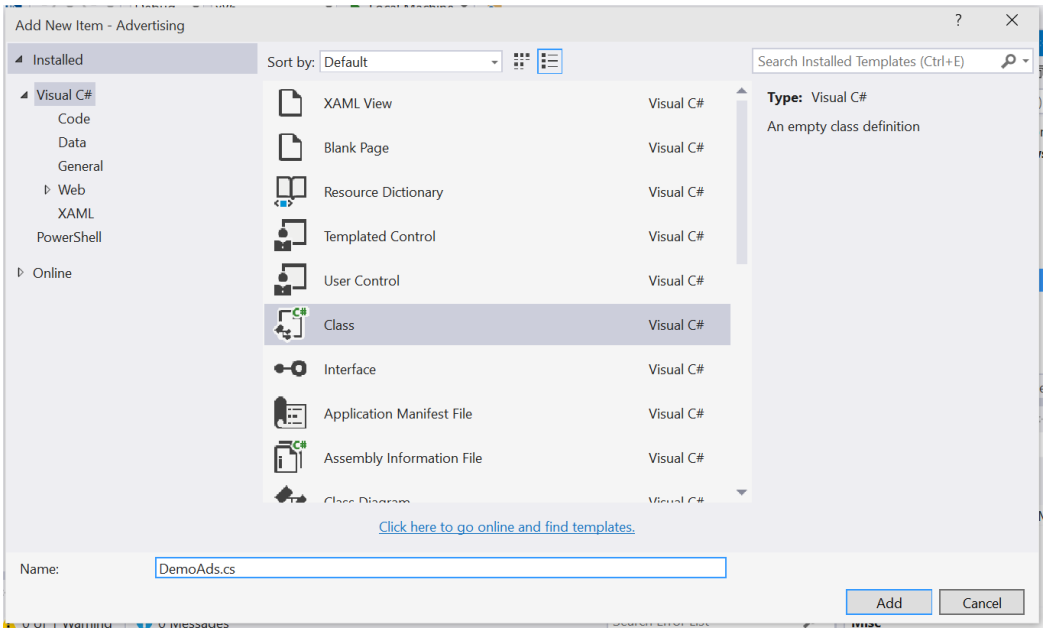

Figure 10

Create a new Visual C# class called DemoAds.

3. Open **DemoAds.cs**. In this step, you will replace the empty class definition with working DemoAds and AdUnit classes. The code block below shows the starting point that Visual Studio provides for you.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace Advertising.Models
{
   class DemoAds
   {
   }
}
```

Replace the empty class definition in the previous code block with the following class definitions shown in red. Save DemoAds.cs.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Advertising.Models
{


   /*
       These demo ad values are drawn from: https://msdn.microsoft.com/en-
US/library/mt125365(v=msads.100).aspx
   */
   public static class DemoAds
   {
       public static Dictionary<string, AdUnit> ImageAdUnits { get; private
set; }
       public static AdUnit VideoAdUnit { get; private set; }

       static DemoAds()
       {
           ImageAdUnits = new Dictionary<string, AdUnit>();

           ImageAdUnits.Add("300 x 50",
               new AdUnit { Size = "300 x 50", AdUnitId = "10865275", AppId =
"3f83fe91-d6be-434d-a0ae-7351c5a997f1" });
           ImageAdUnits.Add("320 x 50",
               new AdUnit { Size = "320 x 50", AdUnitId = "10865270", AppId =
"3f83fe91-d6be-434d-a0ae-7351c5a997f1" });
           ImageAdUnits.Add("300 x 250",
               new AdUnit { Size = "300 x 250", AdUnitId = "10043121", AppId =
"d25517cb-12d4-4699-8bdc-52040c712cab" });
```

```
            ImageAdUnits.Add("300 x 600",
                new AdUnit { Size = "300 x 600", AdUnitId = "10043122", AppId =
"d25517cb-12d4-4699-8bdc-52040c712cab" });
            ImageAdUnits.Add("480 x 80",
                new AdUnit { Size = "480 x 80", AdUnitId = "10865272", AppId =
"3f83fe91-d6be-434d-a0ae-7351c5a997f1" });
            ImageAdUnits.Add("640 x 100",
                new AdUnit { Size = "640 x 100", AdUnitId = "10865273", AppId =
"3f83fe91-d6be-434d-a0ae-7351c5a997f1" });
            ImageAdUnits.Add("728 x 90",
                new AdUnit { Size = "728 x 90", AdUnitId = "10043123", AppId =
"d25517cb-12d4-4699-8bdc-52040c712cab" });

            VideoAdUnit = new AdUnit { Size = "Video", AdUnitId = "11389925",
AppId = "d25517cb-12d4-4699-8bdc-52040c712cab" };
        }
    }

    public class AdUnit
    {
        public string Size { get; set; }
        public string AdUnitId { get; set; }
        public string AppId { get; set; }
    }
}
```

Note: This class references a set of demo ads that have been created and made available by Microsoft for the purpose of testing live ads in your apps. These ads were drawn from https://msdn.microsoft.com/en-US/library/mt125365(v=msads.100).aspx

4. Open **MainPage.xaml.cs**. Add properties for **ShowAds** and **ViewedFullInterstitial**. At first, we will require users to watch the full ad before continuing to the app.

C#

```
public sealed partial class MainPage: Page
{
        bool _showAds = true;
        public bool ShowAds
        {
          get { return _showAds; }
          set { _showAds = value; }
        }

        bool _viewedFullInterstitial = true;
        public bool ViewedFullInterstitial
        {
          get { return _viewedFullInterstitial; }
```

```
        set { _viewedFullInterstitial = value; }
    }
```

5. Add the **Microsoft.Advertising.WinRT.UI** and **Advertising.Models** namespaces to reference DemoAds and the Microsoft Advertising UI.

C#

```
using Microsoft.Advertising.WinRT.UI;
using Advertising.Models;

namespace Advertising
{
```

6. After the class definition, add a private InterstitialAd field.

C#

```
namespace Advertising
{
   public sealed partial class MainPage : Page
   {
       private InterstitialAd _interstitialAd;

       bool _showAds = true;
```

7. Initialize the interstitial class and wire up the AdReady, Cancelled, Completed, and ErrorOccurred events in the constructor.

C#

```
        public MainPage()
        {
            this.InitializeComponent();

            if (ShowAds)
            {
                // initialize the interstitial class
                _interstitialAd = new InterstitialAd();

                // wire up all 4 events
                _interstitialAd.AdReady += interstitialAd_AdReady;
                _interstitialAd.Cancelled += interstitialAd_Cancelled;
                _interstitialAd.Completed += interstitialAd_Completed;
                _interstitialAd.ErrorOccurred += interstitialAd_ErrorOccurred;

                RequestAd();
            }
            else
            {
                // start normally
```

```
            }
        }
```

8. Create the AdReady, Cancelled, Completed, and ErrorOccurred events below the constructor.

C#

```
                RequestAd();
            }
            else
            {
                // start normally
            }
        }

        private void interstitialAd_ErrorOccurred(object sender,
AdErrorEventArgs e)
        {
            // handle errors here
        }

        private void interstitialAd_Completed(object sender, object e)
        {
            // raised when the user has watched the full video
        }

        private void interstitialAd_Cancelled(object sender, object e)
        {
            // raised if the user interrupts the video
        }

        private void interstitialAd_AdReady(object sender, object e)
        {
            // raised when an ad is ready to show
        }
```

9. Add the definition for the **RequestAd** event under the constructor. This handler requests a video ad from the set of demo ads you referenced in the **DemoAds** class.

C#

```
                RequestAd();
            }
            else
            {
                // start normally
            }
        }
```

```
        private void RequestAd()
        {
            _interstitialAd.RequestAd(AdType.Video, DemoAds.VideoAdUnit.AppId,
DemoAds.VideoAdUnit.AdUnitId);
        }

        private void interstitialAd_ErrorOccurred(object sender,
AdErrorEventArgs e)
        {
```

Note: For the purposes of this lab, we will display the interstitial as soon as the ad is ready by adding it to the AdReady event handler.

10. Add the definition for the **ErrorOccurred** event. Be sure to add **async** to the event handler to accommodate the await.

C#

```
         private async void interstitialAd_ErrorOccurred(object sender,
AdErrorEventArgs e)
        {
            // handle errors here
            var dialog = new ContentDialog
            {
                Title = "An Error",
                Content = e.ErrorMessage,
                PrimaryButtonText = "OK",
                IsPrimaryButtonEnabled = true
            };

            await dialog.ShowAsync();
        }
```

Note: An enterprise-level application will require more robust error handling than we provide here.

11. Add the definition for the AdReady event.

C#

```
        private void interstitialAd_AdReady(object sender, object e)
        {
            //raised when an ad is ready to show

            if (_interstitialAd.State == InterstitialAdState.Ready)
            {
                _interstitialAd.Show();
            }
        }
```

12. Build and run the app. The interstitial video ad will play when the app loads.

Task 4 – Require the ad

Let's require that the user watch an entire video ad before proceeding.

1. We will leverage the Cancelled event by displaying a message to the user and restarting the ad.

C#

```
        private void interstitialAd_Completed(object sender, object e)
        {
            // raised when the user has watched the full video
            ViewedFullInterstitial = true;

        }

        private async void interstitialAd_Cancelled(object sender, object e)
        {
            // raised if the user interrupts the video
            var dialog = new ContentDialog
            {
                Title = "Ad Interrupted",
                Content = "You must watch the complete ad!",
                PrimaryButtonText = "OK",
                IsPrimaryButtonEnabled = true
            };

            await dialog.ShowAsync();

            RequestAd();
        }
```

Note: We are going to use a ContentDialog control to display the error message via an async method, so we must be sure to add async to the method definition of the interstitialAd_Cancelled event handler. The ContentDialog is a new control in Windows 10 that makes it easier to display rich content via an app modal dialog. For more on ContentDialogs, visit https://msdn.microsoft.com/library/windows/apps/windows.ui.xaml.controls.contentdialog.aspx

2. Build and run the app. Click on the running video to display a back button. Select the back button while viewing the ad to see the behavior of the interstitialAd_Cancelled event.

3. Stop debugging and return to Visual Studio.

4. Moving forward, it would be inconvenient to be unable to skip the ad during the remainder of this exercise. To give yourself the option to skip the ad, comment out the contents of the cancelled event handler.

C#

```
        private async void interstitialAd_Cancelled(object sender, object e)
```

```
        {
            // raised if the user interrupts the video
            //var dialog = new ContentDialog
            //{
            //    Title = "Ad Interrupted",
            //    Content = "You must watch the complete ad!",
            //    PrimaryButtonText = "OK",
            //    IsPrimaryButtonEnabled = true
            //};

            //await dialog.ShowAsync();

            //RequestAd();
        }
```

5. Build and run the app again. Confirm that you can skip the ad by selecting the back button while it is playing.

6. Stop debugging and return to Visual Studio.

Task 5 – Show an inline ad

In this task, you will use the Microsoft Advertising SDK to display an inline ad in a separate column in your app. The same techniques can be used to place ads inline within other layout controls.

1. Open **MainPage.xaml** and add the **Microsoft.Advertising.WinRT.UI** namespace.

XAML

```
    xmlns:UI="using:Microsoft.Advertising.WinRT.UI"
```

2. To create a container for your inline ad, replace the **Grid** in **MainPage.xaml** with a **Hub**. Add an **Advertising** hub section and an adjacent **Content** hub section to the hub. The Advertising hub section will leverage the **AdControl** from the Advertising SDK to show a static image ad and will display to the left of the app content. In design view, this hub section will display a blue rectangle with the same dimensions as the ad. Set the visibility to collapsed as the default.

XAML

```
<Hub Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <HubSection
            VerticalContentAlignment="Stretch"
            x:Name="AdvertisingSection"
            Header="Advertising"
            Visibility="Collapsed">
            <DataTemplate>
                <Grid VerticalAlignment="Top">
                    <!-- The rectangle acts as a place holder so we can see
where the ad control is located-->
```

```
                    <Rectangle Fill="Blue" Width="300" Height="600"/>
                    <UI:AdControl
                        ApplicationId="d25517cb-12d4-4699-8bdc-52040c712cab"
                        AdUnitId="10043122"
                        Height="600"
                        VerticalAlignment="Top"
                        Width="300"/>
                </Grid>
            </DataTemplate>
        </HubSection>
        <HubSection Header="Content" />
</Hub>
```

3. In the MainPage code behind, add a **ShowInlineAds()** method to display the **AdvertisingSection** if **ShowAds** is true.

C#

```
private void ShowInlineAds()
{
    if (ShowAds)
    {
        AdvertisingSection.Visibility = Visibility.Visible;
    }
    else
    {
        AdvertisingSection.Visibility = Visibility.Collapsed;
    }
}
```

4. Call the **ShowInlineAds()** method after **RequestAd()** in the constructor.

C#

```
    RequestAd();
    ShowInlineAds();
}
```

5. Build and run your app. After the interstitial ad finishes playing, you will see the inline image ad appear to the left of your app content.

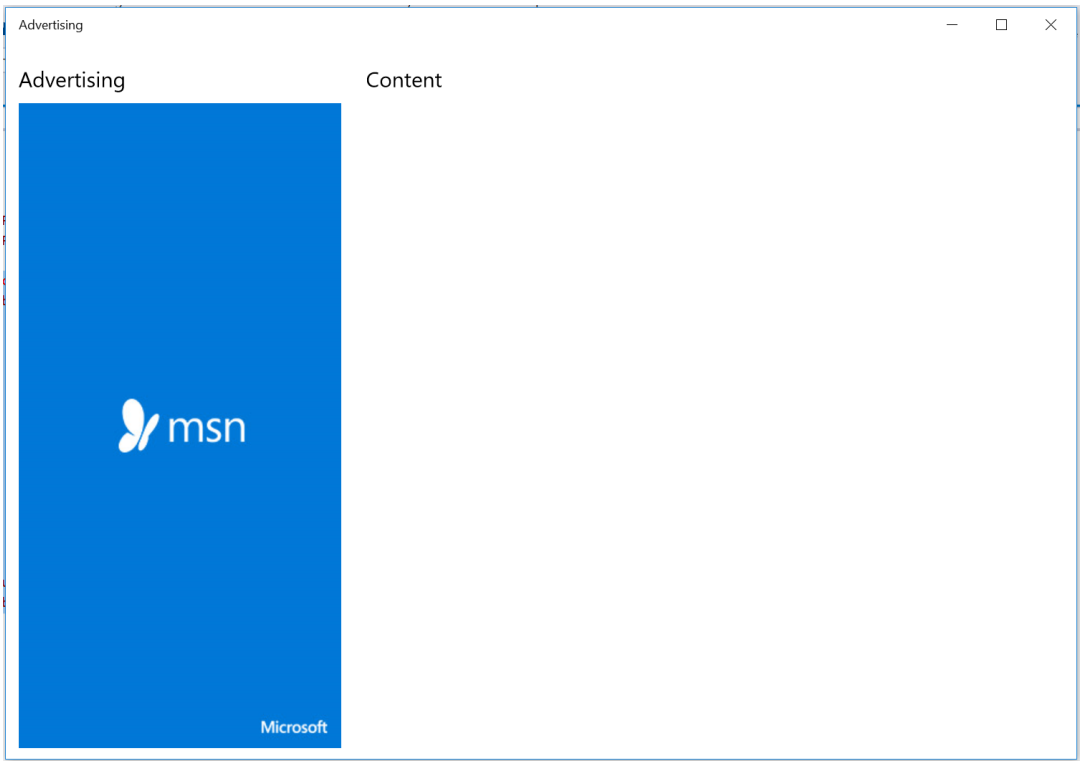

Figure 11

Inline ad in the Advertising app.

6. Stop debugging and return to Visual Studio.

Summary

In this lab, you downloaded and installed the Microsoft Universal Advertising Client SDK and displayed the new video Interstitial Ads. You learned how to require users to view the interstitial ad and added an image-based ad into a Hub control.