FORUM NOKIA

# Adobe Flash Lite Photostream Application Tutorial

Version 1.0; 25 March 2010




NOKIA

### Disclaimer



### Licence

# Contents

## Change history

| | | |
|---|---|---|
| 25 March 2010 | Version 1.0 | Initial document release |

# 1 Introduction

This tutorial, which was made possible by the Open Screen Project Fund, is one in a series of seven. Each tutorial in this series is designed around a user experience where Adobe Flash Lite and Nokia devices are a great fit: rich media, personalised experiences, social media, photo sharing, location based information, music, news, video, advertising, and marketing.

The focus of this tutorial is on using remotely stored image files to create an image gallery. This application is designed to be shown as-is or modified easily. By modifying the application it can be used to quickly present a concept within your company or to pitch to outside clients.

An additional feature of the demonstration application is that it can be reskinned easily: The UI elements and the background can be changed in a matter of minutes.

Please check out the other tutorials in this series, as the concepts taught can be used together in a single application. The other tutorials are:

- Event — Displays information about an event and enables the user to send an SMS message containing text event information or call an event telephone number.
- Video Player — Plays an external `*.flv` video file in both landscape and portrait modes.
- Music Player — Plays external `*.mp3` audio files.
- RSS News — Reads and displays RSS feeds.
- Social Media — Leverages Twitter for promotional purposes.
- LBS Coffee Finder — Leverages GPS location and Google Maps to find the nearest coffee shop.

## 1.1 Prerequisites

These tutorials have been designed for developers with intermediate Flash skills or above.

## 1.2 Requirements

To modify the applications Adobe Flash CS 3 or CS4 is required. A S60 5th Edition device by Nokia is recommended to show the demonstrations. Use of a device is more effective than presenting on a computer.

If you don't have a device, you can run the application using the Forum Nokia Remote Device Access or Virtual Developer Lab services. These services provide access to a range of Nokia devices over the internet.

You can use Adobe Device Central in Flash CS4 also.

## 1.3 Flash Lite skills taught

This tutorial is designed to teach you the skill to use the following capabilities of Flash Lite on Nokia devices:

- Use of the Flickr API.
- Use of the `loadClip` command to load images over a network connection.
- Parsing XML data.

### 1.4 Running the applications

The following methods can be used to run the demonstration applications on a S60 5th Edition device:

- Use a Bluetooth connection to send the `*.swf` files to your device. The Flash Lite application can then be run by opening it from the device's messaging application.
- Create a new folder called *trusted* in the *other* folder on any drive of your S60 device. Copy the `*.swf` files to the *trusted* folder. Open the folder in the device's file manager application and tap any `*.swf` file to run it.
- Install the widget (`*.wgz` file) onto your device and then locate the widget in the device's applications folder. Tap the application icon to run the application.

## 2 About the author

Omega Mobile is a technology-driven design studio that obsesses about mobile and devices. The company is passionate about clean design, simple user interfaces, and targeted experiences. Its specialty is mobile design, user interfaces, user experiences, prototypes, and Flash Lite. Omega Mobile has been producing mobile Flash applications since 1999. For help on a mobile project or for more information, please visit http://www.omegamobile.com or contact Omega Mobile at +1-415-596-6342 or contact2010@omegamobile.com.

## 3 Open Screen Project Fund

Announced by Nokia and Adobe in 2009, the Open Screen Project Fund awards grants to help developers create exciting new applications and content. Grants are made from a fund of US$10 million and the fund will remain available until December 2010. The fund also provides marketing and educational support for the Open Screen Project, which aims to establish cross-platform runtimes, remove development and distribution barriers, and innovate through industry collaboration. To learn more about the Fund, visit Get Started page of the Fund's website.

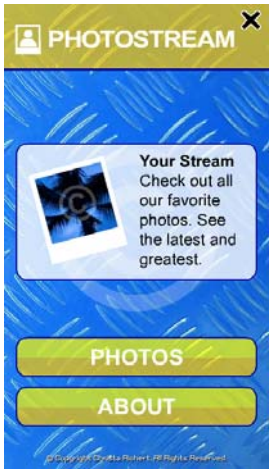
# 4 Application description

The photostream demonstration application is designed to teach you how to make use of image files stored on a remote server. In addition, the application includes code that illustrates how to create a gallery for browsing content items.

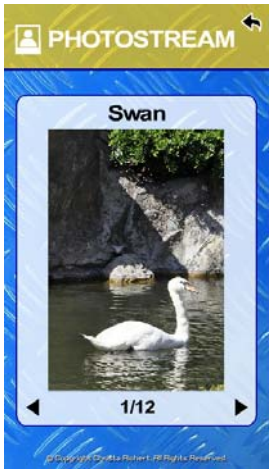
## 4.1 Application screens

The application provides the following screens:

Home




Photos




About


### 4.1.1 Home screen

This screen provides a panel containing an image and text appropriate to the photo stream and enables the user to access the feature of the application through three buttons:

- Photos — Opens the photo screen.
- About — Opens the about application screen.
- Quit icon — Exits the application.

### 4.1.2 Photos screen

The Photos Screen provides a user driven slideshow of the photo stream content. When opened the screen load the most recent image. A spinner provides a navigation guide indicating the current image sequence number from the total number of images in the stream. The screen provides the following buttons:

- Left icon — Loads the previous image or, if on the first image, loads the last image.
- Right icon — Loads the next image or, if on the last image, loads the first image.
- Back icon – Returns to the home screen.

### 4.1.3 About screen

The about screen is designed provide a space to describe the application and the company that created it. It is recommended that information about your company, including your contact details, is

used when distributing this application to potential clients. The screen contains a back icon button, which takes the user back to the home screen.

## 4.2 Code

The bulk of the code (see Example 1) dealing with the image stream is used by the photos screen. The code is found in the PHOTO block.

```
/*
Called when image is loaded from the server
mc:Object - movieclip that the image is loaded into
*/
function onLoadInit(mc:MovieClip)
{
   pictCont._xScale = 100;
   pictCont._yScale = 100;
   adjustPhotoSize();
   setTitle();
   setBottomText();
   spinnerOff();
}


/*
Sets the title of the picture
*/
function setTitle()
{
   var stringToShow = String(images[curImage][PICT_TITLE]);
   if(stringToShow == "(Untitled)"){
      //If the flickr user has not specified a title for the picture.
      stringToShow = "";
   }
   titleCont.textCont.pictTitle.text = stringToShow;
}


/*
Adjusts the photo movieclip size to fit in the panel correctly. Scales
the picture proportionally so that the width does not exceed the width
of the bounding panel and the height does not interfere with the other
ui elements.
*/
function adjustPhotoSize()
{
   CONTAINING_BOX = panel1;

   var origHeight = pictCont._height;
   var origWidth = pictCont._width;

   var isNeedResizePotrait = origHeight > MAX_PORTRAIT_HEIGHT;
   var isNeedResizeLandscape = origWidth > MAX_LANDSCAPE_WIDTH;
   var isNeedResize = isNeedResizePotrait || isNeedResizeLandscape;

   var isPortrait = origHeight > origWidth;

   isOverRatio = (origWidth/origHeight) > MAX_RATIO;
   var imageScale = 100;

   if(isPortrait)
   {
      if(isNeedResizePotrait)
      {
         imageScale = (MAX_PORTRAIT_HEIGHT/origHeight) *100;

      }
```

```
        else if(isOverRatio && isNeedResize)
        {
            imageScale = (MAX_LANDSCAPE_WIDTH/origWidth) *100;
        }

    }
    else if (!isPortrait )
    {
        if(isNeedResizeLandscape)
        {
            imageScale = (MAX_LANDSCAPE_WIDTH/origWidth) *100;
        }
    }
    pictCont._yScale = imageScale;
    pictCont._xScale = imageScale;

    var newImageWidth = pictCont._width;
    var newImageHeight = pictCont._height;
    var xSpacing = (CONTAINING_BOX._width - newImageWidth)/2;
    pictCont._x = xSpacing + CONTAINING_BOX._x;

    var ySpacing = (CONTAINING_BOX._height - newImageHeight)/2;
    pictCont._y = ySpacing + CONTAINING_BOX._y;
}


/*
Loads the data from the feed and places it in an array
*/
function loadPhotos()
{
    spinnerOn();
     photosXML = new XML();
     photosXML.ignoreWhite = true;
     var pictNumToSet = 0;
     photosXML.onLoad = function(success)
    {
         if(success)
        {
            var rootNode:XMLNode = photosXML.firstChild;
            images = new Array();
            for(var i = 0; i<rootNode.childNodes.length; i++)
            {
                if(rootNode.childNodes[i].nodeName == "entry")
                {
                    for(var j = 0; j
<rootNode.childNodes[i].childNodes.length; j++)
                    {
                        var currentNodeName =
rootNode.childNodes[i].childNodes[j].nodeName;
                        var currentValue =
rootNode.childNodes[i].childNodes[j].firstChild;
                        if(currentNodeName == "title")
                        {
                            var pictTitle = String(currentValue);
                        }
                        if(currentNodeName == "id")
                        {
                            var rawIdString = String(currentValue);
                            var locOfSlash = rawIdString.lastIndexOf("/");
                            var pictId = rawIdString.substring(locOfSlash+1);
                        }
                    }
                    images[pictNumToSet] = new Array();
                    images[pictNumToSet][PICT_TITLE] = pictTitle;
                    images[pictNumToSet][PICT_NUM] = pictId;
                    pictNumToSet ++;
                }
```

```
            }
            totalNumImages = images.length;
            loadComplete();
          }
          else
        {
              trace("*ERROR: Unable to load XML!!*");
          }
     }
   photosXML.load(FEED_URL + FEED_USER);
}


/*
Called when feed data is done loading
*/
function loadComplete()
{
   getCorrectUrl();
}


/*
Sets the bottom line text showing the picture number and how many total
pictures there are
*/
function setBottomText()
{
   var curImageString = String(curImage + 1);
   var stringToShow = curImageString + "/" + totalNumImages;
   bText.textCont.bottomLine.text = stringToShow;
}


/*
Uses the flickr api to get the available photo sizes for the current
photo to show.
*/
function getCorrectUrl ()
{
   photoToCheck = images[curImage][PICT_NUM];
   photoSizeXML = new XML();
   photoSizeXML.ignoreWhite = true;
   photoSizeXML.onLoad = function(success)
   {
      if(success)
      {
         var linkNode:XMLNode;
         var rootNodeSize:XMLNode = photoSizeXML.firstChild.firstChild;
         var isMedImage = false;
         for( i = 1; i < rootNodeSize.childNodes.length; i++)
         {
            var currentData = rootNodeSize.childNodes[i].attributes;
            var labToUse = String(currentData.label);
            if( labToUse == "Medium")
            {
               var realLink = String(currentData.source);
               var imageWidth =  String(currentData.width);
               var imageHeight =  String(currentData.height);
               isMedImage = true;
            }
            else if(labToUse == "Large")
            {
               var realLinkLarge = String(currentData.source);
               var imageWidth =  String(currentData.width);
               var imageHeight =  String(currentData.height);
            }
         }
```

```
            if(isMedImage)
            {
               var linkToUse = realLink;
            }
            else
            {
               var linkToUse = realLinkLarge;
            }
            images[curImage][PICT_PATH] = linkToUse;
            loadCurrentImage();
         }
          else
        {
               trace("*ERROR: Unable to load XML!!*");
          }
      }
   photoSizeXML.load( PHOTO_SIZE_URL + photoToCheck);
}


/*
Advanced to the next picture and starts loading it.
*/
function nextPict()
{
   spinnerOn();
   curImage++;
   if(curImage >= totalNumImages)
   {
      curImage = 0;
   }

   if(images[curImage][PICT_PATH])
   {
      loadCurrentImage();
   }
   else
   {
      getCorrectUrl();
   }
}


/*
Goes to the previous picture and starts loading it.
*/
function prevPict()
{
   spinnerOn();
   curImage--;

   if(curImage<0)
   {
      curImage = totalNumImages-1;
   }
   if(images[curImage][PICT_PATH])
   {
      loadCurrentImage();
   } else {
      getCorrectUrl();
   }
}


/*
Loads the current picture to show
*/
```

```
function loadCurrentImage()
{
   pictCont.unloadClip();
   loader.loadClip( images[curImage][PICT_PATH] , pictCont);
}
```

Example 1: The code employed by the photostream application's photo screen.

# 5 Skinning

## 5.1 Introduction

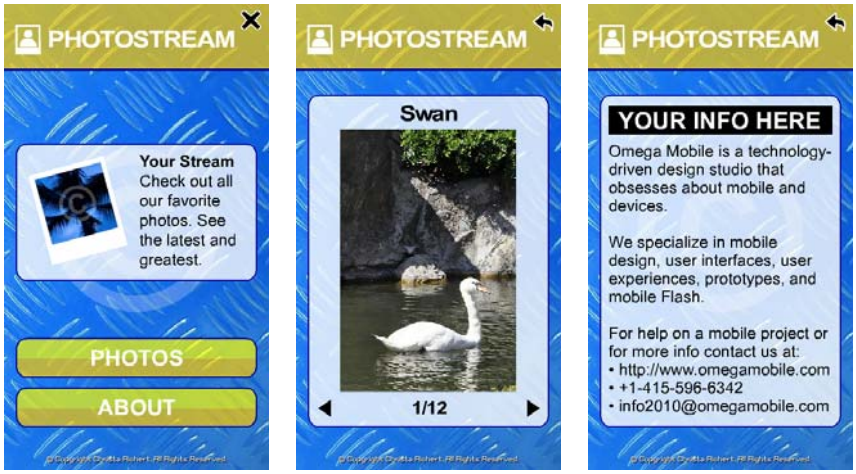
The application illustrates a number of methods that can be used to skin UI elements:

- Buttons, panels, the header, and scrollbars are skinned using preferences set in code.
- The background graphic can be changed by replacing the graphic.
- The text and the graphics used within the application's panels can be changes in code.

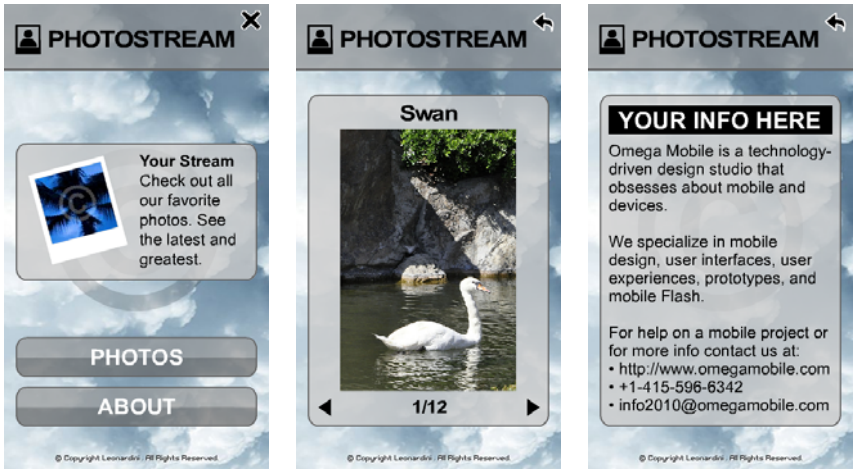
## 5.2 Example skins

The application has been build using three example skins:

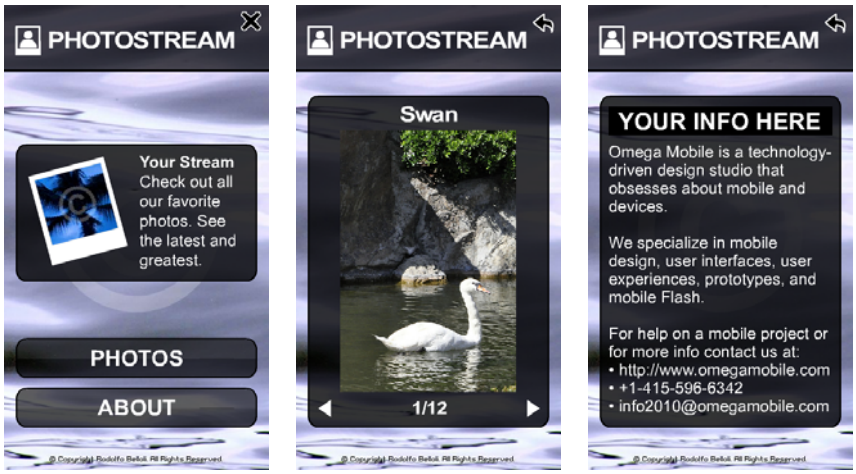
- Skin 1




- Skin 2




- Skin 3



### 5.3 Component skinning using preferences

The `setPrefs` function (see Example 2) is used to define the application's main skinning preferences. These preferences define:

- Colour and alpha levels for the buttons, panel, and header.
- Your Flickr API key.
- The Flickr feed to be loaded.

```
/*
Declare colors for the ui elements and skinning code.
Declare the url for the feed and the user number of the feed
If you're reskinning the application, most of the reskinning can be done
here.
*/
function setPrefs()
{
   //IMPORTANT: YOU MUST GET YOUR OWN API KEY AND INSERT IT HERE TO GET
THE DEMO TO WORK
   //Apply for your own API key here:
http://www.flickr.com/services/apps/create/apply
   API_KEY = "insert_your_32_digit_Flickr_API_key_here_as_a_string";

   FEED_URL =
"http://api.flickr.com/services/feeds/photos_public.gne?id="
   PHOTO_SIZE_URL =
"http://api.flickr.com/services/rest/?method=flickr.photos.getSizes&api_
key=" + API_KEY + "&photo_id=";

   //Change your feed here.
   FEED_USER = "46692107@N02";

   MAX_PORTRAIT_HEIGHT = 360;
   MAX_LANDSCAPE_WIDTH = 324;
   MAX_RATIO = MAX_LANDSCAPE_WIDTH/MAX_PORTRAIT_HEIGHT;

   /**/
   //SKIN 1
   COLOR_BUTTON_UP_STROKE = 0x000099;
   COLOR_BUTTON_UP_TEXT = 0xFFFFFF;
   COLOR_BUTTON_UP_OVERLAY = 0xFEFE00;
   COLOR_BUTTON_UP_BG = 0xFFCC00;
   COLOR_BUTTON_UP_STROKE_ALPHA = 100;
   COLOR_BUTTON_UP_BG_ALPHA = 70;
   COLOR_BUTTON_UP_OVERLAY_ALPHA = 70;

   COLOR_BUTTON_DOWN_STROKE = 0x000099;
   COLOR_BUTTON_DOWN_TEXT = 0xFFFFFF;
   COLOR_BUTTON_DOWN_OVERLAY = 0xFEFE00;
   COLOR_BUTTON_DOWN_BG = 0xFEFE00;
   COLOR_BUTTON_DOWN_STROKE_ALPHA = 100;
   COLOR_BUTTON_DOWN_BG_ALPHA = 100;
   COLOR_BUTTON_DOWN_OVERLAY_ALPHA = 100;

   COLOR_PANEL_STROKE = 0x000099;
   COLOR_PANEL_BG = 0xFFFFFF;
   COLOR_PANEL_BG_ALPHA = 80;
   COLOR_PANEL_STROKE_ALPHA = 100;

   COLOR_HEADER_BG = 0xFFCC00;
   COLOR_HEADER_BG_ALPHA = 70;
   COLOR_HEADER_STROKE = 0x000099;
   COLOR_HEADER_TEXT = 0xFFFFFF;
   COLOR_HEADER_ICON = 0xFFFFFF;

   COLOR_TEXT = 0x000000;
```

```
...//MORE CODE
}
```

Example 2: The code used to set the skinning preferences for the photostream application.

### 5.4 Content skinning

The text and picture in the main panel of the following screens can be changed by modifying the `*.fla` file:

- Home screen.
- About screen.

If you plan to modify and distribute this application, it is recommended that you replace the content in the about screen.

### 5.5 Button text

The text used on any button can be changed by updating the code on the movie clip that contains the button. You will see code similar to this on each movie clip that contains a button:

```
onClipEvent(load){
  txt.btnText.text = "BUTTON TEXT";
}
```

Replace `BUTTON TEXT` with the text you would like displayed on the button.

### 5.6 Backgrounds

You can choose a different background by:

- Going into the bg_mc movie clip on the main timeline.
- Guiding out the images you don't want.
- Turn off the guides on the image you do want.

You can also import a new background image and place it in the bg_mc movie clip.

## 6 Packaging in a widget

There are three methods that can be employed to distribute Flash Lite content for installation on Nokia devices:

- The Flash Lite files can be distributed in a ZIP file that the user unpacks and then copies the content to their device. The user then runs the application by opening the main `*.swf` file from their device's file browser application.
- To pack the `*.swf` file and content into a Symbian Installation System (SIS) file. This requires use of the Symbian build tools and signing of the SIS file with a self-create certificate. The SIS file is then installed by the user and the application run from an icon in the device's application menu or from a shortcut on the device's home screen.
- Embed the Flash Lite content within a Web Runtime (WRT) widget. The widget is then installed by the user and the application run from an icon in the device's application menu or from a shortcut on the device's home screen. The widget may also be created to include a read-only home screen view.

Embedding Flash Lite within a widget is the recommended approach to distributing Flash Lite content. This method offers the user a simple installation method, access to the content in the device menu or home screen and a simple packaging method (zipping the widgets content into a file with the extension `*.wgz`) that does not require signing.

In addition, widgets enable you to combine Flash with content created with standard web technologies (such as HTML, CSS, and the JavaScript™ language). As many websites use both HTML and Flash, by using Flash Lite within a widget a mobile optimised version of a rich internet experience can be made available for mobile devices. As such, this combination of technologies provides an attractive way for media and content partners, social networking companies, and other internet based businesses to optimise their web property for consumption on mobile devices by leveraging their existing web development skills.

**Note:** The WRT security model prevents Flash Lite content from making use of any device integration features, such as the ability to read contacts or determine a device's location. If Flash Lite content within a widget needs access to such information, it can be obtained using the JavaScript™ APIs and the data passed to the Flash Lite content using the `externalInterfaces` class.

To package the demonstration application into a widget, using the files in the `photostream_widget` folder included in the tutorial package, take the following steps:

- In `main.html`, update *Widget Page Title* to reflect your application name.
- If desired, create a graphic of 312 x 85 pixels to represent your widget in home screen. In `main.html`, update `widget_homescreen_image.jpg` to refer to the new graphic.
- Publish your swf file (or take the one that came in this demo folder) and, if necessary, in `main.html` update `widget_name.swf` to refer to the new swf file.
- In `info.plist`, update *Widget Name* to reflect the name of the widget you would like to be displayed in the device's menu. You can also update the graphic that's displayed for the icon by replacing or updating `icon.png`.
- In `info.plist`, update `com.demos.widget_name` with the path on the device you want to use for the application data.
- Zip the widget folder and change the extension of the resulting file from `*.zip` to `*.wgz`.

For more information on packaging Flash Lite in a widget see the document *Packing Flash Lite content in a WRT widget* included in the tutorial package.

# 7 Further areas to explore

The gallery could be enhanced by providing a transition as the user moves between images. The gallery could be given an option to display the image in full-screen mode. Additionally, the code could be updated to detect swipe gestures to enable navigation between pictures.

# 8 Additional resources

The following articles from the Forum Nokia Flash Lite Wiki relate to the code discussed in this tutorial:

- How to load and parse XML files.
- Zoom and Rotate Gestures in Flash Lite for touch-enabled devices.
- How to detect touch gestures in Flash Lite.

In addition, the Forum Nokia Flash Lite: Live XML Data Integration Example will be of interest.

You can also find extensive information on using Flash Lite in Nokia devices in the Flash Lite Developer's Library. There are additional documents and code examples available from the Documentation section of the Forum Nokia website.

You can also find a wealth of information on the Flash Lite on Nokia Devices discussion board.

## 9 Evaluate this resource

Please spare a moment to help us improve documentation quality and recognise the resources you find most valuable, by rating this resource.