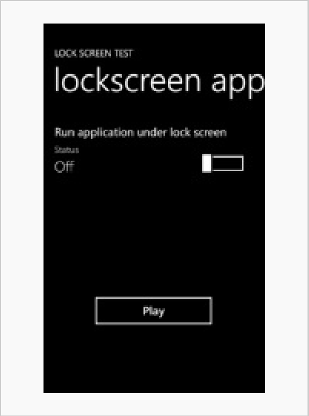
Run Windows Phone application under lock screen

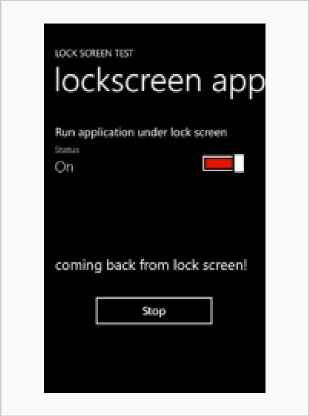
This article demonstrates how to run an application under lock screen on Windows Phone.

Introduction

Many a times I've faced problems while trying to perform some application activities under lock screen i.e. when the lock screen is active. As soon as the lock screen is active, the application goes into a dormant/tombstone mode and thus the application stops running. So, for example, while playing a song when the application goes under lock screen, the song stops playing. There is a technique called Fast Application Switching (FAS) which can be used to resume the song when the application comes back to the foreground. However, in this article we will see how we can play the song when the application is in background without using FAS or any background agent.



Application running in foreground



coming back from lock screen!

Application came back from background

Implementation

Let’s create an empty Windows Phone Project

- Launch Visual Studio
- Click on File
- New Project
- Select Windows Phone Application (Visual C# Template)
- Add Name and Location of the project
- Click OK to create the project.

Once the project is created, we add a MediaElement and set its `Source` property to a media file. A `ToggleSwitch` is being added from Windows Phone Toolkit which is used to alter the application state of running. For more confirmation we have added a TextBlock which notifies when the application returns back from lock screen.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <TextBlock Height="41" FontSize="25" HorizontalAlignment="Left"
Margin="12,24,0,0" Name="textBlock1" Text="Run application under lock screen"
VerticalAlignment="Top" Width="383" />
            <toolkit:ToggleSwitch x:Name="FluxToggle"
                     Header="Status"
                     Checked="ToggleSwitch_Checked"
                     Unchecked="ToggleSwitch_Unchecked"
                     Margin="0,60,0,0"/>
            <Button Content="Play" Height="82" HorizontalAlignment="Left"
Margin="88,388,0,0" Name="button1" VerticalAlignment="Top" Width="277"
Click="button1_Click" />
            <MediaElement AutoPlay="False" Source="/Audio/Song.mp3" Height="0"
HorizontalAlignment="Left" Margin="151,152,0,0" Name="mediaElement1"
```

```
VerticalAlignment="Top" Width="0" />
            <TextBlock FontFamily="Segoe WP" FontSize="31" Height="51"
HorizontalAlignment="Left" Margin="12,307,0,0" Name="textBlockStatus" Text=""
VerticalAlignment="Top" Width="438" />
</Grid>
```

As mentioned earlier, the Windows Phone application goes to dormant/tombstone mode when the phone screen is locked. This behavior can be altered by changing the properties `UserIdleDetectionMode` and `ApplicationIdleDetectionMode` of PhoneApplicationService class. By default `UserIdleDetectionMode` and `ApplicationIdleDetectionMode` are set to `Enable`, which allows the service to lock the phone when the user is idle and these deactivate the application when the phone is locked.

As shown below, we have used `ApplicationIdleDetectionMode` in `ToggleSwitch`, which when set to `Disable` doesn't deactivate the application when the phone is locked.

```
private void ToggleSwitch_Checked(object sender, RoutedEventArgs e)
        {
            MessageBox.Show("The application will run under lock screen!");
            PhoneApplicationService.Current.ApplicationIdleDetectionMode =
IdleDetectionMode.Disabled;
            PhoneApplicationFrame rootFrame = App.Current.RootVisual as
PhoneApplicationFrame;
            if (rootFrame != null)
            {
                rootFrame.Obscured += new
EventHandler<ObscuredEventArgs>(rootFrame_Obscured);
                rootFrame.Unobscured += new EventHandler(rootFrame_Unobscured);
            }
        }
```

Thus, the application continues to run even when the lock screen is activated.

Things to remember

Once `ApplicationIdleDetectionMode` is `Disabled`, it cannot be `Enabled` until the application is re-launched. If you try to enable it, it will throw an exception. To handle the locking and unlocking state of the device we can call the `Obscured` and `Unobscured` event handler of `RootFrame`. `Obscured` will be called when the phone is getting locked. Here we should use minimum CPU, less battery consumption etc. and `Unobscured` gets called when the phone is getting unlocked. This is the stage when we can resume all the states of the application.

Summary

The above code is pretty simple but it should pass the Marketplace Application Certification as mentioned in section 6.3. It states that an application running under a locked screen must take user's permission first to be able to change the settings from application user interface.

Source Code

The full source code of the example is available here: File:LockScreenAppWp.zip