Tutorial

DEVELOPER WORLD THE FAST TRACK FROM MIND TO MARKET

February 2009

Handling external resources with Project Capuchin

for Flash and Java developers

Sony Ericsson

Preface

About this tutorial

This Project Capuchin tutorial illustrates how to access external resources and data from Flash using the ExternalResourceHandler class in the Project Capuchin API.

Sony Ericsson Developer World

At www.sonyericsson.com/developer, developers find the latest technical documentation and development tools such as phone White papers, Developers guidelines for different technologies, Getting started tutorials, SDKs (Software Development Kits) and tool plugins. The Web site also features news articles, go-to-market advice, moderated discussion forums offering free technical support and a Wiki community sharing expertise and code examples.

For more information about these professional services, go to the Sony Ericsson Developer World Web site.

This Tutorial is published by:

Sony Ericsson Mobile Communications AB,
SE-221 88 Lund, Sweden

www.sonyericsson.com/

First edition (February 2009)
Publication number: 1224-6863.1

Prerequisites

In this tutorial Java ME™ and Adobe™ Flash™ CS3 technologies are used for creating a thumbnail viewer.

The following steps are required for setting up the working environment.

- Install Eclipse and the Java runtime environment (JRE) on your system.
  Eclipse is available for download at http://www.eclipse.org/downloads/
  JRE is available for download at http://www.java.com/en/download/manual.jsp

- Install the Eclipse ME plug-in, available for download at http://eclipseme.org/docs/installation.html

- Install the Sony Ericsson SDK for the Java ME available at
  http://developer.sonyericsson.com/site/global/docstools/java/p_java.jsp

- Include the Project Capuchin classes in Eclipse. They are available for download at
  http://developer.sonyericsson.com/site/global/docstools/projectcapuchin/p_projectcapuchin.jsp

- Install Adobe Flash CS3 or any Flash version that is compatible with Flash Lite 2.1 and ActionScript 2. Also make sure that the Adobe Device Central is installed, this is used for choosing the targeted platform and for emulation purposes during the testing phase. Adobe Device Central is available for download at www.adobe.com/products/creativesuite/devicecentral.

Typographical conventions

In this document code examples are written in Courier font:
`FlashPlayer.createFlashPlayer(flashImage,flashCanvas);`

Trademarks and acknowledgements

Adobe, Adobe Flash Lite and Adobe Flash are either trademarks or registered trademarks of Adobe Systems Incorporated in United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc, in the U.S. and other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

Document history

Change history		
2009-02-02	Doc. no. 1224-6863.1	First version published on Developer World

Contents

Introduction 6

Tutorial
The Thumbnail Viewer project 7
The Flash part 7
Creating a new project 7
Data Transfer 10
The Java part 11

Introduction

This Project Capuchin tutorial illustrates how to access external resources and data from Flash using the ExternalResourceHandler class of the Project Capuchin API.

The ExternalResourceHandler interface is used to load external resources that are referenced within Flash content. The interface has a default implementation, provided by the Capuchin API, that handles external files (`file:///`), Internet resources (`http://`) and MIDlet resources (`res://`).

In this tutorial the default handler is overriden by using a custom implementation of the interface to provide a variety of use cases that can encounter Project Capuchin developers. Thus the method `requestResource()` is implemented to handle calls for ActionScript `loadVars()` and `loadMovie()` functions. The `loadVars()` function is used to retrieve names of the newest three pictures in the camera album, while `loadMovie()` calls are used to load thumbnails of each image that has been previously loaded by `loadVars()`.

The tables below were extracted from the Project Capuchin JavaDocs, and shows the interfaces and methods that must be implemented.

ExternalResourceHandler	
`void`	`requestResource(FlashImage image, java.lang.String URI)` This method is invoked when an external resource is required by the Flash content

When the `URI` argument (resource) is available inside the method, the FlashImage `resourceAvailable()` is invoked.

FlashImage	
`void`	`resourceAvailable(java.lang.String URI, java.io.InputStream resourceData)` This method is called when an external resource is available

The application consists of a screen where the three newest pictures in the Camera album are displayed in Flash. In ActionScript we use `LoadVars()` to get the picture names and once they are retrieved another call is made to load each of the pictures separately. The source files and Eclipse project for the thumbnail viewer are found in the “ThumbnailViewer.ZIP” file packed together with this tutorial.

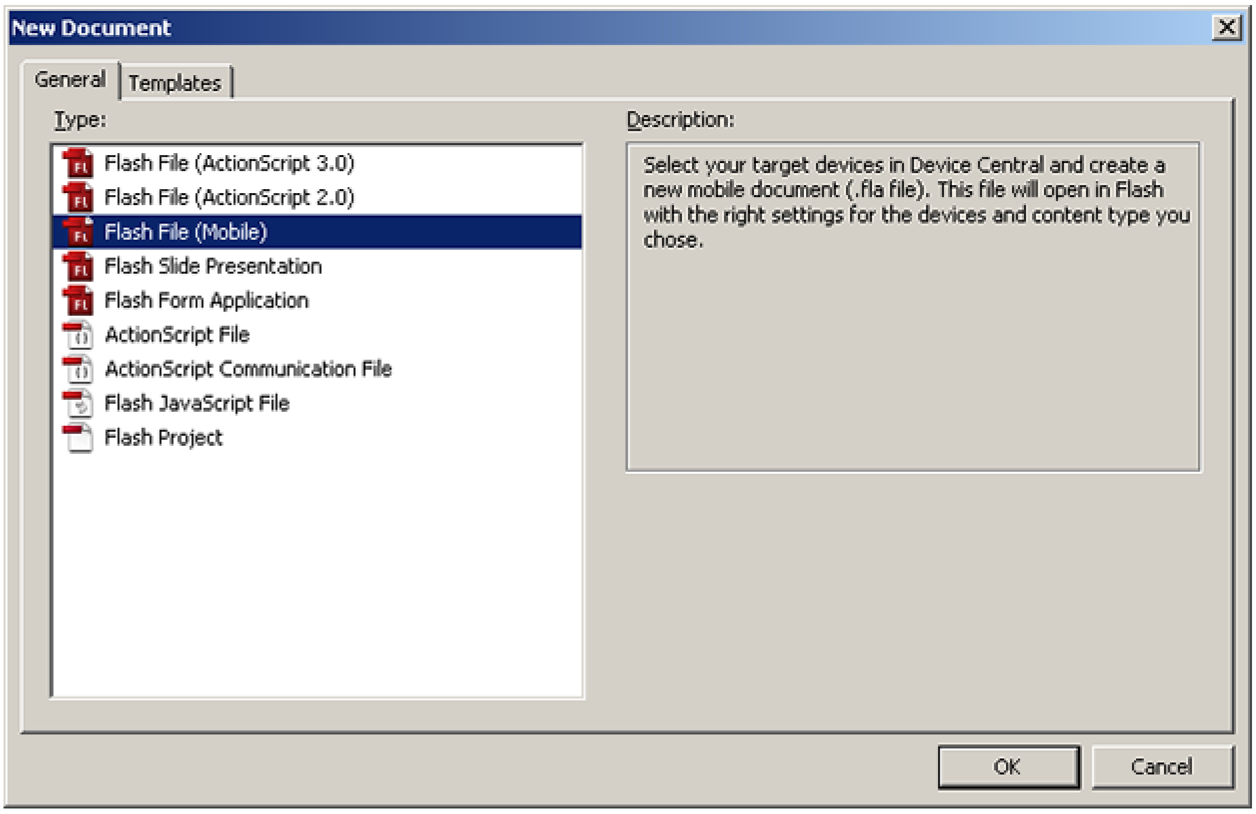
Tutorial
The Thumbnail Viewer project

The Flash part

The Flash file (.fla file) is created in Flash CS3. After compiling, the .fla file is published as a Shockwave (.swf) file. To create a .swf file for a Sony Ericsson phone, the .fla file content needs to be exported/ compiled as a Flash Lite application.

Creating a new project

To create a new .fla file, in Flash CS3 select *File – New* from the menu, select *Flash File (Mobile)* and click the OK button.

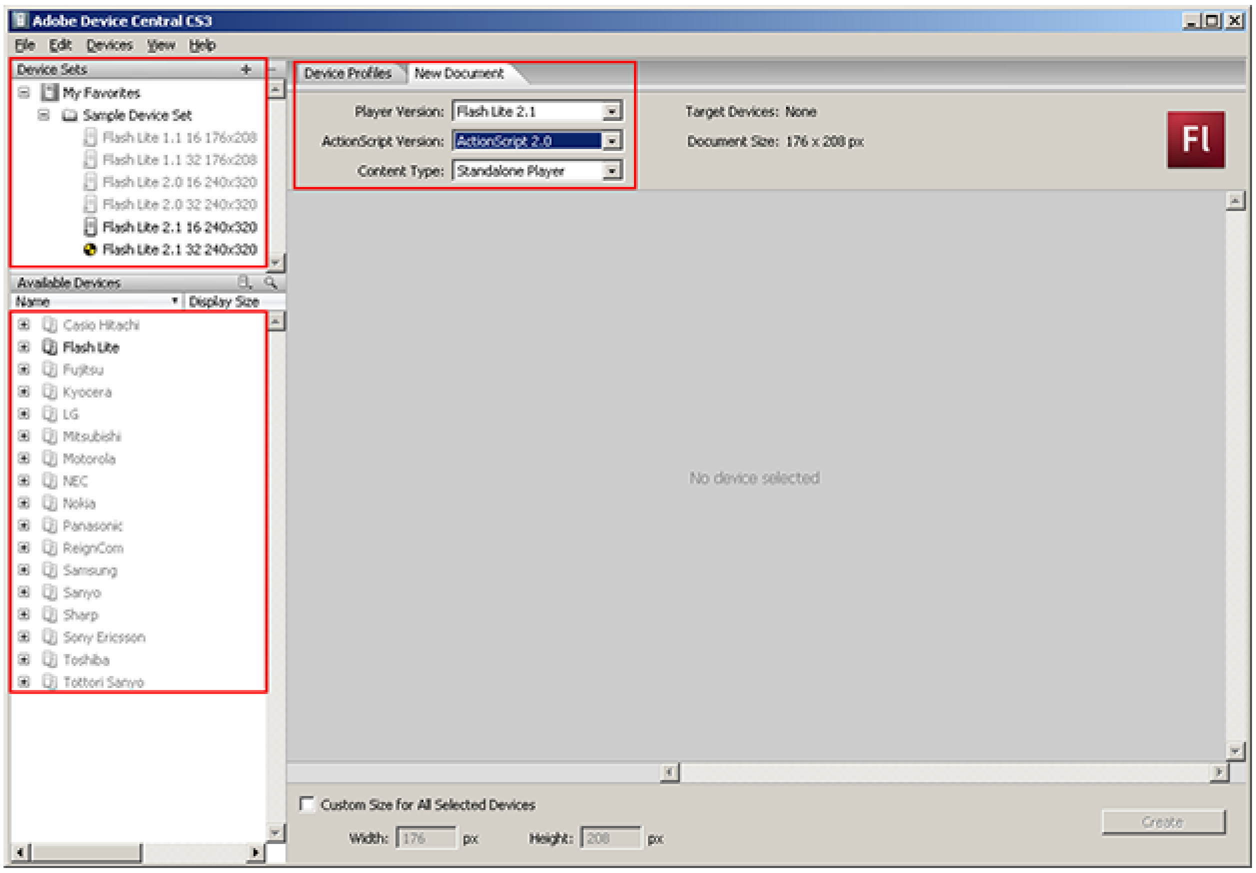


Select your target devices in Device Central and create a new mobile document (.fla file). This file will open in Flash with the right settings for the devices and content type you chose.

Adobe Device Central window opens. In this window the targeted phone profile can be selected, either from the brand names in the list found on the left side or from the generic devices list.

In Adobe Device Central window, under the *New Document* tab, specify the Flash player, ActionScript version and content type. The appropriate Content Type for most configurations is "Standalone Player".

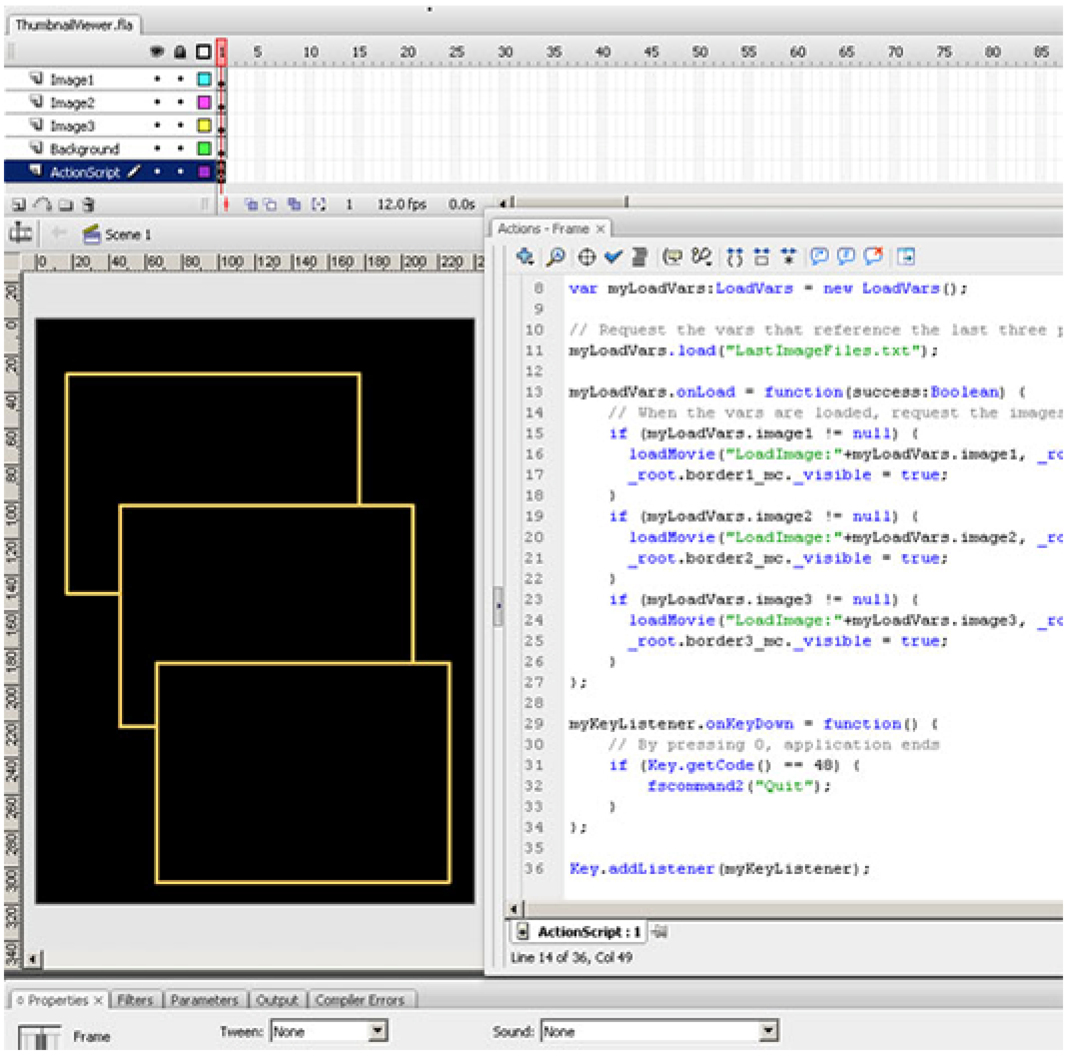
Click the *Create* button to create and open the new project.


Player Version: Flash Lite 2.1
ActionScript Version: ActionScript 2.0
Content Type: Standalone Player
Target Devices: None
Document Size: 176 x 208 px

The new project space opens up. This is the stage area where the Flash content such as movie clips, ActionScript, layers, bitmaps, and so on, can be placed. Note, that in the stage properties inspector, the background colour can be modified to a colour of your choice. This will be the colour for the canvas when the Project Capuchin application runs on the mobile phone. Also note that the frame rate per second can be specified here. This will affect how fast the animation will run, 20 fps is usually sufficient.

The Flash file in this application consists of a screen that automatically displays the snapshot of three pictures received from Java. The .fla file consists of 4 parts, the background and three movie clips where to load the images. The background is just a black screen. There is also some ActionScript for receiving the variables containing the name of the three pictures and for requesting and loading each picture.

The screenshot below demonstrates how the thumbnail viewer example looks in Flash CS3.

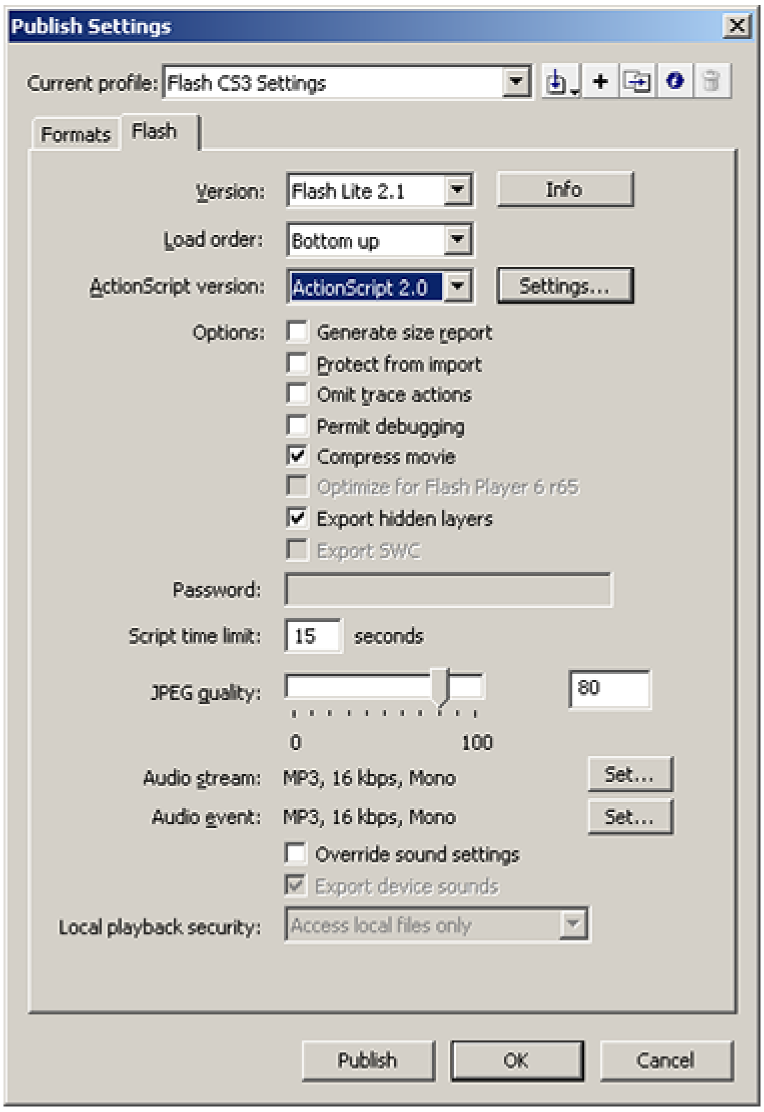


```
var myLoadVars:LoadVars = new LoadVars();

// Request the vars that reference the last three
myLoadVars.load("LastImageFiles.txt");

myLoadVars.onLoad = function(success:Boolean) {
    // When the vars are loaded, request the image
    if (myLoadVars.image1 != null) {
      loadMovie("LoadImage:"+myLoadVars.image1, _r
      _root.border1_mc._visible = true;
    }
    if (myLoadVars.image2 != null) {
      loadMovie("LoadImage:"+myLoadVars.image2, _r
      _root.border2_mc._visible = true;
    }
    if (myLoadVars.image3 != null) {
      loadMovie("LoadImage:"+myLoadVars.image3, _r
      _root.border3_mc._visible = true;
    }
};

myKeyListener.onKeyDown = function() {
    // By pressing 0, application ends
    if (Key.getCode() == 48) {
        fscommand2("Quit");
    }
};

Key.addListener(myKeyListener);
```

The final step is to compile the Flash content and publish it as a .swf file. Select *File – Publish Settings* from the main menu. A dialog with three tabs, *Formats*, *Flash* and *HTML*, appears. First select the *Formats* tab and uncheck *HTML* since we are not exporting Flash content for the web. Ignoring this step will not cause any harm because all what it does is to create HTML and JavaScript files for web publishing.

To compile the Flash file and export it as a .swf file, select the *Flash* tab and check that the Flash Lite version is set properly. The Flash Lite version depends on the targeted platform. All Project Capuching enabled phones support Flash Lite 2.x. Also make sure that the ActionScript version is compatible with the Flash Lite version that has been specified for the targeted phone.

The remaining options are not essential for this example to work. In case the frame rate per second needs to be changed, this can be done in the Script time limit box. For better image quality, the JPEG quality compression can be changed to the desired figure. The higher the number, the better the image quality will be when the .swf file is played on the phone. This may affect the performance, so make sure that the content is optimised appropriately.



Current profile: Flash CS3 Settings
Version: Flash Lite 2.1
Load order: Bottom up
ActionScript version: ActionScript 2.0
Options:
- ☐ Generate size report
- ☐ Protect from import
- ☐ Omit trace actions
- ☐ Permit debugging
- ☑ Compress movie
- ☐ Optimize for Flash Player 6 r65
- ☑ Export hidden layers
- ☐ Export SWC

Password:
Script time limit: 15 seconds
JPEG quality: 80 (0 – 100)
Audio stream: MP3, 16 kbps, Mono
Audio event: MP3, 16 kbps, Mono
- ☐ Override sound settings
- ☑ Export device sounds

Local playback security: Access local files only

Click the *Publish* button to compile and export the project as a .swf file. This is the file needed for the remaining part of the Capuchin application to be completed in Eclipse.

Data Transfer

ActionScript makes two different calls to Java. The first call is for a text file that contains variable values. These values are loaded into Flash variables and used to make the second data request for image files.

The `LoadVars()` method passes a string object as a parameter. This parameter is the file path that contains the files to be loaded.

```
var myLoadVars:LoadVars = new LoadVars();
myLoadVars.load("LastImageFiles.txt");
myLoadVars.onLoad = function(success:Boolean) {
```

Java will handle the call to load and send back the requested data to Flash. Data will be stored in the `LoadVars` object.

The text file containing the variable – value pairs shall have the following format:

```
varName1=VarValue1&varName2=VarValue2&….&varNameX=VarValueX
```

Loaded variables will be available as myLoadVars properties:

```
myLoadVars.varNameX
```

Three different calls are then made to load the proper picture using the passed values stored as properties in myLoadVars object.

The `LoadMovie()` method has two parameters, the first one is a string, which references the file to be loaded and the second one is the target MovieClip in Flash which will hold the loaded resource.

```
loadMovie("LoadImage:"+myLoadVars.image1, _root.image1_mc);
loadMovie("LoadImage:"+myLoadVars.image2, _root.image2_mc);
loadMovie("LoadImage:"+myLoadVars.image3, _root.image3_mc);
```

The Java part

To create a Java application with Capuchin, some basic steps must be followed:

1. Create a FlashImage object with the Flash content.

2. Create a FlashPlayer, attaching the FlashImage.

3. Create a FlashCanvas that will be set as the current Displayable.

A complete tutorial explaining how to load a Flash file inside a Java application using Capuchin can be found at http://developer.sonyericsson.com/getDocument.do?docId=100378.

The Java application contains all the logic and is responsible for handling all the requests from the Flash part. To create a stream between Java and Flash, the `ExternalResourceHandler` interface is used.

Capuchin has a default implementation for `ExternalResourceHandler`. This is used if nothing is specified when creating a Flash image.

```
flashImage = FlashImage.createImage(is, null);
```

The Thumbnail Viewer project implements the `ExternalResourceHandler` interface, so that it becomes responsible for providing Flash with the requested resources.

The reason for doing this, instead of using the default handler, is to provide a different use case. The thumbnail viewer project provides image thumbnails, instead of the actual image files.

Making the Java class implement this interface, obligates it to implement its method.

```
public class ThumbnailViewer extends MIDlet implements
ExternalResourceHandler {

...

public void requestResource(FlashImage fImage, String resource){
```

When creating the flashImage object is specified this is the ExternalResourceHandler, so the default is not used.

```
flashImage = FlashImage.createImage(is, this);
```

Both the Flash calls to loadVars and loadMovie are handled by `requestResource()`, so, the Java part must know how to handle the Flash requests.

```
FileConnection fc = null;
DataInputStream dis = null;
InputStream is = null;
byte[] imgArray = null;

// Receives the call to get the last 3 images from file system
if (resource.endsWith(LAST_IMAGE_FILES)) {
  // Call do get the newest photos from camera
  getLastPicturestoFile();
  fc = (FileConnection)Connector.open(LAST_IMAGE_FILES_FULL_PATH);
  dis = fc.openDataInputStream();
  fImage.resourceAvailable(resource, dis);
} else if (resource.startsWith(LOAD_IMAGE)) {
  fc =
(FileConnection)Connector.open(resource.substring(resource.indexOf(":") +
1));
  dis = fc.openDataInputStream();
  imgArray = new byte[(int)fc.fileSize()];
  dis.readFully(imgArray);
  is = getThumb(imgArray);
  fImage.resourceAvailable(resource, is);
}
```

This code snippet shows how Java is handling the resources to Flash. When receiving a loadVars request, it creates a file with the newest three pictures taken with the mobile camera and passes this file to Flash.

When a picture is requested, Java gets its thumbnail and sends it to Flash.

The requested data must be passed to flash as an InputStream object by calling:

```
fImage.resourceAvailable(resourceString, InputStream);
```

The newest pictures are searched in phone memory and memory card when available, and is then sent to Flash in a file with contents like:

```
image1=file:///c:/camera/100MSDCF/DSC00004.jpg&image2=file:///c:/camera/
100MSDCF/DSC00005.jpg&image3=file:///c:/camera/100MSDCF/DSC00006.jpg
```

Image thumbnails are extracted from JPEG Exif. See http://www.exif.org

```
private InputStream getThumb(byte[] imgArray) {
```

Complete code for both methods, `getLastPicturestoFile()` and `getThumb()` can be found in the "*ThumbnailViewer.zip*" archive, attached to this document.