Flicker Dynamic Lock screen application for Windows Phone 8

This article explains how to create application that change the lock screen image using Flicker images like inbuilt bing app. In this article we will use RSS functionality in getting image and setting it as lock screen.

Introduction

This article shows how to create locks screen application that can change the lock screen with images from RSS URL. Here, we will see how to get images of particular tag from Flicker site using RSS and set them all one by one to Windows Phone 8 lock screen periodically. But first we will see how to set lock screen from a RSS image.

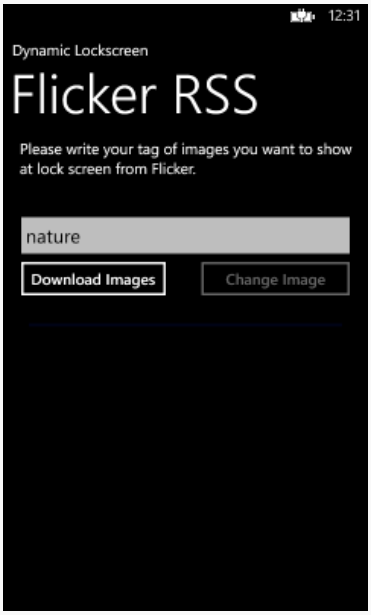
Please read this article first on How to: Dynamic Lock Screen for Windows Phone 8

Video Demo

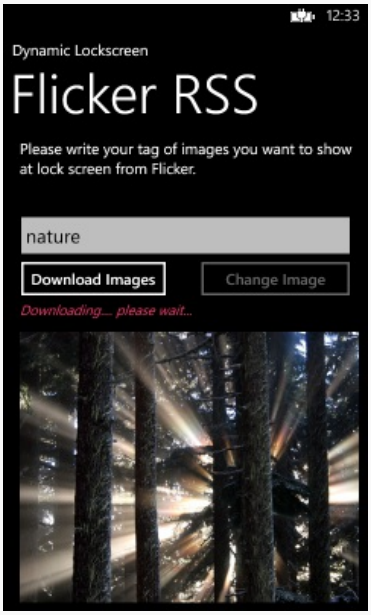
In below video you can see the Flicker lock screen app in action and changing lock screen image every 30 sec.

The media player is loading...

Screen Shots



Flicker RSS Lock screen UI



Clicked on Download Image button

(C) Copyright Nokia 2014. All rights reserved.

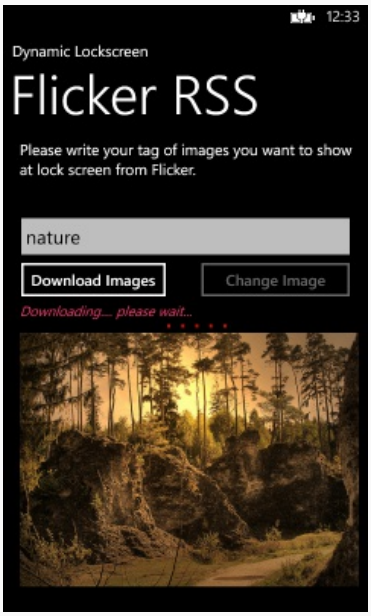
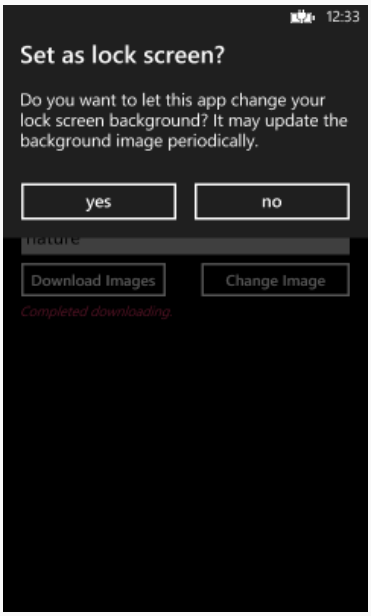
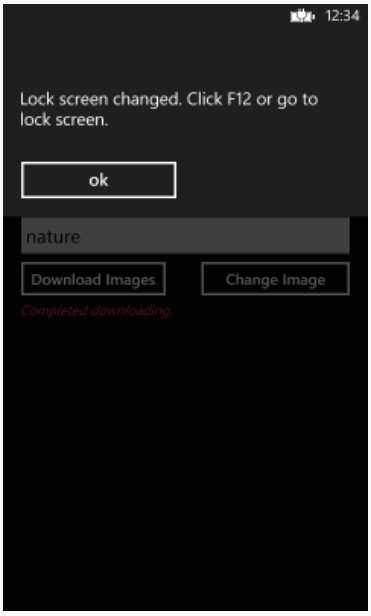


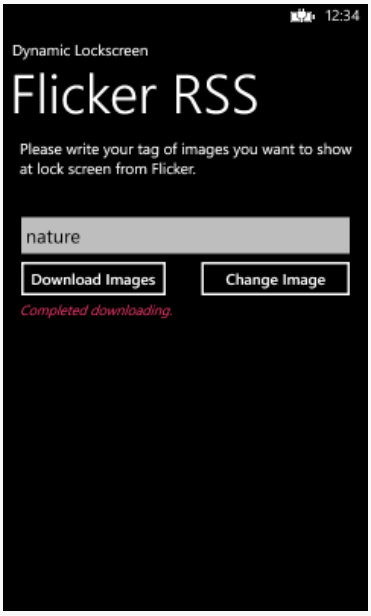
Image downloading



Asking permission to set as Lock screen image provider

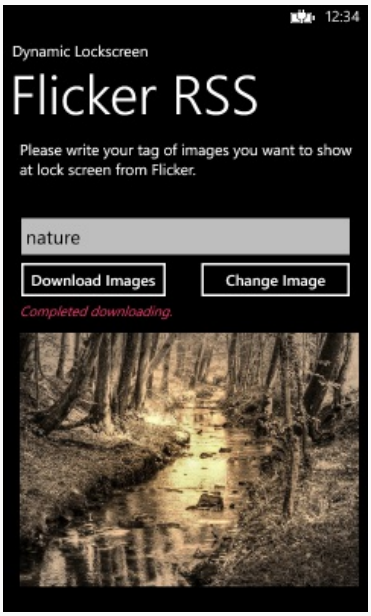


Lock screen image set

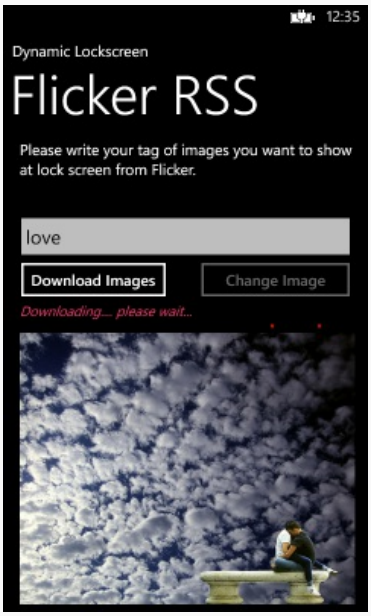


Download Complete, Background Service Started.

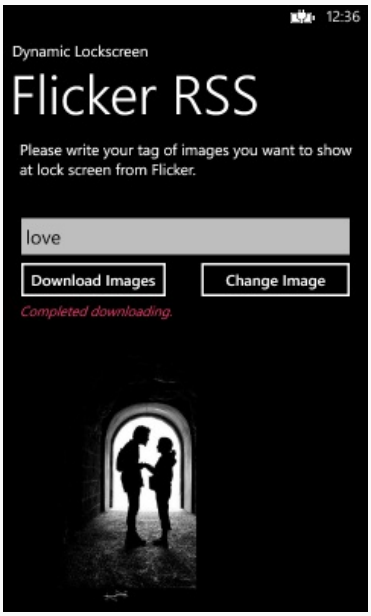
(C) Copyright Nokia 2014. All rights reserved.



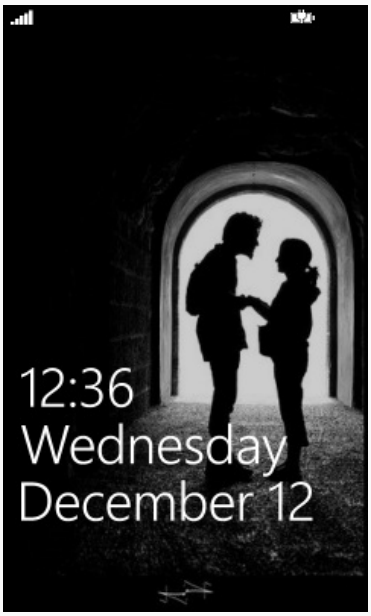
Change Lock screen manually



Change Tag to "love' to download images



Download Stared again for 'love'



Lockscreen Set with new image

(C) Copyright Nokia 2014. All rights reserved.

Lockscreen Set with new image

Lockscreen Set with new image

How to set Lock Screen from a RSS Image

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. We can pass the number of items we want to fetch in RSS in query string **num**.

Sample RSS

Here is the sample RSS which contains Image URL in it. We get this RSS from this RSS link. Here we are passing only 2 in num query string.

```
<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"
```


(C) Copyright Nokia 2014. All rights reserved.

```
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

We need to first download the RSS shown above from the rss page shown below. For that we need to use below code. Pass above URL to `DownloadRSS(string rssURL)` method as input and it will first downloadRSS and parse it and get the all images.

```
// 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")
```


(C) Copyright Nokia 2014. All rights reserved.

```
            .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);
```


(C) Copyright Nokia 2014. All rights reserved.

```
        }

        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++;
           // 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.
```


(C) Copyright Nokia 2014. All rights reserved.

```
        // 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 - (currentImage.ToString().IndexOf('_') +
1)).Replace(".jpg", "");

    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);
```


(C) Copyright Nokia 2014. All rights reserved.

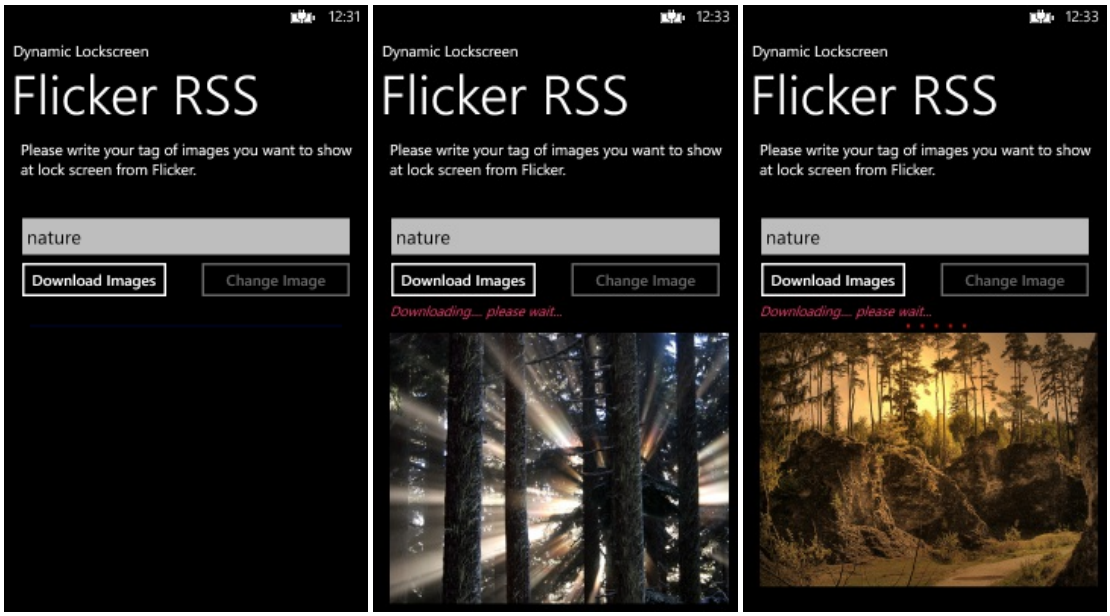
```
#endif

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

Now using above code, we are going to create a real-life application to set lock screen and we will also go advance and will add functionality using which user can get the images he/she like to see in his/her phone's Lock screen. Interesting ? Then Let's get started.

Flicker Dynamic Lock screen App

Using above technique, we want to set lock screen with images downloaded from RSS of Flicker. Used needs to write the tag name of his/her favourite images and our app will download it and set it to lock screen periodically. User can also change the the lock screen manually. During downloading images, we also want to show them one by one to image on mainpage. So that user can see what images will be to phone's lock screen. You can see what I have explained in below screenshots.



For that first we need to design our application on **MainPage.xaml** file. Here is the code of **Mainpage.xaml** file.

MainPage.xaml

```
<Grid x:Name="LayoutRoot" Background="Transparent">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>

    <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="0,14,0,4">
        <TextBlock Text="Dynamic Lockscreen" Style="{StaticResource
PhoneTextNormalStyle}" Margin="12,0"/>
        <TextBlock Text="Flicker RSS" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>
    </StackPanel>

    <!--ContentPanel - place additional content here-->
    <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
        <Image Name="lockscreen" HorizontalAlignment="Left" Height="587"
Margin="0,10,0,0" VerticalAlignment="Top" Width="446" />
        <TextBlock HorizontalAlignment="Left" Margin="10,10,0,0" TextWrapping="Wrap"
Text="Please write your tag of images you want to show at lock screen from Flicker."
VerticalAlignment="Top"/>
```


(C) Copyright Nokia 2014. All rights reserved.

```
        <TextBox HorizontalAlignment="Left" Height="72" Margin="0,103,0,0"
TextWrapping="Wrap" Text="nature" VerticalAlignment="Top" Width="456" Name="txtTag"/>
        <Button Content="Download Images" HorizontalAlignment="Left" Margin="0,162,0,0"
VerticalAlignment="Top" RenderTransformOrigin="-1.275,-0.389" Click="Button_Click_1"
FontSize="20"/>
        <Image Name="DwldImg" HorizontalAlignment="Left" Height="358"
Margin="10,266,0,0" VerticalAlignment="Top" Width="446"/>
        <ProgressBar Name="progress" HorizontalAlignment="Left" Height="9"
Margin="10,252,0,0" VerticalAlignment="Top" Width="436" RenderTransformOrigin="-1.21,-
0.3" Background="#190425FB"/>
        <TextBlock Text="" Name="downloadPercentage" HorizontalAlignment="Center"
Margin="10,225,9,0" TextWrapping="Wrap" VerticalAlignment="Top" FontSize="18"
Width="437" Foreground="#FFF73C6F" FontStyle="Italic"/>
    </Grid>
    <Button Name="btnChangeImg" Content="Change Image" HorizontalAlignment="Left"
Margin="249,162,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="20" Width="219"
Click="Button_Click_2"/>

</Grid>
```

Now our design is ready, Let implement the functionality by coding in code-behind .cs files.

MainPage.xaml.cs

Now we need to add button event shown in design by clicking on it in designer file **MainPage.xaml**. By this, VS will create Button Click Event on its .cs file.

1. In **Download Images** button click event, we want to download the images of tag user has written in textbox above.
2. So for that we will call `DownloadRSS(url)` method which first download RSS and get all images name in String Array and then download images one by one.
3. Here we also need to disable other button **Change Image** as during downloading images, we do not want user to change his/her lock screen.

Here is the code for Click event of button *Download Images'* with above steps.

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    if (!string.IsNullOrEmpty(txtTag.Text))
    {
        btnChangeImg.IsEnabled = false;
        progress.IsIndeterminate = true;
        progress.Visibility = Visibility.Visible;
        downloadPercentage.Text = "Downloading.... please wait...";
        string url = "http://www.degraeve.com/flickr-rss/rss.php?tags=" + txtTag.Text +
"&tagmode=all&sort=interestingness-desc&num=25";
        DownloadRSS(url);

    }
    else
    {
        MessageBox.Show("Please write your tag in textbox of images you want to show on
lock screen");
    }
}
```

As you can see in above code, we are generating RSS url with tag name user has written in `txtTag` textbox. Here we passing 25 in `num` query as we want to fetch 25 images from RSS and set to lock screen. We can also make this number of images dynamic by giving UI to select to user.

(C) Copyright Nokia 2014. All rights reserved.

Now as shown above, `DownloadRSS(url)` will call with generated RSS url as input. `DownloadRSS(url)` method is as below.

```
public void DownloadRSS(string rssURL)
{
    WebClient myRSS = new WebClient();
    myRSS.DownloadStringCompleted += new
DownloadStringCompletedEventHandler(myRSS_DownloadStringCompleted);
    myRSS.DownloadStringAsync(new Uri(rssURL));
}
```

Which will call `myRSS.DownloadStringAsync(new Uri(rssURL))` Event Handler. In this event Handler, we first parse the RSS and get the images from `<media:content>` tags which originally have the image url in it with **type=image/jpeg**. Show using **LINQ** query we are parsing RSS and getting Image URLs and storing it in a string array named `imgarray`. This string array `imgarray` is defined as global. So that we can get array values in other method later.

Here is the code to parse RSS and get image URLs from `<media:content>` tags and store it in `imgarray` Global String array.

```
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();
```

Here is the full code for Event Handler `myRSS.DownloadStringAsync(new Uri(rssURL))` with above code in it which parse RSS and get image URLs.

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

void myRSS_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs e)
{
    //Check if the Network is available
    if
(Microsoft.Phone.Net.NetworkInformation.DeviceNetworkInformation.IsNetworkAvailable)
    {
        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();

        if (imgarray.Length > 0)
        {
            imageCount = 0;
            DownloadImagefromServer(Convert.ToString(imgarray[0]));
        }
        else
        {
            MessageBox.Show("No image found for tag " + txtTag.Text + ". Please try
other tag name for image.");
        }

    }
    else
    {
```


(C) Copyright Nokia 2014. All rights reserved.

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

In `myRSS.DownloadStringAsync(new Uri(rssURL))` method, we are downloading first image by getting image URL from string array `imgarray`. To download image to `IsolatedStorage`, we are calling `DownloadImagefromServer(string imageUrl)` which we have created as below.

```
private void DownloadImagefromServer(string imageUrl)
{
    WebClient client = new WebClient();
    client.OpenReadCompleted += new
OpenReadCompletedEventHandler(client_OpenReadCompleted);
    client.OpenReadAsync(new Uri(imageUrl, UriKind.Absolute));
}
```

`myRSS.DownloadStringAsync(new Uri(rssURL))` method add event handler `client_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)` every time it calls, which download image and store it to `IsolatedStorage` with name starting **DownloadedWalleper_NUM.jpg**. Here *NUM* is the image number.

- During downloading every image, we also setting downloaded image to Image control `DwldImg` on UI.
- Once we download first image, we call `DownloadImagefromServer(string ImageUrl` to download second image by passing Url of second image from array. and So on we are calling method with remaining images up to array length (i.e. 25 here).
- Once we download last image (i.e 25 here), we set first image to lock screen and start `BackgroundAgent` by calling method `StartPeriodicAgent()`.

```
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);
```


(C) Copyright Nokia 2014. All rights reserved.

```
        fileStream.Close();

        System.Diagnostics.Debug.WriteLine("image downloaded: " + tempJPEG);
        DwldImg.Source = null;

        Thread.Sleep(400);

        if (imageCount ==  imgarray.Length - 1)
        {
            btnChangeImg.IsEnabled = true;

            progress.IsIndeterminate = false;
            progress.Visibility = Visibility.Collapsed;
            downloadPercentage.Text = "Completed downloading.";

            // set first image
            LockScreenChange("DownloadedWalleper_0.jpg", false);
            // start service
            StartPeriodicAgent();
        }

        if (imageCount < imgarray.Length - 1)
        {
            imageCount++;
            DownloadImagefromServer(Convert.ToString(imgarray[imageCount]));
            // set current downloaded image to Image control on UI
            DwldImg.Source = bitmap;

        }

    }
}
```

Here is the normal method which set lock screen with applied image url located on IsolatedStorage of app.

```
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.
```


(C) Copyright Nokia 2014. All rights reserved.

```
        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!!");
    }
}
```

Background Agent

So now lock screen is set by first downloaded image from RSS. Now as we already started `BackgroundAgent` by calling method `StartPeriodicAgent()`, `BackgroundAgent` will set other images periodically to lock screen event if our application is closed. Find the `StartPeriodicAgent()` code below.

Get more information about how to add `BackgroundAgent` to VS project on my other article here.
Dynamic_Lock_Screen_for_Windows_Phone_8#Change_Lock_screen_image_Dynamically_using_Background_Agent

```
PeriodicTask periodicTask;
string periodicTaskName = "PeriodicAgent";
public bool agentsAreEnabled = true;

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 thas 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);
```


(C) Copyright Nokia 2014. All rights reserved.

```
#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."))
        {
            // 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.
    }
}
```

Now on **ScheduledTaskAgent1.cs** class, in `OnInvoke(ScheduledTask task)` method, we are changing the lock screen periodically with images we have downloaded from RSS of flicker.

- Here we first get the current image name set as lock screen using below code.

```
var currentImage = LockScreen.GetImageUri();
```

- With that name, we get the next image name by removing **DownloadedWalleper_** from image name.
- After getting next image name, we passing to method `LockScreenChange` to set it as Lock screen. Here we are changing lockscreen after every 30 sec. So we don't need to wait during developing and testing. We can frequency of lock screen changing below method.

```
ScheduledActionService.LaunchForTest(task.Name, TimeSpan.FromSeconds(30));
```

Find below full code of method `OnInvoke(ScheduledTask task)` of `ScheduledTaskAgent1.cs` class.

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

    // Only do further work if the access is granted.
    if (LockScreenManager.IsProvidedByCurrentApplication)
    {

        // 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 - (currentImage.ToString().IndexOf('_') +
1)).Replace(".jpg", "");
```


(C) Copyright Nokia 2014. All rights reserved.

```
        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();
    }
    else
    {
        // remove this service
        ScheduledActionService.Remove(task.Name);
    }
}
```

So We are done with this application almost. But we forget one thing that we have designed in **MainPage.xaml** page.

We wanted to change the lock screen image manually by clicking on button **Change Image**. For that we need to call `ChangeLockscreen()` method that we created in that button's click event as below.

That `ChangeLockscreen()` method,

- first get the current image name set as lock screen using below code.

```
var currentImage = LockScreen.GetImageUri();
```

- With that name, we get the next image name by removing **DownloadedWalleper_** from image name.
- After getting next image name, we set Image control `DwldImg`, `Source` property with it after getting it in `BitmapImage` from `IsolatedStorageFile` as shown below.

```
BitmapImage Bit_Img = new BitmapImage();
using (IsolatedStorageFile ISF = IsolatedStorageFile.GetUserStoreForApplication())
{
    using (IsolatedStorageFileStream FS = ISF.OpenFile(Imagename, FileMode.Open,
FileAccess.Read))
    {
        Bit_Img.SetSource(FS);
    }
}
this.DwldImg.Source = Bit_Img;
```

You can find the full code for changing image on button's click below which I have explained above step by step.

```
private void Button_Click_2(object sender, RoutedEventArgs e)
{
```


(C) Copyright Nokia 2014. All rights reserved.

```
    ChangeLockscreen();
}
private void ChangeLockscreen()
{
    try
    {
        var currentImage = LockScreen.GetImageUri();
        string Imagename = string.Empty;

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

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

        LockScreenChange(Imagename, false);
        BitmapImage Bit_Img = new BitmapImage();
        using (IsolatedStorageFile ISF =
IsolatedStorageFile.GetUserStoreForApplication())
        {
            using (IsolatedStorageFileStream FS = ISF.OpenFile(Imagename, FileMode.Open,
FileAccess.Read))
            {
                Bit_Img.SetSource(FS);
            }
        }
        this.DwldImg.Source = Bit_Img;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Some error occured in changing lock screen. please try again
after some time.");
    }
}
```

So that's it. Our application for Lock screen is completed.

Source Code

- You can find the running source code of shown application on below link.

  File:RssLockscreen.zip

- Also Find installation **.xap** file of this application to install it in Windows Phone 8 device.

  Media:RssLockscreen_1.0.0.zip

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.