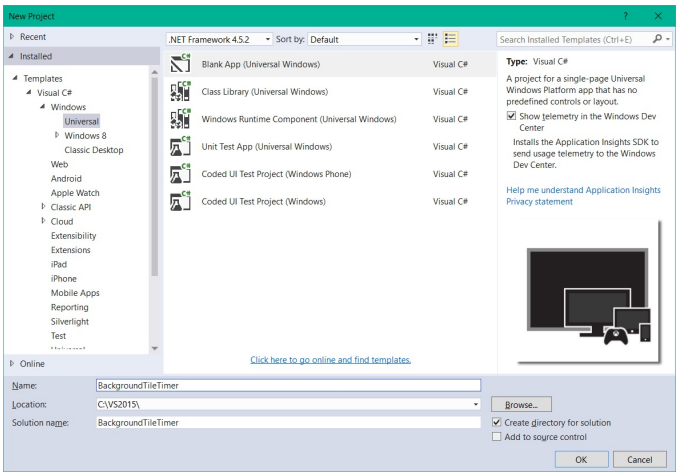
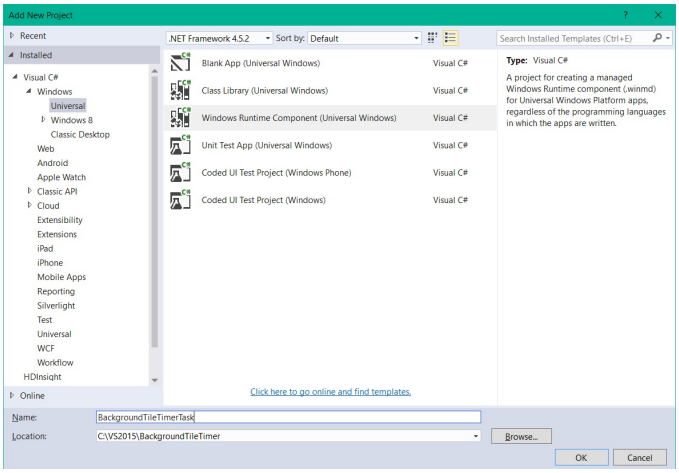
Windows Universal App Tile Updater Using a Background Timer

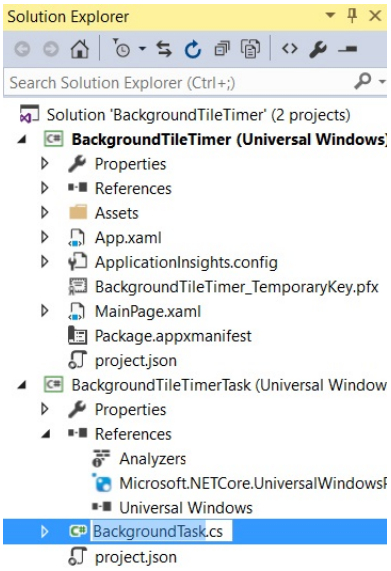
Step 1: Create a new Blank App (Universal Windows) project in Visual Studio 2015



Step 2: Add a new Windows Runtime Component (Universal Windows) project to the solution.



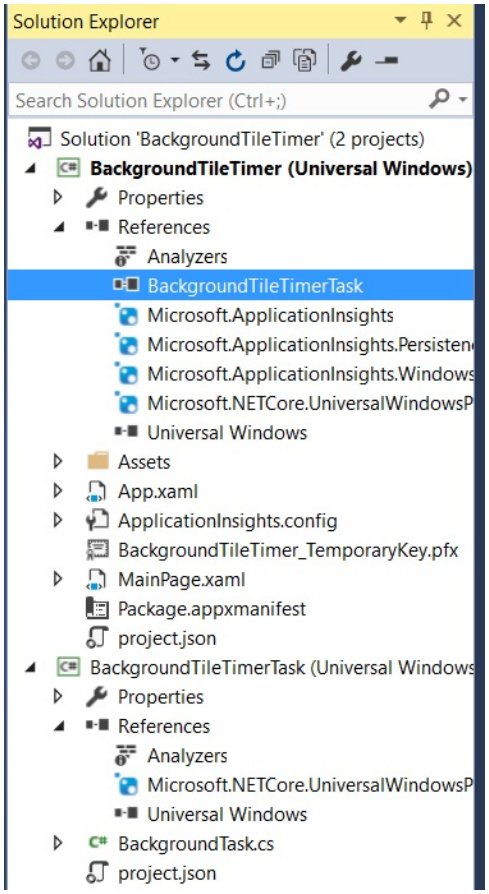
Step 3: Rename the Class1.cs file name as appropriate



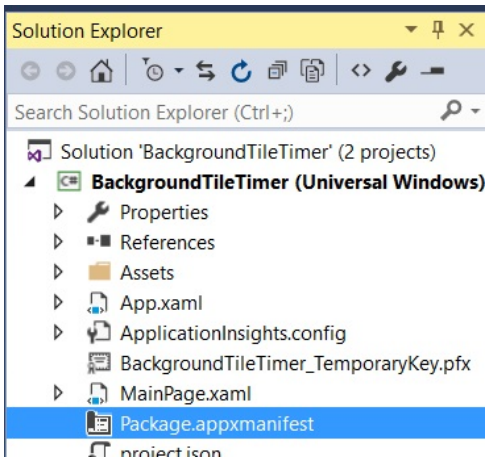
Step 4: Implement the IBackgroundTask interface. Note the class must be public and sealed. Shown is a sample implementation that updates the tile with the current time.

```
using System;
using System.Diagnostics;
using Windows.ApplicationModel.Background;
using Windows.Data.Xml.Dom;
using Windows.UI.Notifications;

namespace BackgroundTileTimerTask
{
    public sealed class BackgroundTask : IBackgroundTask
    {
        public void Run(IBackgroundTaskInstance taskInstance)
        {
            var backgroundTaskDeferral = taskInstance.GetDeferral();
            try{ UpdateTile(); }
            catch(Exception ex){ Debug.WriteLine(ex); }
            finally{ backgroundTaskDeferral.Complete(); }
        }

        private void UpdateTile()
        {
            var now = DateTime.Now.ToString("HH:mm:ss");
            var xml =
                "<tile>" +
                "  <visual>" +
                "    <bindingtemplate='TileSmall'>" +
                "      <text>" + now + "</text>" +
                "    </binding>"+
                "    <bindingtemplate='TileMedium'>" +
                "      <text>" + now + "</text>" +
                "    </binding>" +
                "    <bindingtemplate='TileWide'>" +
                "      <text>" + now + "</text>" +
                "    </binding>" +
                "    <bindingtemplate='TileLarge'>" +
                "      <text>" + now + "</text>" +
                "    </binding>" +
                "  </visual>" +
                "</tile>";

            var tileDom = new XmlDocument();
            tileDom.LoadXml(xml);
            var tile = new TileNotification(tileDom);
            TileUpdateManager.CreateTileUpdaterForApplication().Update(tile);
        }
    }
}
```

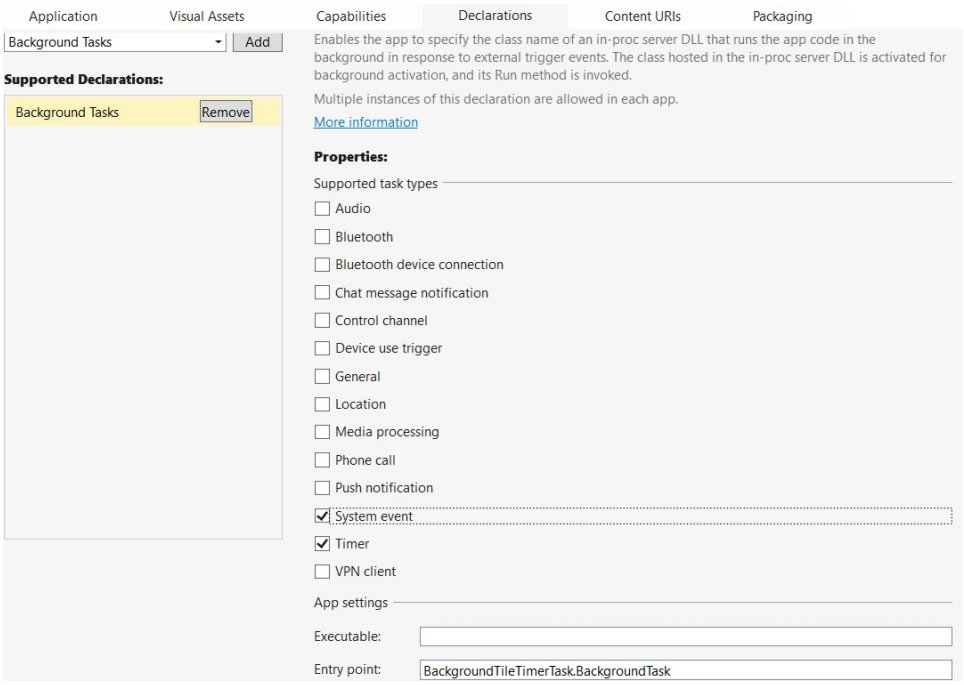
Step 5: Reference the background task project from the main project



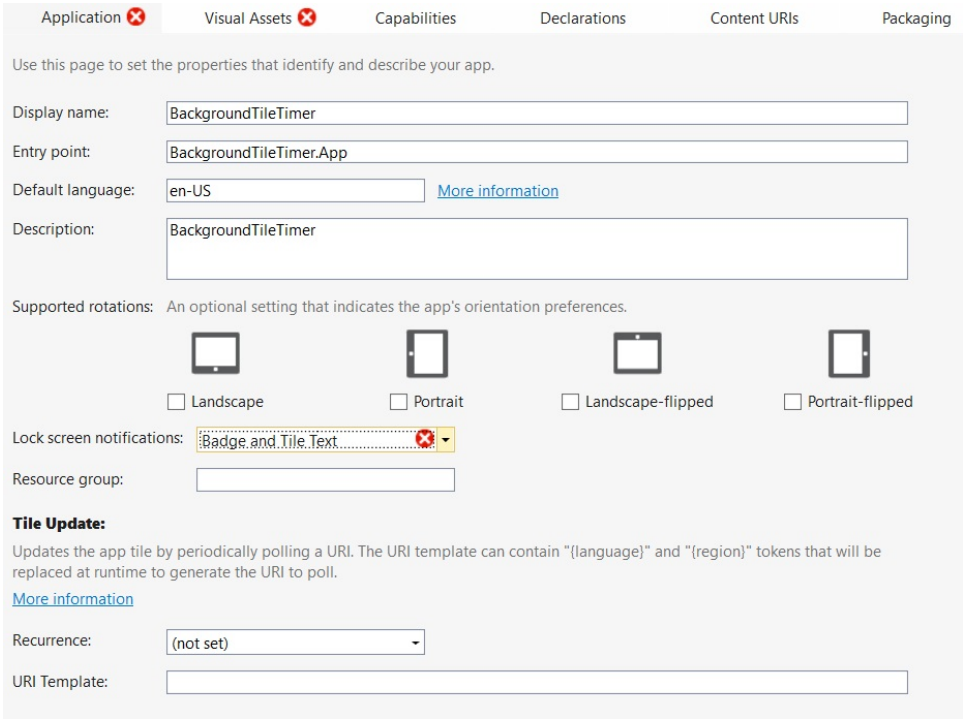
Step 6: Open the package app manifest file



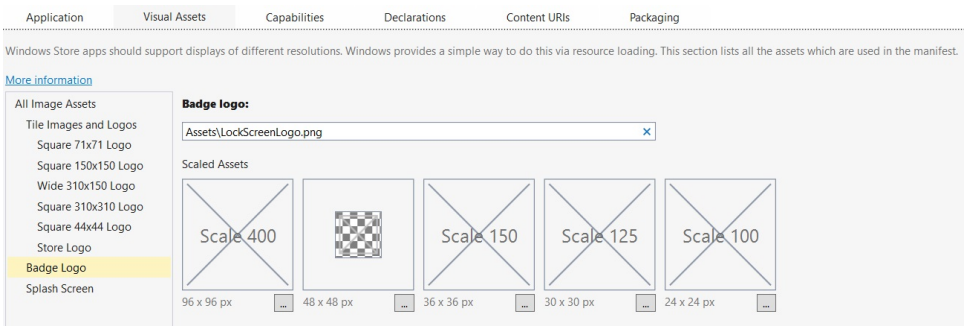
Step 7: Add a background task declaration. Select “System Event” and “Timer” for the type and supply a string for the Entry Point. Note that the entry point value must be the fully qualified type name (namespace and class name) for the class the implemented the IBackgroundTask interface:



Step 8: Set the application “Lock screen notifications” to “Badge and Tile Text”. Ignore the Red X for now



Step 9: Update the Badge Logo. Click the […] for the Scale 200 selection, navigate to the Assets folder and select the LockScreenLogo.png file. You can use another image file if you prefer, but VS is picky about the image size (and other stuff). This serves the purposes of this example.



Step 10: Add a couple of buttons to your MainPage.xaml file:

```
<Page
    x:Class="BackgroundTileTimer.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:BackgroundTileTimer"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <StackPanel Orientation="Horizontal" Height="50" VerticalAlignment="Top">
            <TextBlock Text="BackgroundTask" VerticalAlignment="Center"/>
            <Button x:Name="EnableButton" Click="Enable" Margin="4" VerticalAlignment="Center">Enable</Button>
            <Button x:Name="DisableButton" Click="Disable" Margin="4" VerticalAlignment="Center">Disable</Button>
        </StackPanel>
    </Grid>
</Page>
```

Step 11: Update the MainPage.xaml.cs file:

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Windows.ApplicationModel.Background;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace BackgroundTileTimer
{
     public sealed partial class MainPage : Page
     {
          private const string TASK_NAME = "TILE_UPDATE_TIMER_TASK_SAMPLE";

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

          private void ToggleButtons()
          {
              DisableButton.IsEnabled = GetTask();
              EnableButton.IsEnabled = !DisableButton .IsEnabled;
          }

          private async void Enable(object sender, RoutedEventArgs e)
          {
              try{ await RegisterTask(); ToggleButtons(); }
              catch(Exception ex){ ShowDialog(ex.ToString()); }
          }

          private void Disable(object sender, RoutedEventArgs e)
          {
              try{ UnregisterTask(); ToggleButtons();}
              catch(Exception ex){ ShowDialog(ex.ToString()); }
          }

          private async void ShowDialog(string message)
          {
              var dlg = new MessageDialog(message);
              await dlg.ShowAsync();
          }

          public async Task RegisterTask()
          {
              var backgroundAccessStatus = await BackgroundExecutionManager.RequestAccessAsync();
              if(backgroundAccessStatus == BackgroundAccessStatus.Denied){ return; }

              if(GetTask()){ return; }

              var timeTrigger = newTimeTrigger(15, false);

              var backgroundTaskBuilder = new BackgroundTaskBuilder();
              backgroundTaskBuilder.Name = TASK_NAME;
              backgroundTaskBuilder.TaskEntryPoint = typeof(BackgroundTileTimerTask.BackgroundTask).FullName;
              backgroundTaskBuilder.SetTrigger(timeTrigger);

              backgroundTaskBuilder.Register();
          }

          public void UnregisterTask()
          {
              var task = new KeyValuePair<Guid, IBackgroundTaskRegistration>();
              if(GetTask(ref task))
              {
                  task.Value.Unregister(true);
              }
          }

          public bool GetTask()
          {
              var task = new KeyValuePair<Guid, IBackgroundTaskRegistration>();
              return GetTask(ref task);
          }

          public bool GetTask(ref KeyValuePair<Guid, IbackgroundTaskRegistration> task)
          {
              foreach(var t in BackgroundTaskRegistration.AllTasks)
              {
                  if(t.Value.Name == TASK_NAME)
                  {
                      task = t;
                      return true;
                  }
              }
              return false;
          }
     }
}
```