FORUM NOKIA

# Flash Lite: Tips and Tricks

Version 1.0; August 14, 2008

NOKIA

## Contents

## Change history

| | | |
|---|---|---|
| August 14, 2008 | Version 1.0 | Initial document release |
| | | |

# 1 Introduction

This document lists some useful tips and tricks to assist in creating content and applications using Adobe Flash Lite on Nokia platforms. The tips are presented in no particular order or hierarchy, and are intended for useful guidance only. Where relevant, the version of Flash Lite has been indicated.

# 2 Working with numeric text entry

Flash Lite supports the `SetInputTextType fscommand2` function for specifying whether user input is limited to numeric or alphanumeric characters. Placing the following code in the Flash Lite ActionScript (FLA) will configure the Flash Lite player to allow numeric characters only for text entry into the text field with the variable name `mytextfield`.

```
fscommand2("SetInputTextType", "mytextfield", "Numeric");
```

The `SetInputTextType fscommand2` function takes two arguments. The second argument is the variable name of the affected text field, in this case `mytextfield`. The third argument determines the characters the user can enter into the text field, in this example `Numeric`. Review the Flash Lite help documentation for more information about setting input types using this `fscommand2`.

For Flash Lite 1.1 and Flash Lite 2.0, the `SetInputTextType fscommand2` only affects input text fields on the `_root` timeline. For these versions of the Flash Lite player, it is not possible to specify an input type for text fields inside movie clips. This problem has been fixed in Flash Lite 2.1 for Series 40 5th Edition devices and Flash Lite 3.0 for S60 3rd Edition, Feature Pack 2 devices. Flash Lite 3.0 also displays text entry directly in the text field instead of the device text entry screen for a more intuitive user experience. For backward compatibility with Flash Lite 1.1 and 2.0, use numeric text input text fields on the `_root` timeline.

Review the example FLA and SWF located in the “setinputtype” folder included in the article download files.

# 3 Using a valid SIM card

To view Flash Lite screen savers on Series 40 devices from Nokia, the device must have an active subscriber identity module (SIM) card installed. Series 40 devices without an active SIM card will play and animate SWF assigned to wallpaper but not SWF screen savers. This is not an issue for S60 devices from Nokia.

# 4 Distributing SWF over the air

Over-the-air (OTA) download is a convenient way to distribute personalization content such as screen savers and wallpapers from mobile commerce WML and XHTML mobile sites. Nokia mobile devices are designed to identify content in SWF format as intended for screen saver and wallpaper animation. Before delivering Flash Lite content over the air, configure the Web server to send the `application/x-shockwave-flash` mime type for SWF files so the device will be able to identify the file as Flash Lite content and act accordingly.

## 4.1 Series 40 devices

After completely downloading the SWF file, a Series 40 device will display a dialog message box asking the user to save the SWF in the Images folder. The user will then see the following series of screens to preview and assign the SWF as a screen saver or wallpaper.

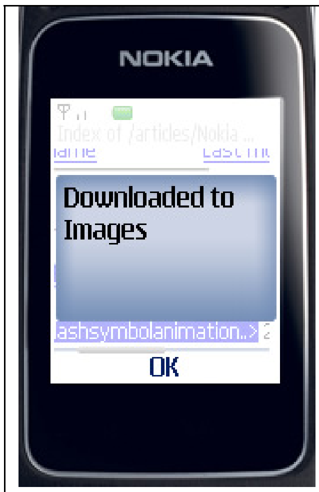


Figure 1: Device asks user to save SWF in Images folder.

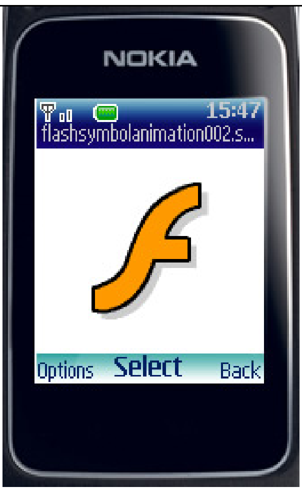
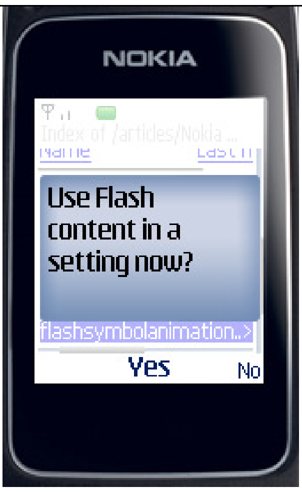


Figure 2: Device launches SWF in standalone player for user to preview content.


NOKIA
Index of /articles/Nokia ...
Use Flash content in a setting now?
flashsymbolanimation..>
Yes
No
Figure 3: User presses Back button to view Use Flash content dialog box.

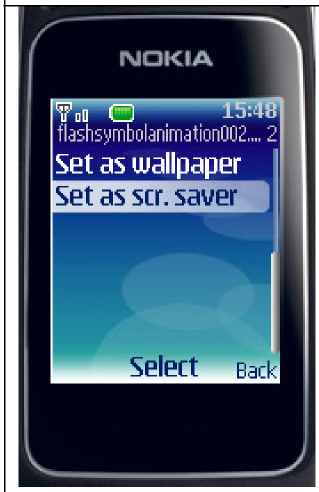


Figure 4: User selects role of Flash Lite content.

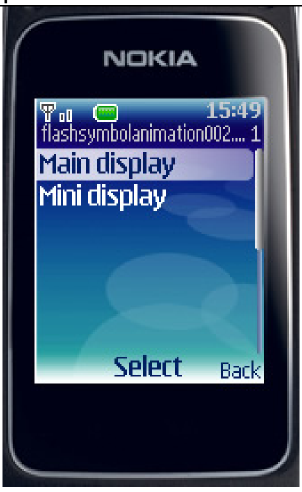


Figure 5: On supporting devices user can select whether content is used for Main display or sub LCD.

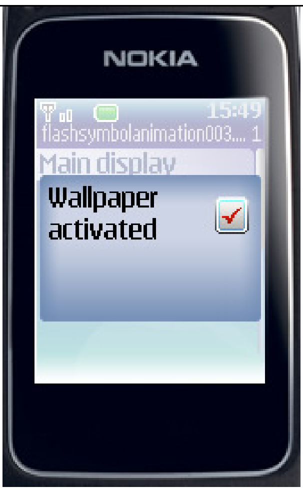


Figure 6: Device informs user that content is activated.

After previewing the content, the user can assign the SWF as a screen saver or wallpaper by pressing the Back key and choosing the content type. A .swf file that displays using the full-screen mode will hide the Back softkey label. This prevents users from easily going to the “Use content now” dialog message box. It is important to avoid using `FullScreen fscommand2` for screen saver and wallpaper content, so that users can see the Back button softkey label.

**Note:** The Series 40 SDK emulates both SWF saving behavior and Flash Lite playback.

## 4.2 S60 devices

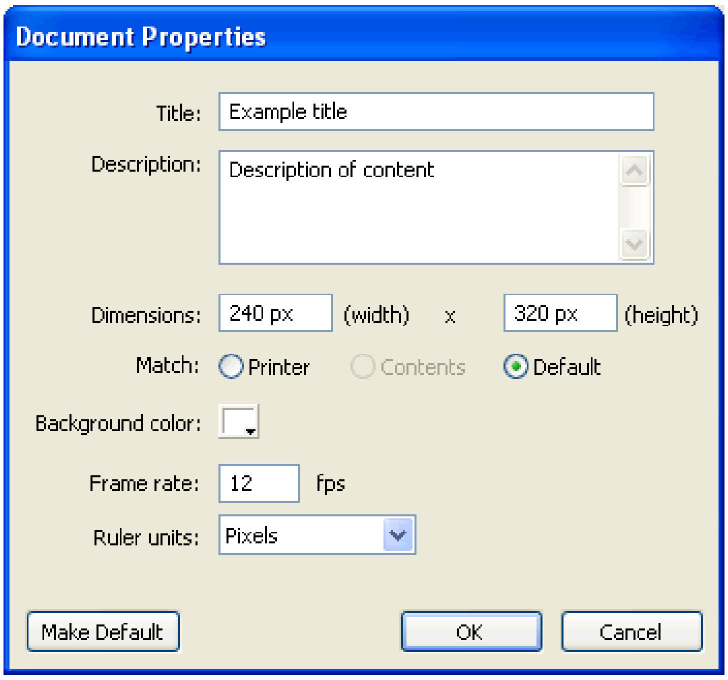
The process for saving SWF on an S60 device is similar to the Series 40 platform. After the SWF downloads, the device will display the SWF in the Flash Lite standalone player so the user may preview the content. Pressing the Back softkey gives the user the option to save the SWF in the Flash folder on the device. To make it easier for the end user to save the content to the device, avoid displaying the SWF in full screen, which will hide the Back softkey label. After saving the SWF file to the device, the user must navigate to **Tools** > **Settings** > **General** > **Personalization** to enable the SWF as a Power Saver animation.

**Note:** S60 3rd Edition, Feature Pack 1 and Feature Pack 2 SDKs do not emulate the SWF saving process and do not support the Flash Lite player or the SWF format.

## 5 Adding metadata to SWF

Developers should add metadata to their Flash Lite content to provide copyright, licensing, terms of use and redistribution, company information, and any other information. One way to add this to SWF files is through the Title and Description fields in the Document Properties window.

To access the Document Properties window, select Document under the Modify menu and enter the appropriate information in the Title and Description fields.


Document Properties

Title: Example title

Description: Description of content

Dimensions: 240 px (width) x 320 px (height)

Match: Printer Contents [✓ Default]

Background color:

Frame rate: 12 fps

Ruler units: Pixels

Figure 7: Document Properties dialog box

Another advantage of including metadata in these fields is that most search engines can index these fields from an SWF file. Consequently, developers can do a search in a popular search engine to keep track of how and where their SWF is being used on the Internet.

## 6 Detecting the device platform with Flash Lite 2.0

In some cases it may be necessary to detect the device platform to execute content differently for Series 40 and S60 devices. Flash Lite supports the `GetPlatform fscommand2` for the purpose of determining the device platform.

The following code example creates a variable named `platform` in the current timeline and assigns the string returned from the `GetPlatform fscommand2` to the variable:

```
// create variable named platform and assign the platform string
fscommand2("GetPlatform","platform");
trace(platform); // outputs "Symbian" for s60 devices
```

Device central returns `"Nokia OS "` (note the extra space) as the value for `GetPlatform` on Series 40 devices and either `"Symbian OS"` or `"Symbian"` plus the version number, e.g., `"Symbian 9.2"` for S60 devices. These values are not the same as those returned by actual devices. Series 40 devices return `"Series 40"` and S60 devices return `"Symbian"` for the `GetPlatform` `fscommand2` function.

**Note:** Flash Lite 1.1 Series 40 devices do not support the `GetPlatform fscommand2`.

The following ActionScript 2.0 code for Flash Lite 2.0 and greater demonstrates how to detect the Nokia platform in both the Device central Flash Lite emulator and on actual devices.

```
// create platform variable
var platform_support:Number = fscommand2("GetPlatform","platform");

if(platform_support > -1){ // device supports platform detection
   if(platform.indexOf("Nokia OS") > - 1  ||
      platform.indexOf("Series 40") > -1){
      // code for Series 40 Flash Lite 2.0+
      trace("Series 40");
   } else if(platform.indexOf("Symbian") > -1){
      // code for S60
      trace("s60");
   } else {
   // non Nokia device
      trace(platform + " - non Nokia device");
   }
} else {
   // device does not support platform detection
   trace("no platform detection");
}
```

Review the example FLA and SWF located in the "detect platform" folder included in the article download files.

## 7 Loading data with Flash Lite 2.0

Flash Lite 2.0 supports loading data in name/value pairs through `loadVariables` or the `LoadVars` class, and XML-formatted data loaded through the `XML` class.

Name/value pairs tend to be more resource efficient to load because Flash Lite automatically parses these into Flash Lite variables. For XML-formatted data, developers must create code to parse XML values into Flash Lite variables; Flash Lite is currently not very efficient at parsing large XML files.

In cases where developers must use XML from a data source that they do not control, they can try to use a proxy server to convert the XML file into name/value pairs, or break the XML file into smaller-sized XML documents that are less resource intensive for Flash Lite to parse.

Another option is to use SWX, an open source system for converting data into Flash native data formats. Loading a data SWF generated by SWX can be more resource efficient and more responsive than loading XML because the data is already converted into a native Flash format. As a matter of coding convenience, the data can also be converted to any Flash Lite 2.0 format such as objects, arrays, or variables. SWX is compatible with Flash Lite 2.0+. Visit http://www.swxformat.org for more information about the SWX project.

## 8 Validating shared objects with Flash Lite 2.0

Because mobile shared objects might not load correctly, it is a good practice to check that data within the shared object exists before attempting to assign the data to a variable. If the stored data does not exist then the variable should be set to a default value instead. Otherwise developers may experience unexpected behavior from their ActionScript or Flash Lite player error messages because their code is expecting a value for the variable that is either not yet defined or has an incorrect value type.

```
// assign default value to a global variable
_global._MYGLOBAL = "default value";

// attempt to load the data and assign it to the global variable
// shared object callback function
function onSharedObjectLoad(so:SharedObject){
   // check that the shared object exists and contains data
   if(so.getSize() != 0){
      // retrieve stored data from shared object
      // check that the data actually exists
      if(so.data.mydata != undefined){
         _global._MYGLOBAL = so.data.mydata;
      }
   }
}
```

## 9 Supporting S60 screen savers for Flash Lite 1.1

Unlike S60 3rd Edition, Feature Pack 1 devices that support both the Flash Lite standalone and screen saver implementations, S60 3rd Edition devices only support the Flash Lite 1.1 standalone player. It is possible to support screen savers for S60 Flash Lite 1.1 devices by developing a custom Symbian C++ application that uses the Screen Saver C++ API for S60 3rd Edition to display an SWF through the Flash Lite 1.1 standalone player when the operating system enters into screen saver mode. Kuneri Lite Kiss60 is an example of a Symbian C++ framework that enables Flash Lite screen savers for any S60 3rd Edition devices.

## 10 Using the [] operator with Flash Lite 1.1

Flash Lite 1.1 supports the `[]` operator, which functions in the same manner as the dot and colon operator for creating or reading variables within a movie clip. For example, the following three statements are equivalent syntax:

```
/*
dot, colon and [] operators can create and read variables in movie clips
*/
mymovieclip:myvariable = "test1";
mymovieclip.myvariable = "test2";
mymovieclip["myvariable"] = "test3";
```

The `[]` operator can also be used to access movie-clip properties.

```
/*
dot and [] operators are equivalent for accessing movie clip properties
*/
mymovieclip._name
mymovieclip["_name"];
```

The `[]` operator, unlike the dot or colon operator, has a third behavior that allows Flash Lite to create a variable with a numeric name.

```
mymovieclip[1] = "test4";
```

Through this technique it is possible to create arrays using standard array syntax that is also compatible with Flash Lite 1.1 ActionScript. The following code example demonstrates how to use the `[]` operator to emulate a Flash Lite 1.1 array that closely follows ActionScript 1.0 syntax.

```
// add a movieclip named "emptymovieclip" to this timeline
// duplicate the emptymovieclip as container for an array named "fruits"
duplicateMovieClip("emptymovieclip","fruits",1);

// create variables within the fruits movieclip using the [] operator
len = 0; // store array length
```

```
fruits[len++] = "apple"; // increment len for each fruit in array
fruits[len++] = "orange";
fruits[len++] = "pear";
fruits[len++] = "grapes";
fruits[len++] = "bannanas";
fruits.length = len;  // assign to a variable in the movieclip

// loop through array and print values
for(i=0; i<fruits.length; i++){
   trace(fruits[i]);
}

// delete the fruits movieclip and variables to clear memory
removeMovieClip("fruits");
```

This code creates a new movie clip by duplicating an empty movie clip to act as a holder for the emulated array variables.  Create a variable, `len`, with a starting value of 0 to represent the length of the array. Use the `[]` operator to create variables in the movie clip, each with a numeric name. For each item in the array, increment the `len` variable to keep track of the length of the array.  The `len` value also assigns the incrementing numeric name of each variable in the movie clip.  After completing the array, assign the `len` variable to the `fruits.length` variable to emulate the length property of an ActionScript 1.0 array.

Now use conventional ActionScript 1.0 code in a `for loop` with an emulated array `length` property and the `[]` operator to access the values of the array.

Review the example FLA and SWF located in the "array operator" folder included in the article download files.

## 11 About the author

Hayden Porter, the author of this document, is a Flash Lite developer with a special interest in developing multimedia content for mobile devices. He has written extensively on the subject of developing mobile content including white papers for leading mobile device manufacturers and articles for publications such as *Electronic Musician Magazine*, *Music Education Technology Magazine*, and DevX.com. For more information, see http://www.aviarts.com.

# 12 Evaluate this resource

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