Archived:Button Component for Flash Lite

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*}}.

Archived because it has been published in pdf format inside the Nokia Developer Flash Lite Components package.

This article explains the use of the Flash Lite Button component and API.

Introduction to the Button component

The Flash Lite Button component is a flexible and scalable user interface component that allows Flash Lite developers to create mobile user interfaces easily, using Flash Lite.

The button component uses placeholder MovieClips, similar to many other Nokia Developer UI components, in order to ease scaling, orientation, and positioning issues.

Figures 1, 2 and 3 give example visual illustrations of the Button component. The exact appearance of the component in your application depends on the parameters and exact style you choose to use.

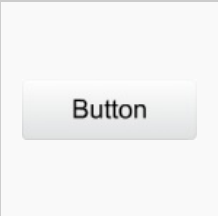


Figure 2. Unpressed button.

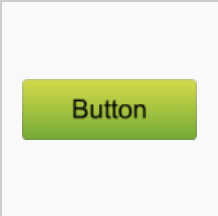


Figure 3. Pressed button.

Requirements

- Adobe Flash Professional CS3 or CS4
- Flash Lite 2.0 Player and above

Download

Button component is included in the Nokia Developer Flash Lite Components package.

Installation

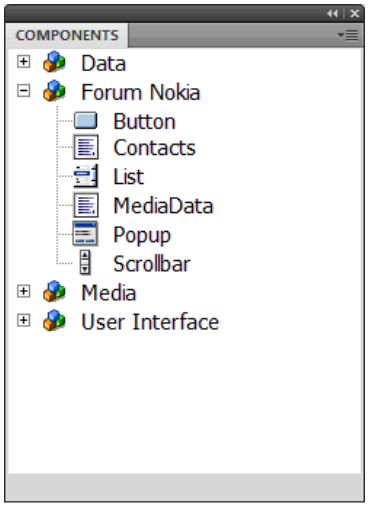
Installation of the Button component is easy. Execute the components MXP file and follow the simple instructions in Adobe Extension Manager to complete the installation process. Restart Adobe Flash CS4 after installation.

Note: Component FLA and AS files should appear in the directories *[INSTALL PATH]\Flash CS4\en\Configuration\Components\Nokia Developer* and *[INSTALL PATH]\Flash CS4\en\Configuration\Classes\com\forumnokia*. In some setup environments, the files may appear under the wrong language directories (for example, *fi* instead of *en*). In such cases, the files should be manually moved to the correct directories as specified above.

Preparations

1. Create a Flash Lite Project. Open the Component panel (Ctrl/Apple key + F7) and drag the Button component onto the stage. Assign a unique instance name for the object using the properties panel (Ctrl/Apple key + F3).

(C) Copyright Nokia 2013. All rights reserved.



2. Nokia Developer Flash Lite UI components require the Flash Lite movie to be set to align to the top left corner of the device screen and not to scale. Some ActionScript code is needed to achieve this. The following code must be added to the first frame of the project:

Invalid language.

You need to specify a language like this: `<source lang="html4strict">...</source>`

Supported languages for syntax highlighting:

4cs, 6502acme, 6502kickass, 6502tasm, 68000devpac, abap, actionscript, actionscript3, ada, algol68, apache, applescript, apt_sources, asm, asp, autoconf, autohotkey, autoit, avisynth, awk, bascomavr, bash, basic4gl, bf, bibtex, blitzbasic, bnf, boo, c, c_loadrunner, c_mac, caddcl, cadlisp, cfdg, cfm, chaiscript, cil, clojure, cmake, cobol, coffeescript, cpp, cpp-qt, csharp, css, cuesheet, d, dcs, delphi, diff, div, dos, dot, e, ecmascript, eiffel, email, epc, erlang, euphoria, f1, falcon, fo, fortran, freebasic, fsharp, gambas, gdb, genero, genie, gettext, glsl, gml, gnuplot, go, groovy, gwbasic, haskell, hicest, hq9plus, html4strict, html5, icon, idl, ini, inno, intercal, io, j, java, java5, javascript, jquery, kixtart, klonec, klonecpp, latex, lb, lisp, llvm, locobasic, logtalk, lolcode, lotusformulas, lotusscript, lscript, lsl2, lua, m68k, magiksf, make, mapbasic, matlab, mirc, mmix, modula2, modula3, mpasm, mxml, mysql, newlisp, nsis, oberon2, objc, objeck, ocaml, ocaml-brief, oobas, oracle11, oracle8, oxygene, oz, pascal, pcre, per, perl, perl6, pf, php, php-brief, pic16, pike, pixelbender, pli, plsql, postgresql, povray, powerbuilder, powershell, proftpd, progress, prolog, properties, providex, purebasic, pycon, python, q, qbasic, rails, rebol, reg, robots, rpmspec, rsplus, ruby, sas, scala, scheme, scilab, sdlbasic, smalltalk, smarty, sql, systemverilog, tcl, teraterm, text, thinbasic, tsql, typoscript, unicon, uscript, vala, vb, vbnet, verilog, vhdl, vim, visualfoxpro, visualprolog, whitespace, whois, winbatch, xbasic, xml, xorg_conf, xpp, yaml, z80, zxbasic

```
fscommand2("DisableKeypadCompatibilityMode");
fscommand2("FullScreen", true);
fscommand2("SetQuality", "high");
Stage.scaleMode = "noScale";
Stage.align = "TL";
_focusrect = false;
```

3. In order to catch the button events, you need to add a listener to the button's onRelease event:

Invalid language.

You need to specify a language like this: `<source lang="html4strict">...</source>`

Supported languages for syntax highlighting:

4cs, 6502acme, 6502kickass, 6502tasm, 68000devpac, abap, actionscript, actionscript3, ada, algol68, apache, applescript, apt_sources, asm, asp, autoconf, autohotkey, autoit, avisynth, awk, bascomavr, bash, basic4gl, bf, bibtex, blitzbasic, bnf, boo, c, c_loadrunner, c_mac, caddcl, cadlisp, cfdg, cfm, chaiscript, cil, clojure, cmake, cobol, coffeescript, cpp, cpp-qt, csharp, css, cuesheet, d, dcs, delphi, diff, div, dos, dot, e, ecmascript, eiffel, email, epc, erlang, euphoria, f1, falcon, fo, fortran, freebasic, fsharp, gambas, gdb, genero, genie, gettext, glsl, gml, gnuplot, go, groovy, gwbasic, haskell, hicest, hq9plus, html4strict, html5, icon, idl, ini, inno, intercal, io, j, java, java5, javascript, jquery, kixtart, klonec, klonecpp, latex, lb, lisp, llvm, locobasic, logtalk, lolcode, lotusformulas, lotusscript, lscript, lsl2, lua, m68k, magiksf, make, mapbasic, matlab, mirc, mmix, modula2, modula3, mpasm, mxml, mysql, newlisp, nsis, oberon2, objc, objeck, ocaml, ocaml-brief, oobas, oracle11, oracle8, oxygene, oz, pascal, pcre, per, perl, perl6, pf, php, php-brief, pic16, pike, pixelbender, pli, plsql, postgresql, povray, powerbuilder, powershell, proftpd, progress, prolog, properties, providex, purebasic, pycon, python, q, qbasic, rails, rebol, reg, robots, rpmspec, rsplus, ruby, sas, scala, scheme, scilab, sdlbasic, smalltalk, smarty, sql, systemverilog, tcl, teraterm, text, thinbasic, tsql, typoscript, unicon, uscript, vala, vb, vbnet, verilog, vhdl, vim, visualfoxpro, visualprolog, whitespace, whois, winbatch, xbasic, xml, xorg_conf, xpp, yaml, z80, zxbasic

```
var myEvent = myButton.onReleaseEvent();
myButton.addEventListener(myEvent, myListener);
function myListener(eventObject:Object):Void {
   if (eventObject.type == myEvent) {
      switch (eventObject.target) {
         case _level0.myButton :
            // My Button action here
            break;
         default :
            break;
      }
   }
```


(C) Copyright Nokia 2013. All rights reserved.

```
}
```

Structure

Assets

Button assets consist of an icon (50x50 pixels) and a label (50x100 pixels). Each asset has separate MovieClips for Up, Down, and Over states of the button, which are organised in the following structure:

```
_assets
    Default
        Up
            Icon
            Label
        Down
            Icon
            Label
        Over
            Icon
            Label
```

NOTE: Custom assets must follow the same folder structure, and linkage identifiers must be named accordingly.

Skins

The skin of a button component is composed of nine different parts, as can be seen below. This structure enables the button interface to be scalable and easily skinnable.

As with assets, skins also have separate MovieClips for Up, Down, and Over states of the button. The structure and default size of these parts is defined below:

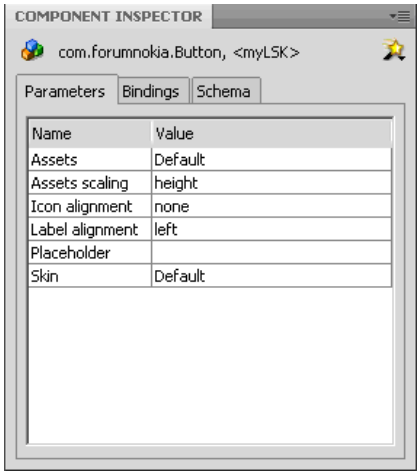
```
_skins
    Default
        Up
            top_left (10x10)
            top (155x10)
            top_right (10x10)
            left (10x40)
            body (155x40)
            right (10x40)
            bottom_left (10x10)
            bottom (155x10)
            bottom_right (10x10)
        Down
            top_left
            top
            top_right
            left
            body
            right
            bottom_left
            bottom
            bottom_right
        Over
            top_left
            top
            top_right
            left
            body
            right
            bottom_left
            bottom
            bottom_right
```

Note: Each new custom skin must follow the same folder structure, with linkage identifiers named accordingly.

Inspectable parameters

Inspectable parameters help customise the Button component from the Component Inspector panel (Shift + F7). All inspectable parameters can also be controlled via ActionScript with the component APIs.

(C) Copyright Nokia 2013. All rights reserved.



Parameter	Description	Value
Skin	Defines the button skin linkage identifier path. For instance, for the *mySkin* value, the button would search for the skin components under the *Button.mySkin*.* linkage path.	Skin identifier string
Assets	Defines the button asset linkage identifier path. For instance, for the *myAssets* value, the component would search for the assets under the *Button.myAssets*.* linkage path.	Asset identifier string
Placeholder	If specified, the button component will locate and scale itself according to this parameter. If not specified, the button locates itself manually and is drawn based on its default dimensions.	Instance of a placeholder MovieClip
Assets scaling	Defines the scaling method for the component assets. If specified, component assets are scaled according to the given width or height of the button.	None/width/height
Icon alignment	Defines the location of the icon asset on the button, if an icon is defined.	None/left/center/right
Label alignment	Defines the text alignment of the label asset.	Left/center/right

---

ButtonComponent ActionScript API

Setting the skin path

public function setSkin(path:String): Void;

Sets a new path for the skin. Reconstructs the component.

Setting the asset path

public function setAssets(path:String): Void;

Sets a new path for the assets. Reconstructs the component.

Defining a placeholder

public function setPlaceholder(nameStr:String): Void;

Sets a placeholder object used to relocate, scale, and reconstruct the component.

Setting the buttons label text

public function setLabel(newLabel:String): Void;

Sets the label text value of the component.

Setting the asset scaling method

public function assetScaling(newProperty:String): Void;

Defines the asset scaling method. The parameter value can be either “none”, “width”, or “height”.

Setting icon aligment

public function setIconAlignment(newPosition:String): Void;

Defines the location of the icon asset according to the parameter, when an icon is defined. The parameter value can be “none”, “left”, “center”, or “right”. Label location changes according to icon alignment.

Setting label alignment

public function setLabelAlignment(newPosition:String): Void;

(C) Copyright Nokia 2013. All rights reserved.

Sets the label text alignment. The parameter value can be “left”, “center”, or “right”.

Activating the component

public function enableComponent() :Void;

Makes the component react to user input.

Disabling the component

public function disableComponent() :Void;

Makes the component stop reacting to user input.

Adding an event listener

public function addEventListener(event:String, listenerFunction:Object): Void;

Specifies a listening function that receives the specified event.

Removing an event listener

public function removeEventListener(event:String, listenerFunction:Object): Void;

Tells the component to stop dispatching events to the listener function.

onRelease event

public function onReleaseEvent():String;

Calls the button component’s release event.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.