Dynamic Lock Screen for Windows Phone 8

This article explains how to change the Lock screen image of Windows Phone 8 using application that can be set in Lock screen setting. Going further, this article also explains how we can change the lock screen periodically with new image which can located on app folder or over internet. We will also see code demo for each method to set lock screen.

Note: This article was a winner in the Windows Phone 8 Wiki Competition 2012Q4.

Introduction

The Lockscreen in Windows Phone 8 is customisable - you can set the background image, text (from notifications) and also "counters" for up to 5 main apps (for example showing how many messages have arrived).

This article explains how to change the Lockscreen background image using LockScreenManager. We will also show you how to change the Lockscreen wallpaper in background periodically using Background Agent Scheduler.

The raw API for setting the background images only allows you to fetch an image from a local URI or the gallery. To make it interesting, the example gets the images from a number of places:

1. Images located on app folder
2. Images located on Internet (like Bing App)
3. Images coming from RSS

To provide the lock screen background image in the phone, we need to declare the app's intent in the app manifest file **WMAppManifest.xml**, and add code to handle changing the background image and then to change image periodically we need to use Windows Phone Scheduler as `Background Agent`.

In the last section we will show how to make your WP8 app enabled for lock screen notifications, to make the lock screen more dynamic.

Video Demo

Here you can see the app in action changing lock screen image every 30 sec. The media player is loading...

Here is the video of RSS demo application "Flicker RSS - Dynamic Lock screen". We will also going to create this app later. You can see application in action The media player is loading...

We are going to create these apps. So **let's get started**! First we will show you how to set a default image to lock screen by app acting as Background image provide.

Setting up a lock screen background image

To provide the lock screen background image on Windows Phone 8, first we need to update the app manifest file to declare your app as a lock screen background provider so that app can get permission to access the lock screen of the phone.

Updating the app manifest file

To Update the app manifest file so that app can access the lock screen of the phone, we need to update **WMAppManifest.xml** file as below.

1. In **Solution Explorer**, expand Properties, right-click **WMAppManifest.xml**, click **Open With**, and then click **Source Code (Text Editor) With Encoding**.
2. Add the lock screen background **<Extension>** element in the **<Extensions>** element. If the **<Extensions>** element doesn't appear in the file, place the entire code example below into the file. The **<Extensions>** element must be placed below the **<Tokens>** element.

Here in **<Extension>** element, we need to define **ExtensionName**, **ConsumerID** and **TaskID**. In your application, you can use same **ConsumerID** and **TaskID** as below.

```
<Extensions>
      <Extension ExtensionName="LockScreen_Background" ConsumerID="{111DFF24-AA15-4A96-8006-2BFF8122084F}" TaskID="_default" />
</Extensions>
```

Now users can see the app in the Lock Screen Settings and choose to let the app change the lock screen.

Setting the default lock screen

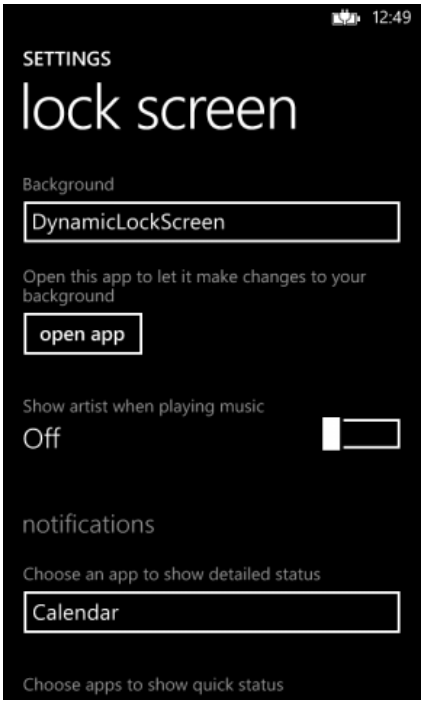
Every lock screen app must have a default image named **DefaultLockScreen.jpg** in the root of the main project's XAP package - this is set first as our app service agent will not get to set the image dynamically

The recommended lock screen image size for lock screen is **480px X 800px** or **960px X 1708px**. If larger image is provided it will be cropped from centre and set in lock screen: if a smaller image is provided then it will be stretched to fit the screen (and may be blurred/scratched).

(C) Copyright Nokia 2013. All rights reserved.

Adding code to change the Lock screen background

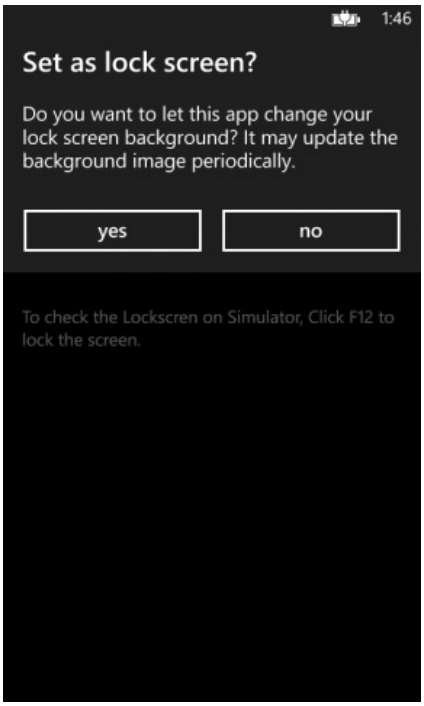
We can change background of Lock screen using `Windows.Phone.System.UserProfile.LockScreenManager` namespace. To change the background, our app must have selected as Lock screen background provider in Lockscreen setting. You can check this in **Setting -> Lock Screen**.



To check whether our app is selected as Lock screen image provider or not through code, we can use LockScreenManager.IsProvidedByCurrentApplication method. And if it return false, we use LockScreenManager.RequestAccessAsync() method to ask user to select our app as Provider.

```
if (!LockScreenManager.IsProvidedByCurrentApplication)
{
    // If you're not the provider, this call will prompt the user for permission.
    // Calling RequestAccessAsync from a background agent is not allowed.
    await LockScreenManager.RequestAccessAsync();
}

// Only do further work if the access is granted.
if (LockScreenManager.IsProvidedByCurrentApplication)
{
      // change the background here.
}
```

This code prompts the user to give permission to our app to access lock screen. Once user clicks on **Yes**, our app is granted to change the lock screen and become current Lock screen image provider.



Then, we need to call the LockScreen.SetImageUri method with the URI of the picture. The URI must be prefixed by `ms-appx:///` if the picture is stored in the resources, or `ms-appdata:///Local/` if the picture is stored in root folder.

(C) Copyright Nokia 2013. All rights reserved.

- `ms-appx:///` for resources bundled within the .xap
- `ms-appdata:///local/` for local images within the root folder.

We can not set image located on server using Image URL directly on LockScreen.SetImageUri method. But we do it by downloading image using `WebClient` and storing it on `IsolatedStorage` and then can use above `ms-appx:///` on LockScreen.SetImageUri method.

The following final code example shows how we can change the lock screen background image. We need to call below function. Here method uses `async` as'the user for permission can only be done by using an `async` call.

```
private async void LockScreenChange(string filePathOfTheImage, bool isAppResource)
{
    if (!LockScreenManager.IsProvidedByCurrentApplication)
    {
        // If you're not the provider, this call will prompt the user for permission.
        // Calling RequestAccessAsync from a background agent is not allowed.
        await LockScreenManager.RequestAccessAsync();
    }

    // Only do further work if the access is granted.
    if (LockScreenManager.IsProvidedByCurrentApplication)
    {
        // At this stage, the app is the active lock screen background provider.
        // The following code example shows the new URI schema.
        // ms-appdata points to the root of the local app data folder.
        // ms-appx points to the Local app install folder, to reference resources bundled in the XAP package
        var schema = isAppResource ? "ms-appx:///" : "ms-appdata:///Local/";
        var uri = new Uri(schema + filePathOfTheImage, UriKind.Absolute);

        // Set the lock screen background image.
        LockScreen.SetImageUri(uri);

        // Get the URI of the lock screen background image.
        var currentImage = LockScreen.GetImageUri();
        System.Diagnostics.Debug.WriteLine("The new lock screen background image is set to {0}", currentImage.ToString());
        MessageBox.Show("Lock screen changed. Click F12 or go to lock screen.");
    }
    else
    {
        MessageBox.Show("Background cant be updated as you clicked no!!");
    }
}
```

Once you say **Yes**, This app becomes current background image provider of Lock screen and we can see the image set by above method in lock screen by pressing **F12** during using in Windows Phone 8 Emulator.

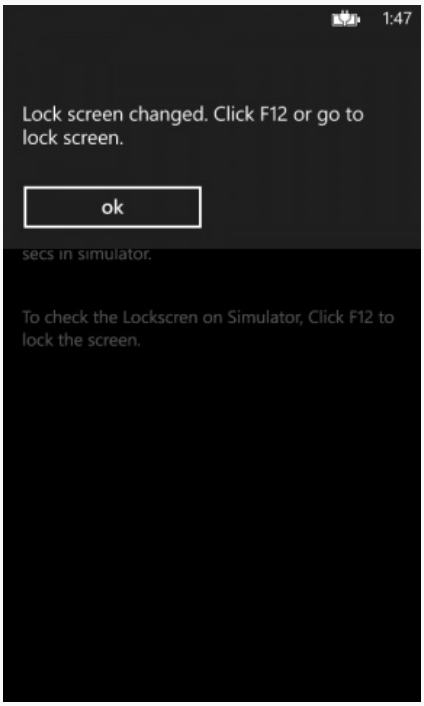


Image set on Lock screen

Home page

To set lock screen image, call `LockScreenChange` method as below. Here we need to pass Image file path that we want to set as Lock screen and whether it is in app resources folder or in root.

```
LockScreenChange("wallpaper/CustomizedPersonalWalleper_0.jpg", true);
```

You can see new lock screen image in below image.

(C) Copyright Nokia 2013. All rights reserved.

New Lock Screen Image

Linking to the lock screen settings screen from within your app

We can’t programmatically turn off our app as a lock screen background image provider from within the app. The user will need to visit the phone's settings screen and make the change themselves. Providing a link to the settings screen makes this useful and easy for user. The following code example shows you how you can route a button click to the phone's lock screen settings screen using URI of setting page in Windows Phone 8 .

```
private async void btnGoToLockSettings_Click(object sender, RoutedEventArgs e)
{
    // Launch URI for the lock screen settings screen.
    var op = await Windows.System.Launcher.LaunchUriAsync(new Uri("ms-settings-lock:"));
}
```

So we have shown above how to set image as lock screen. The following sections show how we can get the image from a number of locations.

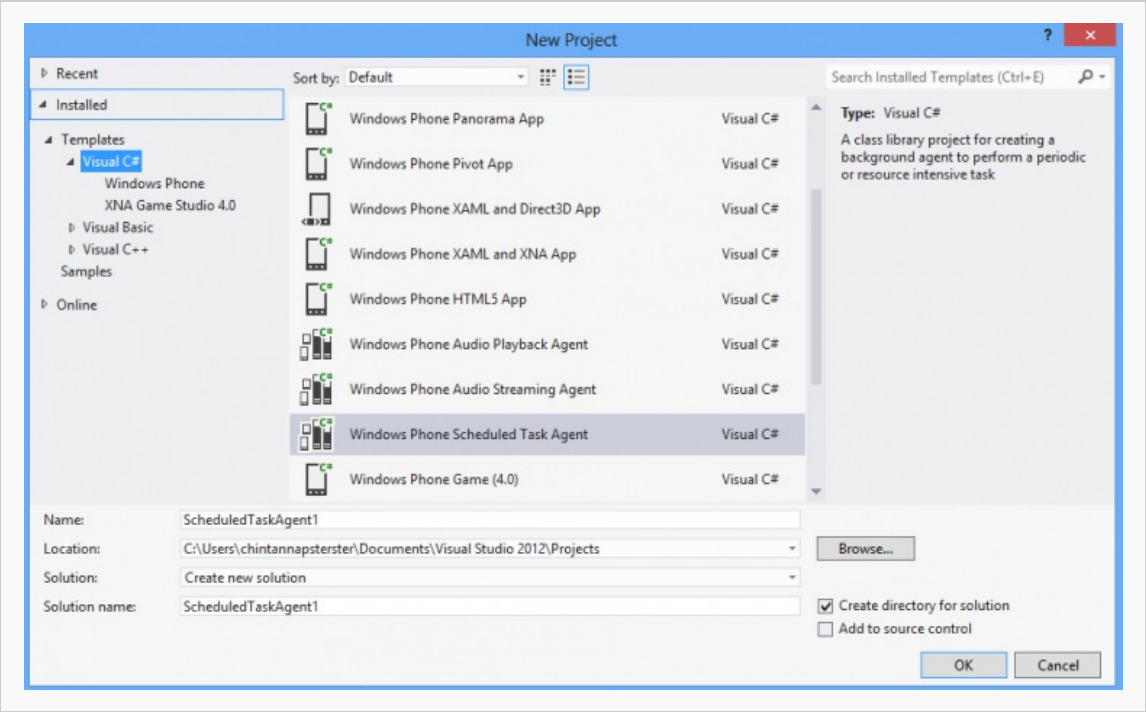
Dynamic Lock screen with images located on app folder

We saw above how we can change the lock screen image. Now we want to make app that change the lock screen periodically using images located on app folder. To change the lock screen after some period, we need to use Windows Phone Scheduled Task Agent, also known as Background Agent.

Change Lock screen image Dynamically using Background Agent

First we need to add Scheduled Task Agent to our solution from "New Project" as seen below. To do that go to **New Project** -> **Windows Phone Scheduled Task Agent**.

(C) Copyright Nokia 2013. All rights reserved.



Windows Phone Scheduled Task Agent

In **MainPage.xaml.cs** class, we need to add PeriodicTask type ScheduleAgentTask as shown below which run after some time (normally after 30 min). But to test this in simulator we cant wait 30 min, so we have made it to run after every 30 sec using LaunchForTest(String, TimeSpan) method which run only on device/simulator when application is in debug mode. You can see we have defined `#define DEBUG_AGENT` before calling LaunchForTest(String, TimeSpan) method.

Here we also defined ExpirationTime of this PeriodicTask by `periodicTask.ExpirationTime` to 14 days or less from the current date. The default value is 14 days from the current date. Applications that use ScheduledTask must renew their schedule at least once every two weeks by calling Remove(String) and then Add(ScheduledAction) to add the task when the main application is running in the foreground. This helps to ensure that unused applications are not draining device resources.

Note: The LaunchForTest(String, TimeSpan) method is provided so that you can run your agent on a more frequent schedule than it will run on an actual device. This method is only for application development. It will not function except for in applications that are deployed using the development tools. You should remove calls to this method from your production application. In this example, the call is placed in an **#if** block to enable you to easily switch between debugging and production functionality. To enable the call, put the following line at the top of the **ScheduledAgent.cs** file. `#define DEBUG_AGENT`

First we need to define PeriodicTask in **MainPage.xaml.cs** as below.

```
PeriodicTask periodicTask;
string periodicTaskName = "PeriodicAgent";
```

Here is the Method which add our PeriodicTask to `ScheduledActionService`.

```
private void StartPeriodicAgent()
{
    // is old task running, remove it
    periodicTask = ScheduledActionService.Find(periodicTaskName) as PeriodicTask;
    if (periodicTask != null)
    {
        try
        {
            ScheduledActionService.Remove(periodicTaskName);
        }
        catch (Exception)
        {
        }
    }
    // create a new task
    periodicTask = new PeriodicTask(periodicTaskName);
    // load description from localized strings
    periodicTask.Description = "This is Lockscreen image provider app.";
    // set expiration days
    periodicTask.ExpirationTime = DateTime.Now.AddDays(14);
    try
    {
        // add this to scheduled action service
        ScheduledActionService.Add(periodicTask);
        // debug, so run in every 30 secs
#if(DEBUG_AGENT)
        ScheduledActionService.LaunchForTest(periodicTaskName, TimeSpan.FromSeconds(30));
        System.Diagnostics.Debug.WriteLine("Periodic task is started: " + periodicTaskName);
#endif

    }
    catch (InvalidOperationException exception)
    {
        if (exception.Message.Contains("BNS Error: The action is disabled"))
        {
            // load error text from localized strings
            MessageBox.Show("Background agents for this application have been disabled by the user.");
        }
        if (exception.Message.Contains("BNS Error: The maximum number of ScheduledActions of this type have already been added."))
        {
```


(C) Copyright Nokia 2013. All rights reserved.

```
            // 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.
    }
}
```

In **ScheduledTaskAgent1** Project, **ScheduledAgent.cs** file, in OnInvoke(ScheduledTask) method which call periodically we need to write code that change the lock screen image periodically. We want to set 10 image one by one periodically in this sample application.

We have 10 images in wallpaper folder. Each images name is start with *CustomizedPersonalWalleper_* and ends with image number(0 to 9). Here we first check which image is currently set as lock screen by LockScreen.GetImageUri method.

```
// Get the URI of the lock screen background image.
var currentImage = LockScreen.GetImageUri();
```

Then we find image number from it by using Substring and String.IndexOf method as below and create the image name that we want to set next and pass it to `LockScreenChange` method. So logic becomes as shown in code below.

```
string imgCount = currentImage.ToString().Substring(currentImage.ToString().IndexOf('_') + 1, currentImage.ToString().Length - (curr

if (imgCount != "9")
    Imagename = "wallpaper/CustomizedPersonalWalleper_" + Convert.ToString(Convert.ToUInt32(imgCount) + 1) + ".jpg";
else
    Imagename = "wallpaper/CustomizedPersonalWalleper_0.jpg";

LockScreenChange(Imagename, true);
```

ScheduledAgent.cs

So Final code of OnInvoke(ScheduledTask) Method is as below.

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

    // Get the URI of the lock screen background image.
    var currentImage = LockScreen.GetImageUri();
    string Imagename = string.Empty;

    string imgCount = currentImage.ToString().Substring(currentImage.ToString().IndexOf('_') + 1, currentImage.ToString().Length - (

    if (imgCount != "9")
        Imagename = "wallpaper/CustomizedPersonalWalleper_" + Convert.ToString(Convert.ToUInt32(imgCount) + 1) + ".jpg";
    else
        Imagename = "wallpaper/CustomizedPersonalWalleper_0.jpg";

    LockScreenChange(Imagename, true);

    // If debugging is enabled, launch the agent again in one minute.
    // debug, so run in every 30 secs
#if(DEBUG_AGENT)
    ScheduledActionService.LaunchForTest(task.Name, TimeSpan.FromSeconds(30));
    System.Diagnostics.Debug.WriteLine("Periodic task is started again: " + task.Name);
#endif

    // Call NotifyComplete to let the system know the agent is done working.
    NotifyComplete();
}
```

Dynamic Lock screen using images located on Internet

Currently you can only set a local or gallery based image as the lockscreen image using `LockScreen.SetImageUri()`. In order to use an image from the Internet we first download the image to `IsolatedStorage` and then use `LockScreen.SetImageUri()` to set it as the Lock Screen image.

Below we will see how to download image from internet and store it in `IsolatedStorage` and then how to set that image as Lock screen.

```
private void DownloadImagefromServer()
{
    WebClient client = new WebClient();
    client.OpenReadCompleted += new OpenReadCompletedEventHandler(client_OpenReadCompleted);
    // give the image url that we need to download and store on IsolatedStorage
    client.OpenReadAsync(new Uri("http://ajtroxell.com/wp-content/uploads/2012/02/wp7_vintage.jpg", UriKind.Absolute));
}
void client_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
{
    BitmapImage bitmap = new BitmapImage();
    bitmap.SetSource(e.Result);
    //img.Source = bitmap;

    // Create a filename for JPEG file in isolated storage.
    String tempJPEG = "DownloadedWalleper.jpg";

    // Create virtual store and file stream. Check for duplicate tempJPEG files.
    using (IsolatedStorageFile myIsolatedStorage = IsolatedStorageFile.GetUserStoreForApplication())
    {
        if (myIsolatedStorage.FileExists(tempJPEG))
        {
            myIsolatedStorage.DeleteFile(tempJPEG);
        }
```


(C) Copyright Nokia 2013. All rights reserved.

```
        }

        IsolatedStorageFileStream fileStream = myIsolatedStorage.CreateFile(tempJPEG);

        StreamResourceInfo sri = null;
        Uri uri = new Uri(tempJPEG, UriKind.Relative);
        sri = Application.GetResourceStream(uri);

        //BitmapImage bitmap = new BitmapImage();
        //bitmap.SetSource(sri.Stream);
        WriteableBitmap wb = new WriteableBitmap(bitmap);

        // Encode WriteableBitmap object to a JPEG stream.
        Extensions.SaveJpeg(wb, fileStream, wb.PixelWidth, wb.PixelHeight, 0, 85);

        //wb.SaveJpeg(fileStream, wb.PixelWidth, wb.PixelHeight, 0, 85);
        fileStream.Close();
    }

    // call function to set downloaded image as lock screen
    LockScreenChange("DownloadedWalleper.jpg", false);
}
```

You can see below in screenshot that a image located over Internet is now set as Lock screen.

Dynamic Lock screen image like Bing App

We need to upload different image over Internet periodically in other platform so that our app in Windows Phone 8 can download it and set it as Lock screen image. To Upload image periodically over Internet in other platform, We can create some Windows Service in .net that will run on Windows and upload images to a specific URL over Internet by picking image from provided folder path. OR we can have bunch of image URL to set as lock screen periodically.

Concept becomes Reality

Below concept can now become reality. To make it real, We need to have one image URL path which will have different images periodically with some details written on image and our lock screen app will download image from that Image URL path and set as Lock screen. We just need to upload different images to specific image URL path. To upload image, we can use Windows Service of other platform or we can upload it manually.

We can not write text over image at runtime in Windows Phone 7 and 8 but we can do that in other platform using imaging API for example in .net and upload composed image to specific Image URL and our lock screen app in windows phone 8 will set it as lock screen.

So this way, below concept becomes real app.

(C) Copyright Nokia 2013. All rights reserved.

Dynamic Lock screen with images coming from RSS

Here we will see how to set image from RSS as lock screen in windows Phone 8. To do this, RSS must have image URL in it. To set lock screen image from RSS, first we need to download image from RSS using image URL shown in RSS. So first we need to parse the RSS and checking for each image with type .jpeg or .png and then download image from it and set it to lock screen.

We also need to consider the structure of the RSS while downloading image from it as we need to parse RSS according to RSS structure and image URL location in RSS.

Sample RSS

Here is the sample RSS which contains Image URL in it. We get this RSS from this RSS link.

```
<rss xmlns:media="http://search.yahoo.com/mrss/" xmlns:dc="http://purl.org/dc/elements/1.1/" version="2.0">
  <channel>
    <title>Flickr Photos tagged with nature</title>
    <link>
      www.flickr.com
    </link>
    <description>Flickr Photos tagged with nature</description>
    <language>en-us</language>
    <pubDate>Mon, 10 Dec 2012 01:40:13 -0500</pubDate>
    <lastBuildDate>Mon, 10 Dec 2012 01:40:13 -0500</lastBuildDate>
    <generator>http://www.degraeve.com/flickr-rss/</generator>
    <webMaster>webmaster@degraeve.com</webMaster>
    <item>
      <title>mandarin duck in full color</title>
      <link>http://www.flickr.com/photos/54419396@N00/45599281</link>
      <description>
        <img src="http://farm1.static.flickr.com/27/45599281_0e33a17e35.jpg" alt="" />
      </description>
      <pubDate>Thu, 22 Sep 2005 11:42:56 -0400</pubDate>
      <guid>
        http://farm1.static.flickr.com/27/45599281_0e33a17e35.jpg
      </guid>
      <media:content url="http://farm1.static.flickr.com/27/45599281_0e33a17e35.jpg" type="image/jpeg"/>
      <media:title>mandarin duck in full color</media:title>
      <media:text type="html">
        <img src="http://farm1.static.flickr.com/27/45599281_0e33a17e35.jpg" />
      </media:text>
      <media:thumbnail url="http://farm1.static.flickr.com/27/45599281_0e33a17e35.jpg"/>
    </item>
    <item>
      <title>Japanese Maple</title>
      <link>
        http://www.flickr.com/photos/52317893@N00/263966036
      </link>
      <description>
        <img src="http://farm1.static.flickr.com/103/263966036_ee04032cc8.jpg" alt="" />
      </description>
      <pubDate>Sun, 08 Oct 2006 12:17:38 -0400</pubDate>
      <guid>
        http://farm1.static.flickr.com/103/263966036_ee04032cc8.jpg
      </guid>
      <media:content url="http://farm1.static.flickr.com/103/263966036_ee04032cc8.jpg" type="image/jpeg"/>
      <media:title>Japanese Maple</media:title>
      <media:text type="html">
        <img src="http://farm1.static.flickr.com/103/263966036_ee04032cc8.jpg" />
      </media:text>
      <media:thumbnail url="http://farm1.static.flickr.com/103/263966036_ee04032cc8.jpg"/>
    </item>
  </channel>
</rss>
```

Parse RSS

(C) Copyright Nokia 2013. All rights reserved.

We need to first download the RSS shown above from the rss page shown below. http://www.degraeve.com/flickr-rss/rss.php?tags=nature&tagmode=all&sort=interestingness-desc&num=24

For that we need to use below code. Pass above URL to `DownloadRSS(string rssURL)` method as input.

```
// Global variable
public int imageCount = 0;
public string[] imgarray;

// pass above url in this method as input argument
public void DownloadRSS(string rssURL)
{
    WebClient myRSS = new WebClient();
    myRSS.DownloadStringCompleted += new DownloadStringCompletedEventHandler(myRSS_DownloadStringCompleted);
    myRSS.DownloadStringAsync(new Uri(rssURL));
}
void myRSS_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs e)
{
    //Check if the Network is available
    if (Microsoft.Phone.Net.NetworkInformation.DeviceNetworkInformation.IsNetworkAvailable)
    {
        // filter all images from rss located on media:content
        XNamespace media = XNamespace.Get("http://search.yahoo.com/mrss/");
        imgarray = XElement.Parse(e.Result).Descendants(media + "content")
            .Where(m => m.Attribute("type").Value == "image/jpeg")
            .Select(m => m.Attribute("url").Value)
            .ToArray();

        // check that images are there in rss
        if (imgarray.Length > 0)
        {
            imageCount = 0;
            // download images
            DownloadImagefromServer(Convert.ToString(imgarray[0]));
        }
        else
        {
            MessageBox.Show("No image found in applied RSS link");
        }

    }
    else
    {
        MessageBox.Show("No network is available..");
    }
}
```

As you can see in above code, we first parse the RSS, get all the images URLs from that RSS and store Image URLs in global string array `imgarray`. So that we can use that string array `imgarray` in other methods and can get the image URL we have retrieved from given RSS.

Download RSS images

Now after getting all image URLs from RSS, we need to download those images one by one and store it in `IsolatedStorage` as we can not use image url during setting image using method `DownloadImagefromServer(string imageUrl)`. So that's why we need to download images first and store it in `IsolatedStorage`. We have done that in below code in method `DownloadImagefromServer(string imageUrl)` which called in above Event handler `myRSS_DownloadStringCompleted`.

```
// to download image from server
private void DownloadImagefromServer(string imageUrl)
{
    WebClient client = new WebClient();
    client.OpenReadCompleted += new OpenReadCompletedEventHandler(client_OpenReadCompleted);
    client.OpenReadAsync(new Uri(imageUrl, UriKind.Absolute));
}
void client_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
{
    BitmapImage bitmap = new BitmapImage();
    bitmap.SetSource(e.Result);

    // Create a filename for JPEG file in isolated storage.
    String tempJPEG = "DownloadedWalleper_" + Convert.ToString(imageCount) + ".jpg";

    // Create virtual store and file stream. Check for duplicate tempJPEG files.
    using (IsolatedStorageFile myIsolatedStorage = IsolatedStorageFile.GetUserStoreForApplication())
    {
        if (myIsolatedStorage.FileExists(tempJPEG))
        {
            myIsolatedStorage.DeleteFile(tempJPEG);
        }

        IsolatedStorageFileStream fileStream = myIsolatedStorage.CreateFile(tempJPEG);

        StreamResourceInfo sri = null;
        Uri uri = new Uri(tempJPEG, UriKind.Relative);
        sri = Application.GetResourceStream(uri);

        WriteableBitmap wb = new WriteableBitmap(bitmap);

        // Encode WriteableBitmap object to a JPEG stream.
        System.Windows.Media.Imaging.Extensions.SaveJpeg(wb, fileStream, wb.PixelWidth, wb.PixelHeight, 0, 85);

        //wb.SaveJpeg(fileStream, wb.PixelWidth, wb.PixelHeight, 0, 85);
        fileStream.Close();

        System.Diagnostics.Debug.WriteLine("image downloaded: " + tempJPEG);

        if (imageCount == imgarray.Length - 1)
        {
            // set first image
            LockScreenChange("DownloadedWalleper_0.jpg", false);
            // start service
            StartPeriodicAgent();
        }

        if (imageCount < imgarray.Length - 1)
        {
            imageCount++;
```


(C) Copyright Nokia 2013. All rights reserved.

```
            // download remaining images
             DownloadImagefromServer(Convert.ToString(imgarray[imageCount]));
        }
    }
}
```

As you can see above in `client_OpenReadCompleted` event handler, we are downloading other remaining images from `imgarray` by calling `DownloadImagefromServer(string imageUrl)` method.

Set Lock screen with RSS Images

After downloading all Images from RSS to `IsolatedStorage` with specific name, we need to set it all to lock screen using `LockScreen.SetImageUri(uri)` method periodically by Background Agent Scheduler.

```
private async void LockScreenChange(string filePathOfTheImage, bool isAppResource)
{
    if (!LockScreenManager.IsProvidedByCurrentApplication)
    {
        // If you're not the provider, this call will prompt the user for permission.
        // Calling RequestAccessAsync from a background agent is not allowed.
        await LockScreenManager.RequestAccessAsync();
    }

    // Only do further work if the access is granted.
    if (LockScreenManager.IsProvidedByCurrentApplication)
    {
        // At this stage, the app is the active lock screen background provider.
        // The following code example shows the new URI schema.
        // ms-appdata points to the root of the local app data folder.
        // ms-appx points to the Local app install folder, to reference resources bundled in the XAP package
        var schema = isAppResource ? "ms-appx:///" : "ms-appdata:///Local/";
        var uri = new Uri(schema + filePathOfTheImage, UriKind.Absolute);

        // Set the lock screen background image.
        LockScreen.SetImageUri(uri);

        // Get the URI of the lock screen background image.
        var currentImage = LockScreen.GetImageUri();
        System.Diagnostics.Debug.WriteLine("The new lock screen background image is set to {0}", currentImage.ToString());
        MessageBox.Show("Lock screen changed. Click F12 or go to lock screen.");
    }
    else
    {
        MessageBox.Show("Background cant be updated as you clicked no!!");
    }
}
```

Then we need to add `BackgroundAgent` as shown above Change Lock screen image Dynamically using Background Agent. So please refer steps written in that section. So that I am not explaining here again.

Change lock screen periodically with RSS Images

Below is the code to change the lock screen with RSS images periodically.

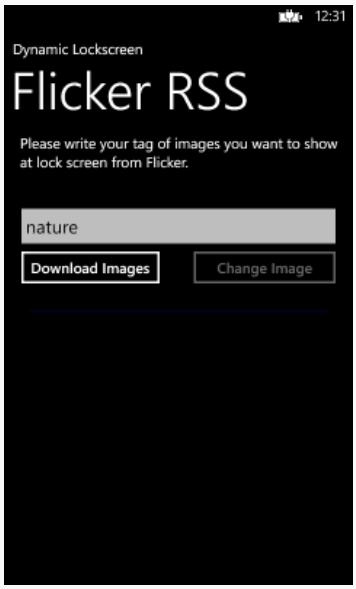
In **ScheduledTaskAgent1** project, `ScheduledAgent.cs` class, in `OnInvoke(ScheduledTask task)` Method, we need to write code as shown below. Here we are just downloading 25 images from given RSS. as we know that that RSS will have only 25 images in it.

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

    // Get the URI of the lock screen background image.
    var currentImage = LockScreen.GetImageUri();
    string Imagename = string.Empty;

    string imgCount = currentImage.ToString().Substring(currentImage.ToString().IndexOf('_') + 1, currentImage.ToString().Length - (

    if (imgCount != "24")
        Imagename = "DownloadedWalleper_" + Convert.ToString(Convert.ToUInt32(imgCount) + 1) + ".jpg";
    else
        Imagename = "DownloadedWalleper_0.jpg";

    LockScreenChange(Imagename, false);

    // If debugging is enabled, launch the agent again in one minute.
    // debug, so run in every 30 secs
#if(DEBUG_AGENT)
    ScheduledActionService.LaunchForTest(task.Name, TimeSpan.FromSeconds(30));
    System.Diagnostics.Debug.WriteLine("Periodic task is started again: " + task.Name);
#endif

    // Call NotifyComplete to let the system know the agent is done working.
    NotifyComplete();
}
```

DEMO APP: Flicker RSS Dynamic Lock Screen Application

I have also written a article for Flicker RSS Dynamic Lock Screen Application which set lock screen periodically with images from Flicker RSS and also images which user asked for. Please find that article and source code of app on below link. Flicker RSS Dynamic Lock Screen

(C) Copyright Nokia 2013. All rights reserved.



Flicker RSS Lock screen UI

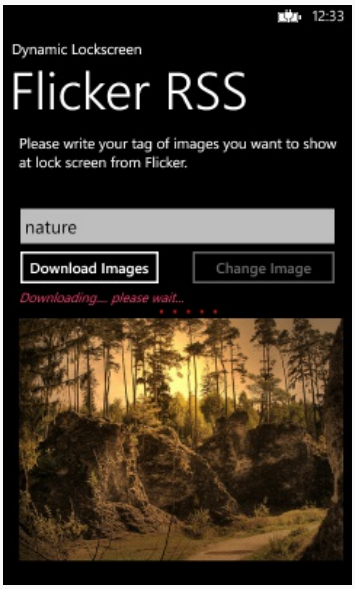
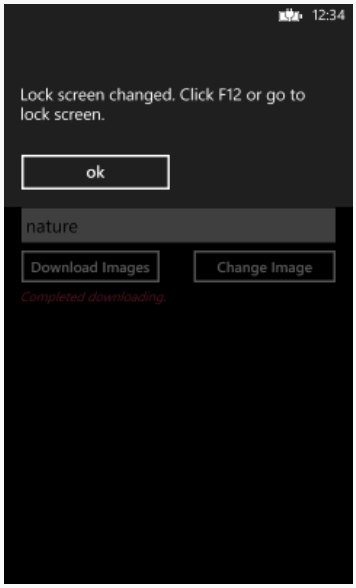
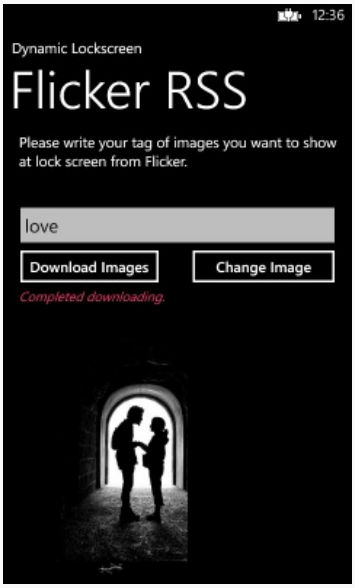


Image downloading

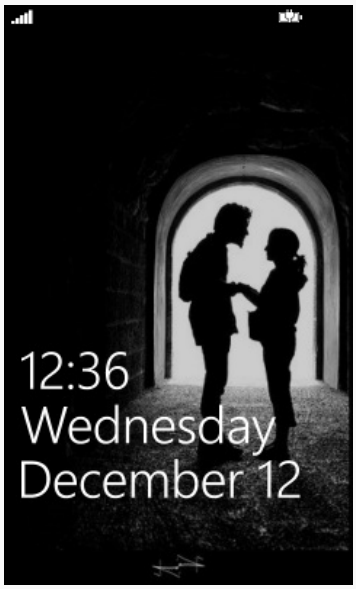


Lock screen image set



Download Stared with 'love' tag

(C) Copyright Nokia 2013. All rights reserved.



Lockscreen Set with new image

Lockscreen Set with new image

Please find the source code of a demo app on below Source Code section.

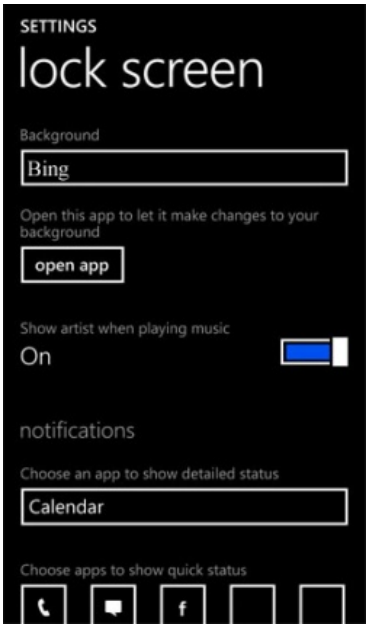
WP8 Lock Screen App Notifications

One complaint many users had about WP7 was a very restricted customization options. But now with WP8, many customization channels are open and one of those is to show application notifications on lock screen.

WP8 allows the lock screen notifications to be shown in two ways:

1. Detailed status of one application (default Calendar)
2. Five quick status notifications as per the choice of user

- The detailed app status shows detailed textual information.
  Example - if you have chosen Email app, then the latest email subject will be displayed.
- Quick status notifications are icons + number
  Example - Facebook icon and its notification count.



Enabling your App for lock screen notifications

To add Lock screen notification for WP8 app, edit the app's manifest file - wmAppManifest.xml:

- Declare the notification capability in the Extensions

```
<Extension ExtensionName="LockScreen_Notification_IconCount"
ConsumerID="{000AEC10-CZ01-1C12-7771-1ADC289137C}" TaskID="_default">
<Extension ExtensionName="LockScreen_Notification_TextField"
ConsumerID="{000AEC10-CZ01-1C12-7771-1ADC289137C}" TaskID="_default">
```


(C) Copyright Nokia 2013. All rights reserved.

- Add the icon image entry:

```
<DeviceLockImageURI IsRelative="true"  IsResource="false">icon.png
</DeviceLockImageURI>
```

Note: The lock screen icon should be only in white pixels and some level of transparency is allowed.

We are done with our aim and now whenever the app updates its live tile, lock screen notification will be also updated automatically.

Summary

So this way, we can create application for Windows Phone 8 that change the lock screen background periodically. We can also set image which located on server as lock screen. This image located over internet server might have some information written on it as text or image.

So we have seen how to create set a image as lock screen, how to make lock screen image dynamic with images located on app folder or over internet and finally how to make a WP8 app enabled for dynamic lock screen notifications.

Source code

You can find the running source code of all demo applications presented in this article on below links.

- **Dynamic Lock Screen - Image located on local folder of app**
  Media:DynamicLockScreen.zip
- **Dynamic Lock Screen - Image located over internet**
  Media:DynamicLockScreen - Image Located over server.zip
- **Dynamic Lock Screen - Images from RSS URL Flicker App** (Find article Flicker_Dynamic_Lock_screen_application_for_Windows_Phone_8
  Media:RssLockscreen.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.