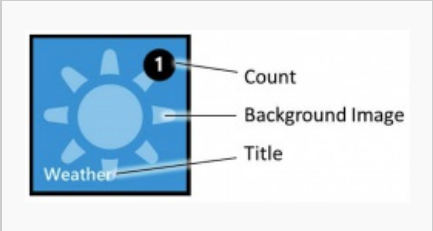
Working with Live Tiles on Windows Phone

This code example shows how to work with Live Tiles in Windows Phone, covering how to create application primary and secondary tiles and how to update tiles data from Scheduled Task Agent and with Push Notification from server side with PHP.

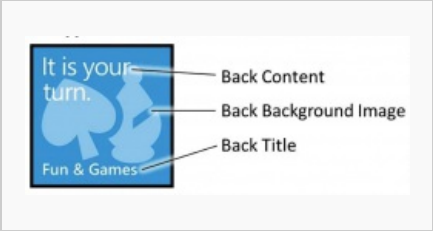
Note: This article was written for and has been tested on Windows Phone 7. It is also accurate for Windows Phone 8 but at time of writing has not been tested, and omits the discussion of new tile templates added in Windows Phone 8.

Introduction

Live Tiles represent an application on the Windows Phone Start screen, and when selected provide a link to a URI within the application. Tiles are two-sided and randomly switch between sides. Both sides have a title and background image; the front side also displays a *counter* while the back can have additional text designated as its *back content*. Tile content may be updated in code, and updates can be initiated from push or scheduled notifications.



Front tile



Back of tile

Apps have a *Application Tile* which may be manually created and removed by the user (it is pinned to the Start page when the users presses and holds the application icon in the application list). This tile cannot be programmatically added or removed, but it's properties may be set and updated from code.

Apps can also have as many *secondary tiles* as needed (whether or not an *Application Tile* has been pinned): these are usually links to specific "views" on the app - for example a particular song in a music app, or the weather for a particular city in a city app.

This article demonstrates how to create, delete and update tiles, and covers advanced techniques including updating their content based on push and scheduled notifications. It extends the information provided in the #Related Articles section.

Video Demo

The video below shows the operation of example provided with this article. It is best viewed "full screen": The media player is loading...

Application Tile

Application tiles should confirm with the UI guidance in the start section of Essential Graphics, Visual Indicators, and Notifications for Windows Phone, and with Application Certification Requirements for Windows Phone. These are important - defining important parameters like the expected size of tile images (173px by 173px). The following sections explain how to set the initial/default value of the application tiles. Note that the application tile is updated just like any other secondary tile.

Creating the application tile and setting its default properties

You can't create the application tile programmatically, but you can set its default values (the tile is created manually by the user). Most properties are set within your app's properties, which you access using the **Solution Explorer**. Others have to be set directly in the **WMAppManifest.xml**.

The process is explained in the article How to: Set the Initial Properties for the Application Tile for Windows Phone.

Tip: While you can't create the application tile in code, you can create a secondary tile in code that emulates it. There is an example of this in How to pin the application tile to the start screen from code.

Updating the application tile

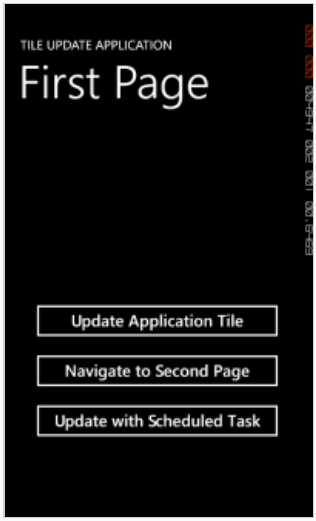
Both application and secondary tiles are updated using the ShellTile and StandardTileData classes:

(C) Copyright Nokia 2014. All rights reserved.

- `ShellTile` provides functions to create tiles, find tiles that already exist based on their properties, and to update their contents
- `StandardTileData` is used to define the properties of the tile for update

The following code snippet shows the common pattern for updating the first tile, which will usually be the application tile (you can check its properties to be certain). `ShellTile.ActiveTiles.First()` or `ShellTile.ActiveTiles.FirstOrDefault()` are used to return the first tile. If a matching tile is found then a `StandardTileData` is created, populated with the new parameters, and then used to update the tile.

```
// modify Application Tile data
private void updateTile_Click(object sender, RoutedEventArgs e)
{
 // some random number
 Random random = new Random();
 // get application tile
 ShellTile tile = ShellTile.ActiveTiles.First();
 if (null != tile)
 {
  // create a new data for tile
  StandardTileData data = new StandardTileData();
  // tile foreground data
  data.Title = "Title text here";
  data.BackgroundImage = new Uri("/Images/Blue.jpg",
UriKind.Relative);
  data.Count = random.Next(99);
  // to make tile flip add data to background also
  data.BackTitle = "Secret text here";
  data.BackBackgroundImage = new Uri("/Images/Green.jpg", UriKind.Relative);
  data.BackContent = "Back Content Text here...";
  // update tile
  tile.Update(data);
 }
}
```



LiveTile Application Main Page Image

The code above is used in the example application, and called when the users presses the **Update Application Tile** button. You can see this in operation by pinning the application to the Start menu, click button in Main Page and see how application tile is modified with a new data.

Secondary Tile

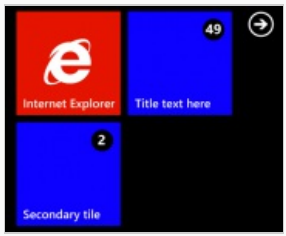
Secondary tiles are created and updated in exactly the same way as above. First we check whether the tile we're interested in exists using `ShellTile`, if it doesn't exist we create it. In either case we use `StandardTileData` to provide the new tile data.

```
// check if secondary tile is already made and pinned
ShellTile Tile = ShellTile.ActiveTiles.FirstOrDefault(x =>
x.NavigationUri.ToString().Contains("Title=SecondaryTile"));
if (Tile == null) {
// create a new secondary tile
StandardTileData data = new StandardTileData();
// tile foreground data
data.Title = "Secondary tile";
data.BackgroundImage = new Uri("/Images/Blue.jpg", UriKind.Relative);
data.Count = 2;
```


(C) Copyright Nokia 2014. All rights reserved.

```
// create a new tile for this Second Page
ShellTile.Create(new Uri("/SecondPage.xaml?Title=SecondaryTile", UriKind.Relative),
data);
}
```

The above code snippet is used in (**SecondPage.xaml**) of the code example. When the example is run, a secondary tile with count of 2 is added (as shown in image below).



Application Secondary Tile

Deleting a tile

To delete tile, first check that tile exists based on its properties and then just delete it. Note that you cannot delete the *Application Tile* added by a user.

```
// remove secondary tile
ShellTile Tile = ShellTile.ActiveTiles.FirstOrDefault(x =>
x.NavigationUri.ToString().Contains("Title=SecondaryTile"));
if (Tile != null) Tile.Delete();
```

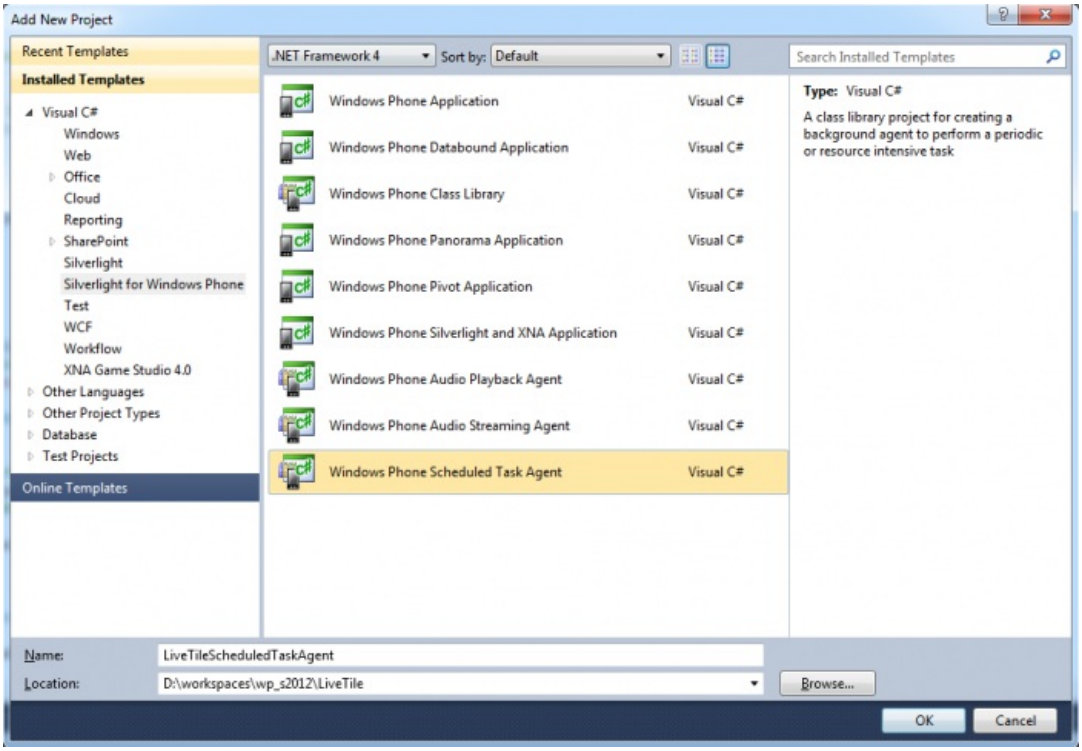
Update Live Tile with Scheduled Task Agent

One possibility to update tile's data is to use Background Agents. Scheduled Task Agent is used in this coding example to demonstrate tile's updating in device.

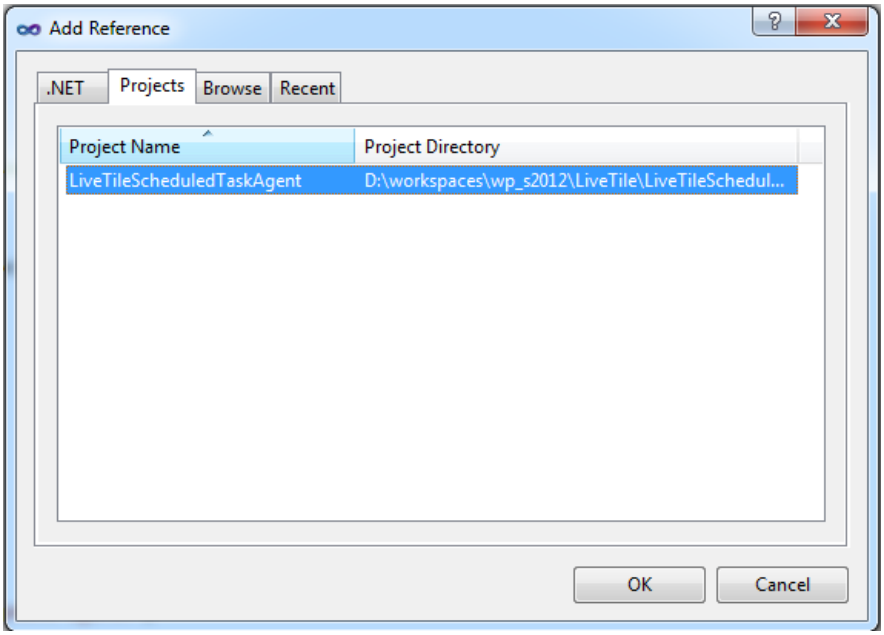
Add a new Scheduled Task Agent to your solution:

1. select your solution from Solution Explorer
2. add a new project
3. select Installed Templates, Silverlight for Windows Phone, Windows Phone Scheduled Task Agent
4. name your project for example LiveTileScheduledTaskAgent

(C) Copyright Nokia 2014. All rights reserved.



Scheduled Task Agent is not working right a way in your project, you should add reference to it from LiveTile project.



Start Agent from MainPage.cs

Start using Scheduled Task Agent from your main project's MainPage.cs class with PeriodicTask Class. This class represents a scheduled task that runs regularly for a small amount of time. In this coding example, `StartPeriodicAgent` method is called from Button in application Main Page.

```
private void StartPeriodicAgent()
{
 // is old task running, remove it
 periodicTask = ScheduledActionService.Find(periodicTaskName) as PeriodicTask;
 if (periodicTask != null)
 {
  try
  {
   ScheduledActionService.Remove(periodicTaskName);
  }
  catch (Exception)
```


(C) Copyright Nokia 2014. All rights reserved.

```
  {
  }
 }
 // create a new task
 periodicTask = new PeriodicTask(periodicTaskName);
 // load description from localized strings
 periodicTask.Description = "This is LiveTile application update agent.";
 // 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(10));
  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."))
  {
   // 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.
 }
}
```

Note: Add `#define DEBUG_AGENT` line in your code (first line) to enable Background Agent debugging more often than once in 30 mins.

Programming LiveTileScheduledTaskAgent Project

Background agent will call `ScheduleAgent.cs` class and more specially it's OnInvoke method to display new data in Live Tile.

```
protected override void OnInvoke(ScheduledTask task)
{
 // some random number
 Random random = new Random();
 // get application tile
 ShellTile tile = ShellTile.ActiveTiles.First();
 if (null != tile)
 {
```


(C) Copyright Nokia 2014. All rights reserved.

```
  // creata a new data for tile
  StandardTileData data = new StandardTileData();
  // tile foreground data
  data.Title = "Title text here";
  data.BackgroundImage = new Uri("/Images/Blue.jpg", UriKind.Relative);
  data.Count = random.Next(99);
  // to make tile flip add data to background also
  data.BackTitle = "Secret text here";
  data.BackBackgroundImage = new Uri("/Images/Green.jpg", UriKind.Relative);
  data.BackContent = "Back Content Text here...";
  // update tile
  tile.Update(data);
 }
#if DEBUG_AGENT
 ScheduledActionService.LaunchForTest(task.Name, TimeSpan.FromSeconds(30));
 System.Diagnostics.Debug.WriteLine("Periodic task is started again: " + task.Name);
#endif

 NotifyComplete();
}
```

Tip: Add `#define DEBUG_AGENT` line in your code (first line) to enable Background Agent debugging more often than once in 30 minutes.

Update Live Tile with Push Notifications

Another option to update tile is to use Push Notifications. There is excellent code sample available in MSDN for Tile Notification using with the Microsoft Push Notification Service - you should try and open that example first to get familiar with Push Notifications.

This coding example uses PHP as a back end to push tile notification to Windows Phone.

Steps:

- obtain Push Notification end point URL, this is done via this application on the phone (and copy pasted to PHP script -> just testing purpose)
- use Client URL Library in PHP to push messages

Getting the Push Notification end point URL

In windows phone HttpNotificationChannel class can be used to try to find the push channel. If the channel was not found, then create a new connection to the push service (if found, just register for all the events) and bind new channel for Tile events. Returned `ChannelUrl` is written to Output windows in this coding example.

```
// this method is called from Main Page Constructor
private void CreatePushChannel()
{
 // try to find the push channel.
 pushChannel = HttpNotificationChannel.Find(channelName);

 // if the channel was not found, then create a new connection to the push service.
 if (pushChannel == null)
 {
  pushChannel = new HttpNotificationChannel(channelName);
  // tegister for all the events before attempting to open the channel.
  pushChannel.ChannelUriUpdated += new
EventHandler<NotificationChannelUriEventArgs>(PushChannel_ChannelUriUpdated);
  pushChannel.ErrorOccurred += new
EventHandler<NotificationChannelErrorEventArgs>(PushChannel_ErrorOccurred);
```


(C) Copyright Nokia 2014. All rights reserved.

```
  pushChannel.Open();
  // bind this new channel for Tile events.
  pushChannel.BindToShellTile();
 }
 else
 {
  // the channel was already open, so just register for all the events.
  pushChannel.ChannelUriUpdated += new
EventHandler<NotificationChannelUriEventArgs>(PushChannel_ChannelUriUpdated);
  pushChannel.ErrorOccurred += new
EventHandler<NotificationChannelErrorEventArgs>(PushChannel_ErrorOccurred);
  // fisplay the URI for testing purposes. Normally, the URI would be passed back to
your web service at this point.
  System.Diagnostics.Debug.WriteLine(pushChannel.ChannelUri.ToString());
 }
}
```

If new `ChannelUri` is found just show it in Output window.

```
void PushChannel_ChannelUriUpdated(object sender, NotificationChannelUriEventArgs e)
{
 // display the new URI for testing purposes. Normally, the URI would be passed back to
your web service at this point.
 System.Diagnostics.Debug.WriteLine(e.ChannelUri.ToString());
}
```

Error handling is sending error message to Output window.

```
void PushChannel_ErrorOccurred(object sender, NotificationChannelErrorEventArgs e)
{
       // error handling logic for your particular application would be here.
       System.Diagnostics.Debug.WriteLine("A push notification {0} error occurred.  {1}
({2}) {3}", e.ErrorType, e.Message, e.ErrorCode, e.ErrorAdditionalData);
 }
```

Using PHP to send Push Notification

This coding example uses Client URL Library (cURL) to send push messages to Windows Phone. Send and Receive Tile Notifications for Windows Phone is described nicely in MSD documentation. Read it carefully and more specially `ButtonSendTile_Click` method to know what kind of XML structure and headers has to be send to Microsoft Push Notification Service.

Note: You have to change `$deviceURL` each time when you run this coding example.

```
<?php
// unique device url
$deviceURL = 'device url here, copy paste it from Output window when application
starts.';

// tile data
$imageURL = "/Images/Purple.jpg";
$count = rand(0,99);
$title = "Push from server!";
```


(C) Copyright Nokia 2014. All rights reserved.

```
// headers
$headers  = array("X-WindowsPhone-Target: token","X-NotificationClass: 1");

// message
$msg = "<?xml version=\"1.0\" encoding=\"utf-8\"?>".
       "<wp:Notification xmlns:wp=\"WPNotification\">".
          "<wp:Tile>".
          "<wp:BackgroundImage>".$imageURL."</wp:BackgroundImage>".
          "<wp:Count>".$count."</wp:Count>".
          "<wp:Title>".$title."</wp:Title>".
          "</wp:Tile>".
       "</wp:Notification>";

// use Client URL Library
$ch = curl_init();
// set an options for a cURL transfer
// look options here:  http://www.php.net/manual/en/function.curl-setopt.php
// and what are needed here: http://msdn.microsoft.com/en-
us/library/windowsphone/develop/hh202970(v=vs.92).aspx
curl_setopt($ch, CURLOPT_URL, $deviceURL);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_HEADER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER,$headers + array('Content-Type: text/xml','Accept:
application/*'));
curl_setopt($ch, CURLOPT_POSTFIELDS, $msg);
// perform a cURL session
curl_exec($ch);
// close a cURL session
curl_close($ch);

?>
```

Related Articles

- Tiles for Windows Phone
- Essential Graphics, Visual Indicators, and Notifications for Windows Phone (start section)
- Application Certification Requirements for Windows Phone
- Anatomy of a Windows Phone Tile

Summary

This code example shows how to use application tiles in Windows Phone. Hope you find this article useful and it helps you work with Windows Phone 7.

You can download source codes here: File:PTMLiveTile.zip.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.