How to create simple Background Agent in Windows Phone

This article provides a minimal example on how to create Background Agents (Tasks) in Windows Phone. There is a more comprehensive example on Dev Center How to implement background agents for Windows Phone.

Introduction

Windows Phone apps are made dormant when running in the background or when the phone is locked (to conserve device memory and battery life). If your application needs to perform processing when it is (may be) in the background you can use a background agent to run activities according to some schedule.

Background agents are divided into *Periodic Tasks* and *Resource-intensive Tasks*. *Periodic Task* can run for no more than 25 seconds every 30 minutes. This sort of agent is suitable for many common use-cases:

- Downloading data from remote server and then scheduling notifications (for example, when monitoring twitter or facebook updates)
- Changing lock screen wallpaper (like the Bing app)
- Synchronizing data in the background, for example emails

Apps which require more device resources and/or which do not require a fixed update schedule, can use a resource intensive task. These tasks run when the device is connected to external power and/or connected to a WiFi network - they are very useful when you need to transfer a lot of data or perform some other long running task in the background.

This article provides a very simple example of how to create a *Periodic Task*.

Tip: If you want to also understand Resource-intensive tasks then check out How to implement background agents for Windows Phone (Dev Center). This is more complicated to read, but is more comprehensive and covers both periodic and resource-intensive tasks.

Limitations

Background agents have certain limitations, as listed below.

- Background tasks can minimally be run every 30 minutes. There is a debug-only API to run them more regularly, but this is not available for released apps.
- There are certain APIs which are not supported , you can check here.
- Some low power devices do not support background agents
- Background tasks are limited by number on each device and can be enabled or disabled from application settings.
- They do not work when power saver mode is activated.

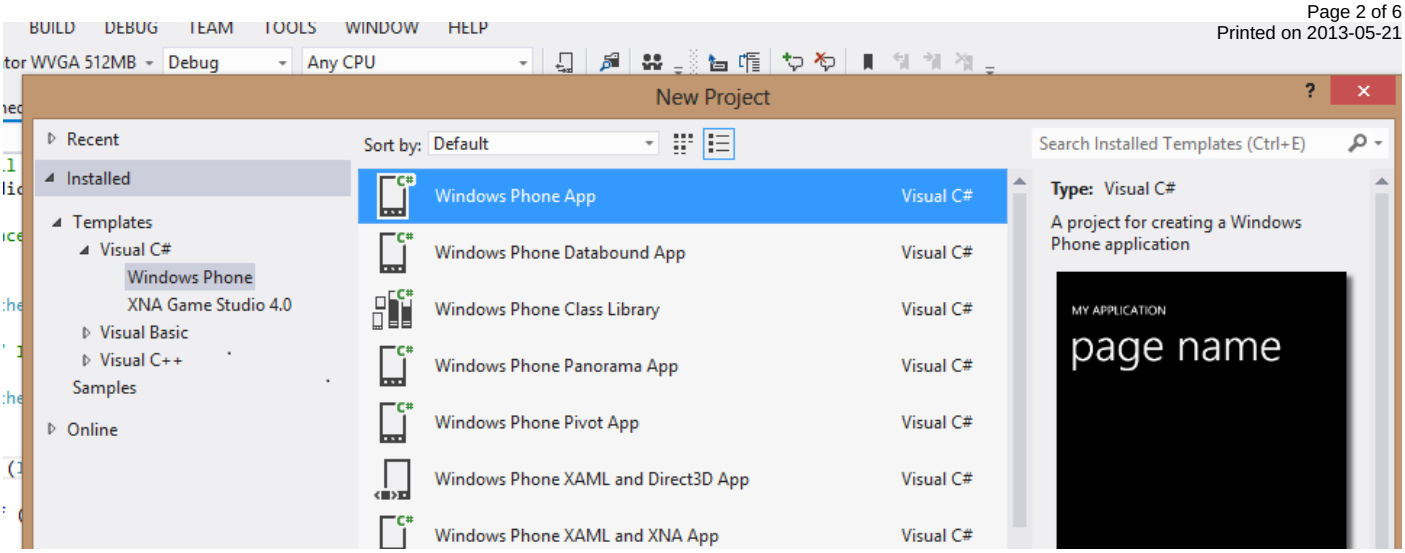
Basic requirements

- You need to create an app project, it can be anything from simple basic windows phone app to panorama or pivot based application.
- Then create your background agent project in the same solution and add its reference to the main app, with necessary code (which is hardly few lines) you can have your background agent running. You can test agents in the simulator as well.

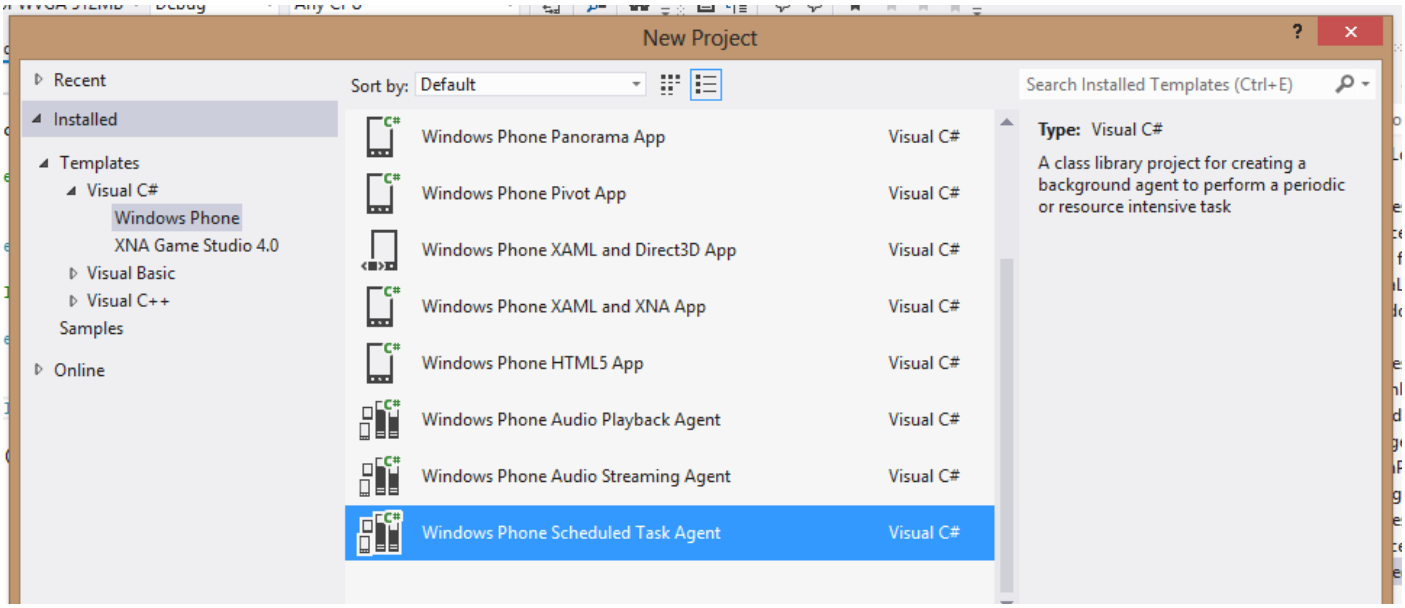
How to....

- Create a simple windows phone application project as follows and give some suitable name to it.

(C) Copyright Nokia 2013. All rights reserved.

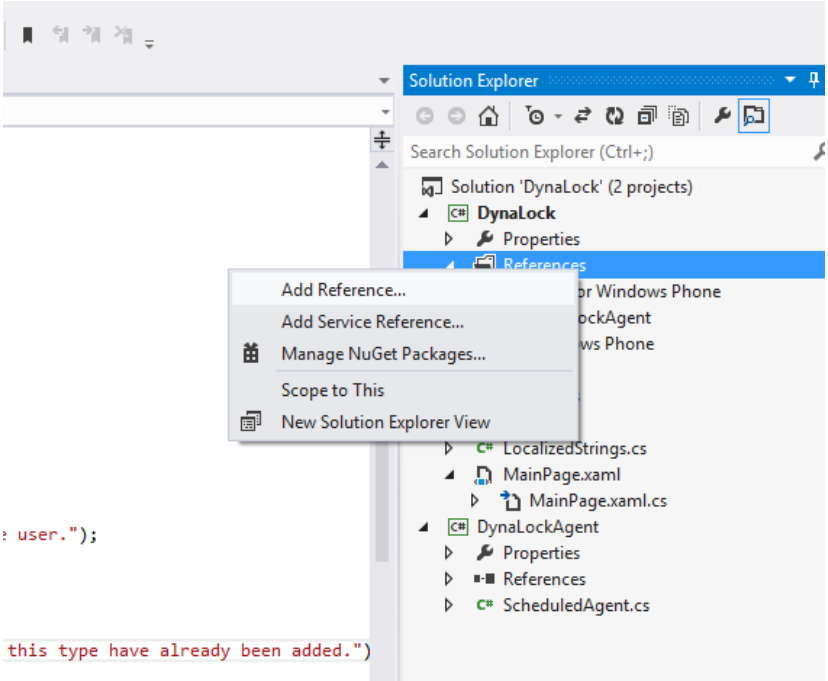


- Create a *scheduled task agent* project in Visual Studio, this should be added to the same solution. An easy way to do this is: goto solution explorer of main application, right click on solution and select add, select add project and automatically the project created will get added to the existing solution. Assign some suitable name to the project.

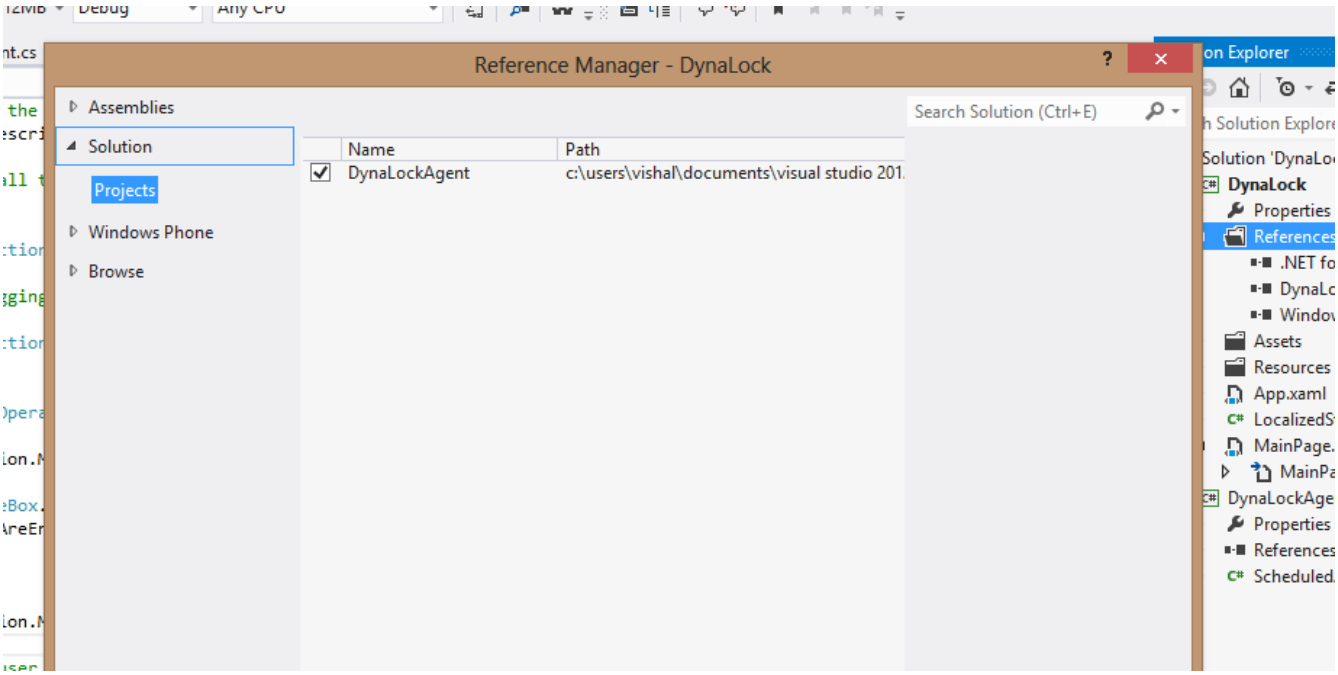


- The next step is to add a reference to the main application. This can be done by going to Solution Explorer, right clicking references under main project and selecting the Add Reference option.

(C) Copyright Nokia 2013. All rights reserved.



- Next select Solutions section on the Add Reference dialog and you should be able to see the name of your scheduled task agent project name there select it and its done.



- Next open **ScheduledAgent.cs** file from your background agent project and add following lines of code in `OnInvoke()` method

```
  protected override void OnInvoke(ScheduledTask task)
        {

            string toastMessage = "Periodic task running.";

            ShellToast toast = new ShellToast();
            toast.Title = "Background Agent Sample";
            toast.Content = toastMessage;
            toast.Show();
#if DEBUG_AGENT
  ScheduledActionService.LaunchForTest(task.Name,TimeSpan.FromSeconds(60));
#endif

            NotifyComplete();
         }
```

Once the above code is added the Scheduled Agent or your background task is ready to fire a Toast notification whenever its called. As you may tun this app to test, dont forget to add following *macro in at the top* in your **ScheduledAgent.cs** file

```
#define DEBUG_AGENT
```

and after the above macro make sure you have all following namespace imports in place

 (C) Copyright Nokia 2013. All rights reserved.

```
using System.Diagnostics;
using System.Windows;
using Microsoft.Phone.Scheduler;
using Microsoft.Phone.Shell;
using System;
```

- Now next step is to setup this Agent (task) to run in background from your main app, it can be done with the help of following lines of code. Make sure you place them in your **MainPage.xaml.cs** file of main project, I am attaching the complete file for your exposure to the code.

```
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using DynaLock.Resources;
using Microsoft.Phone.Scheduler;

namespace Application1
{
    public partial class MainPage : PhoneApplicationPage
    {
        PeriodicTask periodicTask;


        string periodicTaskName = "PeriodicAgent";
         // Constructor

        public MainPage()
        {
            InitializeComponent();

            // Sample code to localize the ApplicationBar
            //BuildLocalizedApplicationBar();
            agentsAreEnabled = true;

            // Obtain a reference to the period task, if one exists
            periodicTask = ScheduledActionService.Find(periodicTaskName) as PeriodicTask;

            if (periodicTask != null)
            {
                RemoveAgent(periodicTaskName);
            }

            periodicTask = new PeriodicTask(periodicTaskName);

            // The description is required for periodic agents. This is the string that the user
            // will see in the background services Settings page on the device.
            periodicTask.Description = "This demonstrates a periodic task.";

            // Place the call to Add in a try block in case the user has disabled agents.
            try
            {
                ScheduledActionService.Add(periodicTask);

                // If debugging is enabled, use LaunchForTest to launch the agent in one minute.
#if(DEBUG_AGENT)
                ScheduledActionService.LaunchForTest(periodicTaskName, TimeSpan.FromSeconds(10));
#endif
            }
            catch (InvalidOperationException exception)
            {
                if (exception.Message.Contains("BNS Error: The action is disabled"))
                {
                    MessageBox.Show("Background agents for this application have been disabled by the user.");
                    agentsAreEnabled = false;

                }

                if (exception.Message.Contains("BNS Error: The maximum number of ScheduledActions of this type have already been add
                {
                    // No user action required. The system prompts the user when the hard limit of periodic tasks has been reached.

                }

            }
            catch (SchedulerServiceException)
            {
                // No user action required.
            }


        }
        private void RemoveAgent(string name)
        {
            try
            {
                ScheduledActionService.Remove(name);
            }
            catch (Exception)
            {
            }
        }
    }
}
```

We call `LaunchForTest()` function only for debugging, which should be remove once your application goes into production environment.

Make sure to remove the DEBUG macro once you are done debugging of the project which will stop calling the agent after every few seconds, which should be done only if you wish to debug the application. If you keep that line as it is it may cause failure in app submission process.

Summary

This is how you can create the `SchedulesAgent` Task to run in background from using your own app. The example is simpler than that provided by Dev Center and therefore makes it easier to understand the basic usage.

(C) Copyright Nokia 2013. All rights reserved.

Add your feedback to comment section.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.