

Archived:Sharing Cinemagraphs in your Windows Phone 8 App

This article explains how to add [Nokia Cinemagraph](#) sharing to your Windows Phone 8 application. In other words, after reading this article you will know how to share "Cinemagraphs" in your own applications on Windows Phone 8.



05 May
2013



Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template `{{ReviewForRemovalFromArchive|user=~~~~|write your reason here}}`.

The latest version of Cinemagraph no longer uses gifs as the "main sharing format" and so no gif is saved to the camera roll - H.264 is used instead. The app now stores two files: jpg preview and a "*.nar" file - a zip archive containing jpg for all frames, animation mask, and xml with metadata. The article has been archived because it does not cover how to work with the .nar file format - if a developer wishes to add this functionality it could be restored from archive.

Introduction

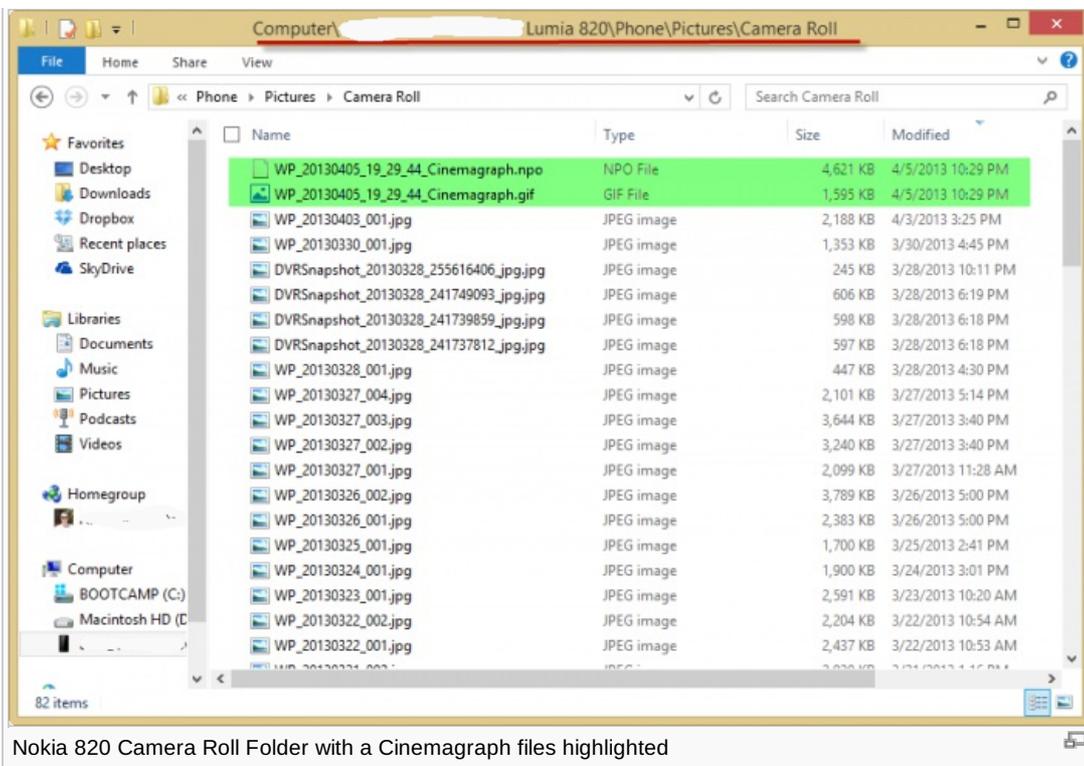


Nokia provides all Windows Phone 8 devices with a very nice app called Cinemagraph which allows one to create, edit and share animated pictures, or "cinemagraphs" (you can find out more on the [blog here](#) or see the video of the Cinemagraph app in action below:

If you're creating a service or an application which is interested in sharing Cinemagraph, just follow the instructions below. You will need real Nokia Windows Phone 8 device to run the sample code.

What is a Cinemagraph?

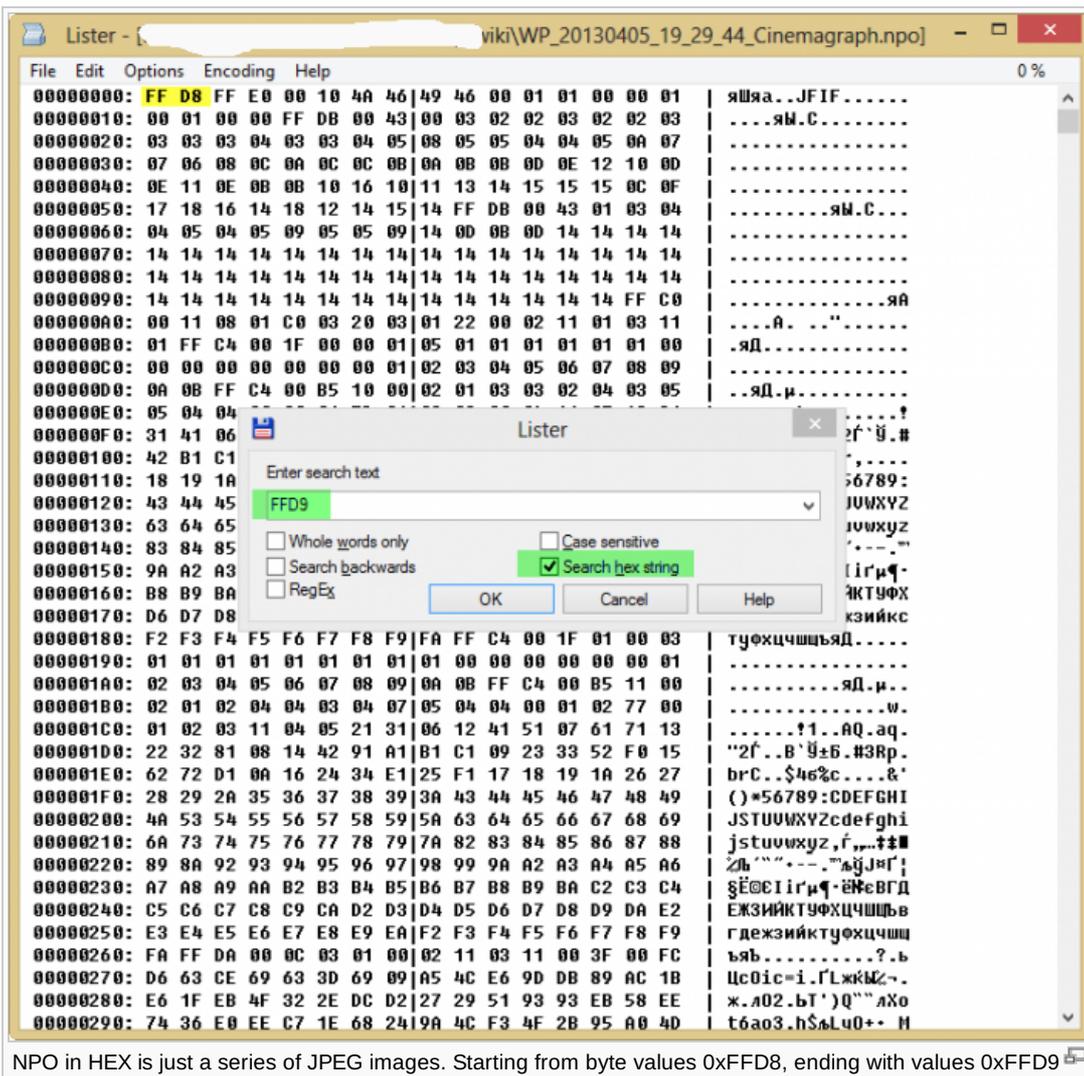
If you ever tried to connect a Windows Phone 8 device to your PC, you may have noticed the device *Pictures->Camera Roll* folder



Nokia 820 Camera Roll Folder with a Cinamagraph files highlighted

As you can see, every Cinamagraph produces 2 files in the Camera Roll: .NPO and .GIF. While .NPO is an internal file format, .GIF is a well-known format for animated pictures.

NPO is actually a series of JPEG files concatenated together with some additional data added to the end (possibly it's animation mask). [JPEG structure](#) states every JPEG file starts with values 0xFF 0xD8 and ends with values 0xFF 0xD9:



NPO in HEX is just a series of JPEG images. Starting from byte values 0xFFD8, ending with values 0xFFD9

So one can easily load this NPO file, do some very easy parsing and get source frames.

Now, when we know what makes up a Cinemagraph picture and where it is located on the device, let's see some code.

Getting Cinemagraphs through code

First create a new Windows Phone app and install there MVVM Light nuget. Let's use <http://nuget.org/packages/MvvmLight/> for a quick start, because this package creates one viewmodel.

```
PM> Install-Package MvvmLight
```

Also, let's install nuget Coding4Fun.Toolkit.Controls - it has BooleanToVisibilityConverter which will be useful later today.

```
PM> Install-Package Coding4Fun.Toolkit.Controls
```

To access Camera Roll, we are going to use **Microsoft.Xna.Framework.Media** namespace from **Microsoft.Xna.Framework** assembly.

Now, open `MainViewModel` and add some properties and commands.

- Properties `HasCinemagraphs` and `Cinemagraphs` will be used to display a list of cinemagraphs or a message that user still has no cinemagraphs yet.
- `SelectedCinemagraph` property is used to keep selected cinemagraph.
- `LoadCinemagraphs` command is used to load or refresh list of cinemagraphs on UI and `ShareCinemagraph` command is used to share selected cinemagraph to SkyDrive.
- `SelectCinemagraph` command is used to overcome WP8 limitation - [LongListSelector.SelectedItem is not bindable](#)

```
using Microsoft.Xna.Framework.Media;

namespace CinemagraphSkyDrive.ViewModel
{
    public class MainViewModel : ViewModelBase
    {
        public MainViewModel()
        {
            ShareCinemagraph = new RelayCommand(OnShareCinemagraphExecuted);
            LoadCinemagraphsCommand = new
RelayCommand(OnLoadCinemagraphsCommandExecuted);
            SelectCinemagraph = new RelayCommand<Picture>(OnSelectCinemagraphExecuted);
        }

        private void OnSelectCinemagraphExecuted(Picture picture)
        {
            SelectedCinemagraph = picture;
        }

        private void OnSignInToSkyDriveExecuted()
        {
        }

        private void OnShareCinemagraphExecuted()
        {
        }

        private void OnLoadCinemagraphsCommandExecuted()
        {
        }
    }
}
```

```

    }

    public RelayCommand LoadCinmagrapsCommand { get; set; }
    public RelayCommand<Picture> SelectCinmagraph { get; set; }
    public RelayCommand ShareCinmagraph { get; set; }

    private bool _hasCinmagraps = false;
    public bool HasCinmagraps
    {
        get { return _hasCinmagraps; }
        set
        {
            if (_hasCinmagraps == value)
                return;
            _hasCinmagraps = value;
            RaisePropertyChanged("HasCinmagraps");
        }
    }

    private List<Picture> _cinmagraps = null;
    public List<Picture> Cinmagraps
    {
        get { return _cinmagraps; }
        set
        {
            if (_cinmagraps == value)
                return;
            _cinmagraps = value;
            RaisePropertyChanged("Cinmagraps");
        }
    }

    private Picture _selectedCinmagraph = null;
    public Picture SelectedCinmagraph
    {
        get { return _selectedCinmagraph; }
        set
        {
            if (_selectedCinmagraph == value)
                return;
            _selectedCinmagraph = value;
            RaisePropertyChanged("SelectedCinmagraph");
        }
    }
}
}
}

```

We're checking all albums for pictures ending with .gif (see highlight) and then sort them by date so on your UI you can show most recent in the beginning (as it is in Pictures hub).



Warning: The following code requires ID_CAP_MEDIALIB_PHOTO capability in WMAppManifest.xml.

The implementation of the OnLoadCinmagrapsCommandExecuted() command handler is as follows:

```
private void OnLoadCinmagrapsCommandExecuted()
```

```
{  
    MediaLibrary library = new MediaLibrary();  
    var allAlbums = library.RootPictureAlbum.Albums;  
    var picturesByAlbums = allAlbums.SelectMany(a => a.Pictures.Where(p =>  
p.Name.ToLower().EndsWith(".gif")));  
    Cinemagraphs = picturesByAlbums.OrderByDescending(p => p.Date).ToList();  
    HasCinemagraphs = Cinemagraphs != null && Cinemagraphs.Count > 0;  
}
```

Now, open **MainPage.xaml** and create some very basic layout.

1. `<phone:PhoneApplicationPage`
2. `x:Class="CinemagraphSkyDrive.MainPage"`
3. `xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"`
4. `xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"`
5. `xmlns:d="http://schemas.microsoft.com/expression/blend/2008"`
6. `xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"`
7. `xmlns:my="clr-
 namespace:Microsoft.Live.Controls;assembly=Microsoft.Live.Controls"`
8. `xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"`
9. `xmlns:i="clr-
 namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"`
10. `xmlns:m="clr-
 namespace:GalaSoft.MvvmLight.Command;assembly=GalaSoft.MvvmLight.Extras.WP8"`
11. `mc:Ignorable="d"`
12. `DataContext="{Binding Main, Source={StaticResource Locator}}"`
13. `>`
14. `<Grid>`

15. `<Grid.RowDefinitions>`

16. `<RowDefinition Height="Auto"/>`

17. `<RowDefinition Height="*/>`

18. `</Grid.RowDefinitions>`

19. `<StackPanel Grid.Row="0" Margin="12,17,0,28">`

20. `<TextBlock Text="CINEMAGRAPH SHARE" Style="{StaticResource
PhoneTextNormalStyle}" Margin="12,0"/>`

21. `<TextBlock Text="sample" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>`

22. `</StackPanel>`

23. `<Grid Grid.Row="1" Margin="12,0,12,0">`

24. `<Grid.RowDefinitions>`

25. `<RowDefinition Height="Auto"/>`

26. `<RowDefinition Height="Auto"/>`

27. `<RowDefinition Height="*/>`

28. `<RowDefinition Height="Auto"/>`

29. `</Grid.RowDefinitions>`

30. `<phone:LongListSelector Grid.Row="2"`

31. `Visibility="{Binding HasCinemagraphs,
Converter={StaticResource BooleanToVisibilityConverter}}"`


```
48. </Button.Template>
49. </Button>
50. </DataTemplate>
51. </phone:LongListSelector.ItemTemplate>
52. </phone:LongListSelector>
53. <StackPanel Grid.Row="2"
54.     Visibility="{Binding HasCinemagraphs,
55.     Converter={StaticResource BooleanToVisibilityConverter},
56.     ConverterParameter=true}">
57.     <TextBlock Margin="12,0" Text="Seems you have no Cinemagraphs"/>
58. </StackPanel>
59. <Button Content="Load cinemagraphs" Command="{Binding
60.     LoadCinemagraphs}" Grid.Row="3" HorizontalAlignment="Left"/>
61. <Button Content="Upload selected" Command="{Binding ShareCinemagraph}"
62.     Grid.Row="3" HorizontalAlignment="Right"
63.     Visibility="{Binding SelectedCinemagraph,
64.     Converter={StaticResource NotNullToVisibilityConverter}}"/>
65. </Grid>
66. </Grid>
67. </phone:PhoneApplicationPage>
```

Lines from 30 to 51 display list of cinemagraphs, next we display message when user has no cinemagraphs.

At the end, there are two buttons - the first for loading cinemagraphs, the second for uploading a selected cinemagraph to SkyDrive.

The converters which are used are pretty simple, and you'll find them in the sample code package.

Uploading Cinemagraph file to SkyDrive

Let's do the sharing. In this article we will be sharing cinemagraphs to user's SkyDrive. It's very easy, just follow these steps:

1. Install Live SDK for WP8. Easiest is to use [NuGet package LiveSDK](#)

```
PM> Install-Package LiveSDK
```

2. Generate or take existing Live Client ID. Read ["Configuring your App" in Live SDK MSDN Section](#).
3. Modify **MainPage.xaml** and add SignIn button and TextBlock to display the logged-in user. The code below is inserted into second Grid, approximately at line 30.

```
30.         <my:SignInButton Name="btnSignin" ClientId="00000000400F1EAD"
31.                               Scopes="wl.signin wl.skydrive_update wl.basic"
32.                               Branding="Skydrive" TextType="SignIn"
33.                               HorizontalAlignment="Left"
34.                               VerticalAlignment="Top">
35.             <i:Interaction.Triggers>
36.                 <i:EventTrigger EventName="SessionChanged">
37.                     <m:EventToCommand PassEventArgsToCommand="True"
38.                         Command="{Binding LiveIdSessionChanged}"/>
39.                 </i:EventTrigger>
40.             </i:Interaction.Triggers>
41.         </my:SignInButton>
42.         <TextBlock Margin="12,0" Grid.Row="1" Text="{Binding
43.             UserName}"/>
```

4. Modify **MainViewModel** by adding command **LiveIdSessionChanged**, property **UserName** and field of type **LiveConnectClient**:

```
private LiveConnectClient _client;

public MainViewModel()
```

```
{
    /* .... */
    LiveIdSessionChanged = new
RelayCommand<LiveConnectSessionChangedEventArgs>(OnLiveIdSessionChangedExecuted);
    /* .... */
}

public RelayCommand<LiveConnectSessionChangedEventArgs>
LiveIdSessionChanged { get; set; }

private string _userName = null;
public string UserName
{
    get { return _userName; }
    set
    {
        if (_userName == value)
            return;
        _userName = value;
        RaisePropertyChanged("UserName");
    }
}

private async void
OnLiveIdSessionChangedExecuted(LiveConnectSessionChangedEventArgs e)
{
    string userName = null;

    if (e.Status == LiveConnectSessionStatus.Connected)
    {
        _client = new LiveConnectClient(e.Session);
        LiveOperationResult operationResult = await
_client.GetAsync("me");
        try
        {
            dynamic meResult = operationResult.Result;
            if (meResult.first_name != null &&
                meResult.last_name != null)
            {
                userName = "Hello " +
                    meResult.first_name + " " +
                    meResult.last_name + "!";
            }
            else
            {
                userName = "Hello, user!";
            }
        }
        catch (LiveConnectException exception)
        {
            userName = "Error calling API: " +
                exception.Message;
        }
    }
    else
    {
        userName = "Not signed in.";
    }
}
```

```

        _client = null;
    }

    DispatcherHelper.CheckBeginInvokeOnUI(() =>
    {
        UserName = userName;
    });
}

```

Implementation of LiveIdSessionChanged command handler is [almost completely borrowed from MSDN article](#). I think you'll forgive me this move because goal of this article is to show how can you share cinemagraphs, not how to use LiveSDK correctly.

5. Implement OnShareCinemagraphExecuted() method (handler of ShareCinemagraph command)

```

private async void OnShareCinemagraphExecuted()
{
    if (SelectedCinemagraph != null && _client != null)
    {
        try
        {
            var picture = SelectedCinemagraph;
            string folderId = await PrepareFolder("Cinemagraphs");
            using (var stream = picture.GetImage())
            {
                var uploadResult =
                    await
                        _client.UploadAsync(folderId, picture.Name,
picture.GetImage(), OverwriteOption.Overwrite);
            }
            DispatcherHelper.CheckBeginInvokeOnUI(() =>
            {
                MessageBox.Show("Upload completed");
            });
        }
        catch (Exception e)
        {
            DispatcherHelper.CheckBeginInvokeOnUI(() =>
            {
                MessageBox.Show("Upload error:\r\n" + e.ToString());
            });
        }
        Debug.WriteLine("SkyDrive upload completed");
    }
}

```

We're going to upload all cinemagraphs into single folder with name "Cinemagraphs", so we need to get Folder ID from SkyDrive first. That's what method PrepareFolder() does.

```

private async System.Threading.Tasks.Task<string> PrepareFolder(string
folderName)
{
    var searchResult = await _client.GetAsync("me/skydrive/search?q=" +
folderName);
    dynamic folder = null;
}

```

```

        if (searchResult.Result != null)
        {
            var searchResults = (List<object>)searchResult.Result["data"];
            folder = searchResults.FirstOrDefault();
        }

        //create new one if required
        if (folder == null)
        {
            var folderData = new Dictionary<string, object>();
            folderData.Add("name", folderName);
            var createResult = await _client.PostAsync("me/skydrive",
folderData);
            folder = createResult.Result;
        }

        if (folder == null)
            throw new InvalidOperationException("SkyDrive folder was not
found");

        return folder.id;
    }

```

6. Add binding converters' code.

BooleanToVisibilityConverter and NotNullToVisibilityConverter are quite common, so you may already have them in your favourite framework.

PictureToImageSourceConverter converter is specially made for this project.

Anyway, below is code for all three:

```

    public class PictureToImageSourceConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            var picture = value as Picture;
            if (picture != null)
            {
                var bitmapImage = new BitmapImage()
                {
                    CreateOptions = BitmapCreateOptions.BackgroundCreation
                };
                bitmapImage.SetSource(picture.GetThumbnail());
                return bitmapImage;
            }
            return null;
        }

        public object ConvertBack(object value, Type targetType, object
parameter, CultureInfo culture)
        {
            return null;
        }
    }

    public class BooleanToVisibilityConverter : IValueConverter

```

```

{
    #region IValueConverter Members

    public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
    {
        bool valueAsBool = (bool)value;
        if (parameter != null)
            valueAsBool = !valueAsBool;

        Visibility result = valueAsBool ? Visibility.Visible :
Visibility.Collapsed;

        return result;
    }

    public object ConvertBack(object value, Type targetType, object
parameter, System.Globalization.CultureInfo culture)
    {
        var vis = (Visibility)value;
        return vis == Visibility.Visible;
    }

    #endregion
}

public class NotNullToVisibilityConverter : IValueConverter
{
    #region IValueConverter Members

    public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
    {
        var valueIsNotNull = value != null &&
!string.IsNullOrEmpty(value.ToString());
        if (parameter != null)
            valueIsNotNull = !valueIsNotNull;

        var result = valueIsNotNull ? Visibility.Visible :
Visibility.Collapsed;
        return result;
    }

    public object ConvertBack(object value, Type targetType, object
parameter, System.Globalization.CultureInfo culture)
    {
        throw new NotImplementedException();
    }

    #endregion
}

```

7. Update **App.xaml** and add there 3 converters

Here's a little tip for you: copy converter's type name into x:Key field and you'll never mess with remembering correct **StaticResource** handle.

```
<c:BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter"/>
<c:NotNullToVisibilityConverter x:Key="NotNullToVisibilityConverter"/>
<c:PictureToImageSourceConverter x:Key="PictureToImageSourceConverter"/>
```

8. Don't forget to initialize DispatcherHelper in **App.xaml.cs** in App ctor method.

```
DispatcherHelper.Initialize();
```

Now we're finished with main parts, so let's add some more goodness.

Some more goodness

I'd suggest to show a tip what is a Cinemagraph app, what is a cinemagraph and a button to install Cinemagraph app from the Marketplace if user has no cinemagraphs at his Camera Roll.

Here's the code for the installation shortcut:

```
63. <StackPanel Grid.Row="2"
64.             Visibility="{Binding HasCinemagraphs,
65.             Converter={StaticResource BooleanToVisibilityConverter},
66.             ConverterParameter=true}">
67.     <TextBlock Margin="12,0" Text="Seems you have no Cinemagraphs"/>
68.     <Button Content="Install Cinemagraph app" Command="{Binding
        InstallCinemagraph}"/>
68. </StackPanel>
```

```
public RelayCommand InstallCinemagraph { get; set; }

public MainViewModel()
{
    /* .... */
    InstallCinemagraph = new RelayCommand(OnInstallCinemagraphExecuted);
    /* .... */
}

private void OnInstallCinemagraphExecuted()
{
    var task = new MarketplaceDetailTask();
    task.ContentIdentifier = "594477c0-e991-4ed4-8be4-466055670e69";
    task.Show();
}
```

Don't forget to verify if it's Nokia device ;)

```
/// <summary>
/// Class to check device brand and OS version for compatibility with Cinemagraph
/// </summary>
public static class NokiaDevice
{
    /// <summary>
    /// http://msdn.microsoft.com/en-
us/library/windowsphone/develop/jj720574(v=vs.105).aspx
    /// </summary>
    private static Version Targeted78Version = new Version(7, 10, 8858);
    private static Version Targeted80Version = new Version(8, 0);

    public static bool IsNokiaDevice { get; private set; }

    public static bool Is780rLaterVersion { get; private set; }
    public static bool IsWP8 { get; private set; }

    public static bool SupportsCinemagraph { get; private set; }

    static NokiaDevice()
    {
        Is780rLaterVersion = System.Environment.OSVersion.Version >=
Targeted78Version;
        IsWP8 = System.Environment.OSVersion.Version >= Targeted80Version;
        IsNokiaDevice =
(DeviceStatus.DeviceManufacturer.ToLower().Contains("nokia"));

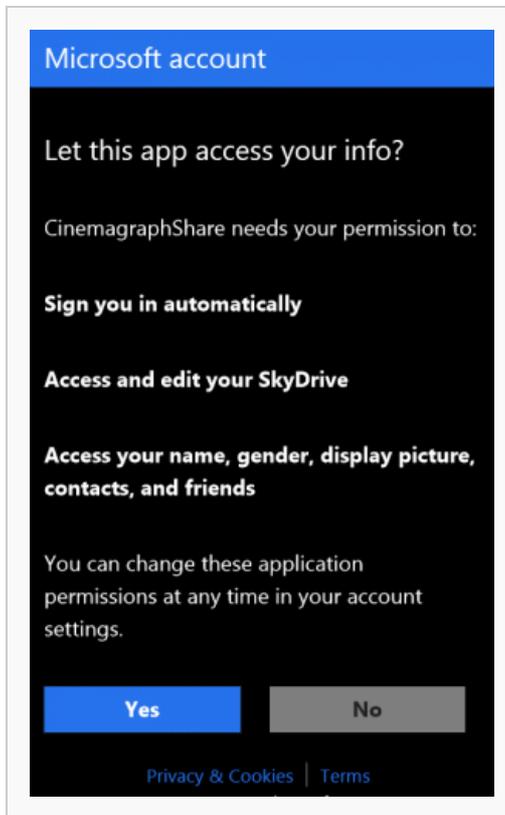
        SupportsCinemagraph = Microsoft.Devices.Environment.DeviceType ==
DeviceType.Emulator || (IsNokiaDevice && Is780rLaterVersion);
    }
}
```

Everything at one place

Quick screenshot gallery how sample application works on Nokia Lumia 820:



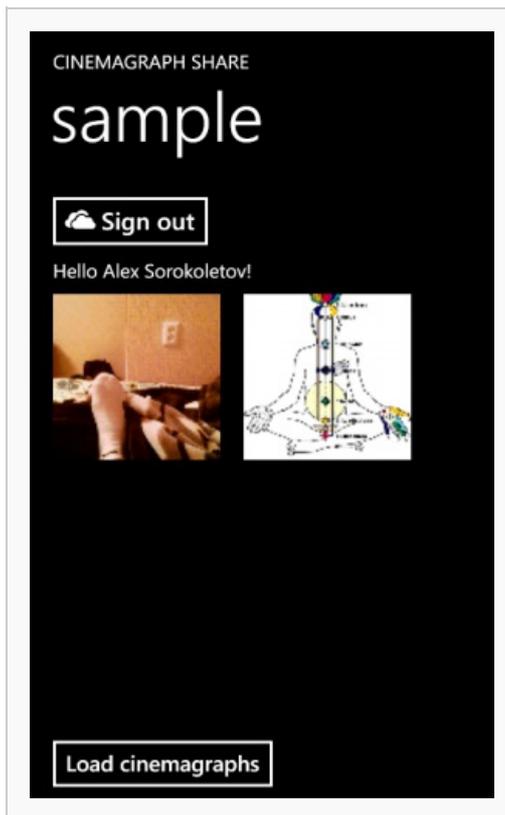
This how sample app looks after running it



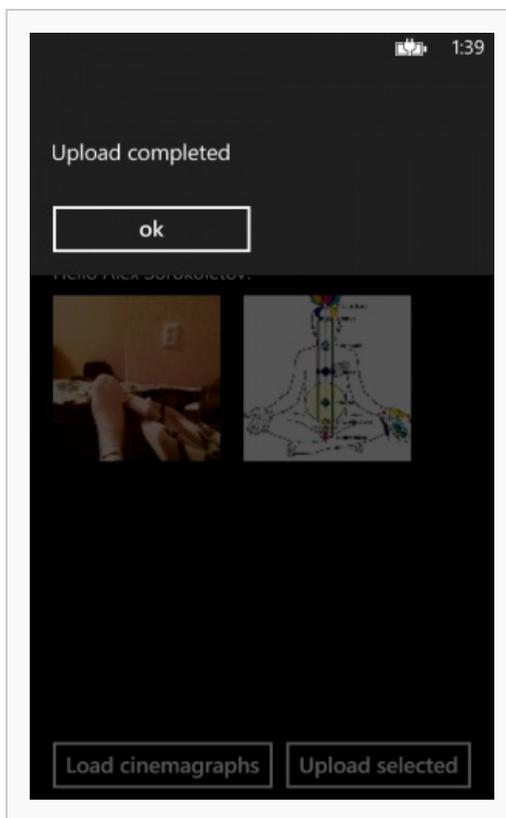
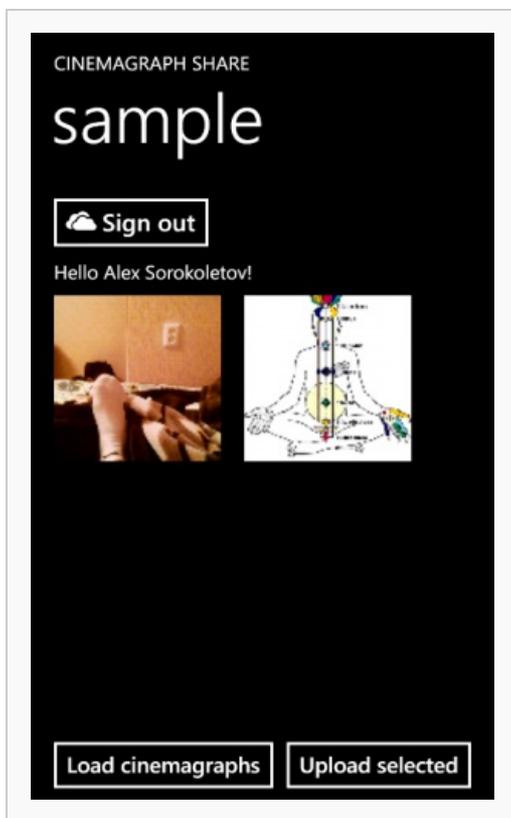
SkyDrive login consent



After logging to SkyDrive you will see your SkyDrive account name and Sign Out button

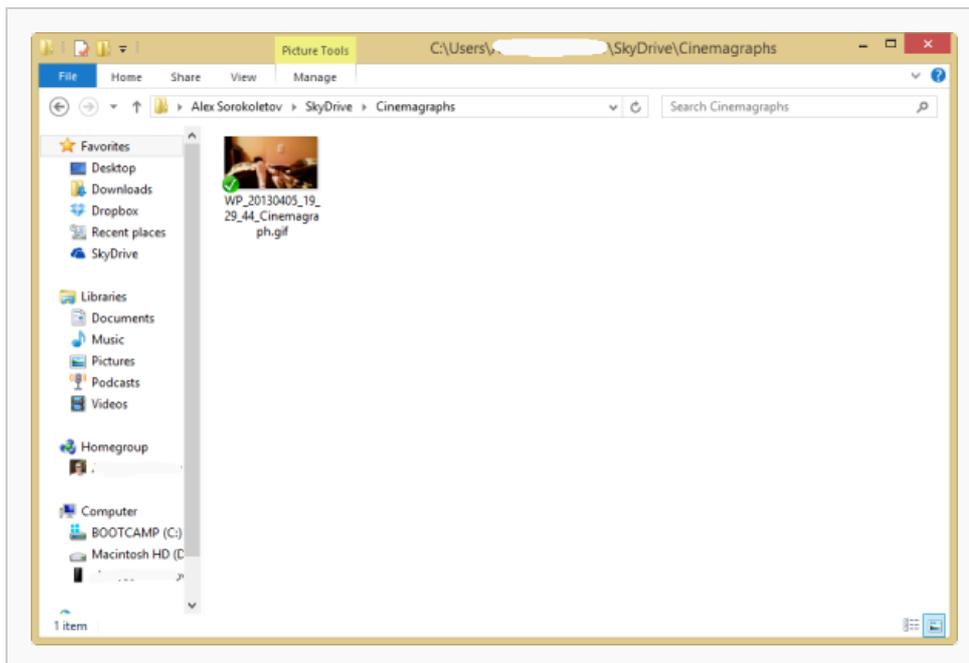


List of cinemagraphs after pressing "Load cinemagraphs" button



After selecting the cinemagraph you can upload the image to SkyDrive using "Upload selected cinemagraph" button

Message after completed upload



Synchronized cinemagraph in my local SkyDrive folder

Cinemagraph support on Windows Phone 7.8?

Well, the story is short. On Windows Phone 7.8 Cinemagraph app store GIFs to its isolated storage folder and doesn't save files into Camera Roll.

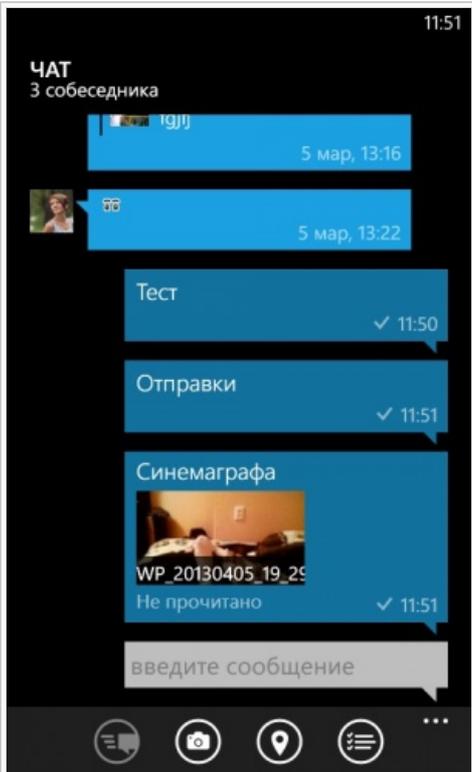
Until Nokia update the app, we can't access cinemagraph files on Windows Phone 7.8

Summary

I hope this article will help Windows Phone community integrating this cool Nokia Windows Phone 8 feature called Cinemagraphs into all existing and some new apps. Before I finish, I'd like to extend my **special thanks to Nokia Department in Moscow for providing Windows Phone 8 developer devices!**

If you have ideas on improving the article or code, feel free to say that in comments or in twitter (here's my twitter handle [@AlexSorokoletov](#)). This article has a sample application source code attached.

To view this code working in a production application, download [VK Messenger](#) to your Windows Phone 8 Nokia device.



Example how Cinemagraph is integrated into VK Messenger