KDP skinning

# Overview

KDP uses a number of standard flash components as well as custom components.

This document explains the basics of skinning a KDP and the use of the Skin Project.

# Skin project

This project allows you to edit KDP skin and see the results immediately.

The skin.fla file is located under the assets folder. Edit the desired parts of this file.

After publishing skin.fla to generate skin.swf you can see KDP with your newly created skin – just open the file kdp3.html in a web browser and you should see the player wearing the new skin.

Please note that no matter what type of player you chose to edit (single player, playlist, multiplaylist etc.) The skin.fla file contains all assets for all types of players. This should make it easy to create similar skins for all kinds of players.

The config and HTML files are different for the different players, though, so you don't want to override these.

Copy the config and HTML files from one of the sub-directories to the main one to change the player type. We recommend you keep the original config files intact so you will always have a clear copy to start with.

# KDP skinning

## Button skinning

### Glossary

**Skin assets:** the assets that build the background of the button.

There are up to 8 states: up, over, down, disabled, selected up, selected over, selected down, selected disabled. Some buttons use the selected states (play/pause are actually the same button where the selected state is the pause state and the unselected is the play state. The same thing goes for full screen button).

**Icons assets:** the icons for the button.

As the skin – the icons can appear as states icons, or one icon for all states. a button can get icon and/or label.

**label:** the text field that appears on the button (right/top/left/bottom according to the labelPlacement attribute).

### Using the default button style

In this case you do nothing in terms of style attribute. A button without styleName or skin / styles attribute will get the default button skin as it is set in the skin.swf. This default skin can have all 8 states, and a label, and by definition – does not have icons.

### Applying a styleName

Setting a styleName to a button is as simple as adding the attribute to the relevant Button-tag in the config file:

<Button … styleName=”myStyle” />

In this case the KDP tries to find matching assets with the same postfix. It will try to set these assets as skin for the given button:

• Button\_upSkin\_myStyle

• Button\_downSkin\_myStyle

• Button\_overSkin\_myStyle

• Button\_disabledSkin\_myStyle

• Button\_selectedUpSkin\_myStyle

• Button\_selectedOverSkin\_myStyle

• Button\_selectedDownSkin\_myStyle

• Button\_selectedDisabledSkin\_myStyle

These should be on the skin.swf as assets on stage with matching class definitions. The symbol's name in the library is irrelevant, but as best practice we recommend to give it the same name as the class definition. If there are no such assets, the button will get a fallback skin that should be at every skin.swf with the same structure but with the “default” postfix:

• Button\_upSkin\_default

• Button\_downSkin\_default

• Button\_overSkin\_default

• Button\_disabledSkin\_default

• Button\_selectedUpSkin\_default

• Button\_selectedOverSkin\_default

• Button\_selectedDownSkin\_default

• Button\_selectedDisabledSkin\_default

KDP will try to set these assets as matching icons:

• Button\_upIcon\_myStyle

• Button\_overIcon\_myStyle

• Button\_downIcon\_myStyle

• Button\_disabledIcon\_myStyle

• Button\_selectedUpIcon\_myStyle

• Button\_selectedOverIcon\_myStyle

• Button\_selectedDownIcon\_myStyle

• Button\_selectedDisabledIcon\_myStyle

If there are no matching icons the button will not have a default icon. In this case there should be a label on the button.

KDP will try to set a text format copied from a MovieClip containing a single TextField. KDP will copy all of the text field's attributes (size, color, bold, italic, spacing, margins, justify etc’) and set it to the label of the given button. To do so the MovieClip which holds that TextField must have the following class definition: Button\_Label\_myStyle

Handling fonts will be elaborated next.

### Applying assets directly

We can use the advantage of override to reuse assets without duplicating them in the skin file. For example if we have 3 buttons with the same skin but with different icons we can use the styleName to set the skin, and override the icons by using different icons attributes in the config file:

<Button … styleName=”myStyle”

upIcon="someIcon" overIcon="someIcon" downIcon="someIcon" disabledIcon="someIcon"

selectedUpIcon="someIcon" selectedOverIcon="someIcon" selectedDownIcon="someIcon" selectedDisabledIcon="someIcon" />

This example shows the use of one icon for all states, but we may also use different icons for the different states.

NOTE: an icon's class name must contain the string "icon" in order for it to accept color definitions.

It is also possible to override specific skin parts by specifying the matching attribute names:

<Button … upSkin="Button\_upSkin\_myStyle" downSkin="Button\_downSkin\_myStyle"

overSkin="Button\_overSkin\_myStyle" disabledSkin="Button\_disabledSkin\_myStyle"

selectedUpSkin=" Button\_selectedUpSkin\_myStyle"

selectedOverSkin="Button\_selectedOverSkin\_default"

selectedDownSkin="Button\_selectedDownSkin\_default"

selectedDisabledSkin="Button\_selectedDisabledSkin\_default" />

NOTE: a skin asset's class name must contain the string "skin" in order for it to behave properly.

### Type (font)

Setting button dimensions

The button of KDP is actually an autosize button. This means that if the button has a label it will set its width automatically. This means that in some cases the width/height we want and the actual width/height are not equal. We have 4 attributes which help us get more accurate dimensions. The attributes are minWidth, maxWidth, minHeight, maxHeight. The following tag defines a button whose height will be 22 pixels height at most and will have a width of 40 pix even if its label is shorter or icon is smaller.

<Button … minWidth=”40” maxHeight=”22” />

### Commands & behavior

There are two internal buttons commands – play and fullscreen. Both buttons are toggle buttons, meaning that when clicked they change their state to selected. This said, when you design the skin & icons place the close full screen / pause to the selected state, and open fullscreen / play to the non selected. These 2 buttons have the code behind to command the KDP what to do in each state and click.

In addition, there are 3 general click behaviors:

• jsCall – call a JS function with or without arguments:

e.g. kClick=” jsCall('myJsFunc', mediaProxy.entry.id )"

This will call a js function called ‘myJsFunc’ and will pass it the current entry id once the button is clicked.

• navigate to an external url:

e.g. kClick="navigate('http://www.kaltura.com')"

• send notification to the KDP. The kdp has internal notification, like the events mechanism, and a button can send a notification with or without data.

e.g. kClick="sendNotification('changeVolume',1)”

## Containers skinning

A container can get a rectangle skin asset. There is no default skin for containers.

To apply a skin asset to a container use the styleName attribute on the relevant tag in the config file.

The following tag defines a container with an asset named “gradientBlack”:

<VBox id=”controllerContainer” width=”100%” styleName=”gradientBlack” … />

The asset will be stretched or shrunk to match the actual size of the container. This asset can also use flash's 9-scale attribute.

## Scrubber skinning

The scrubber has a predefined structure that has to be used. Basically it has 3 assets for the scrubber, and 4 thumbnail (the scrubber handler) states. You don’t need to set the scrubber styleName attribute in the config file; you have to directly edit its assets.

* scrubber background: Scrubber\_track\_default
* for the track highlight – the part over the background that marks the part already played – to the left of the thumbnail: Scrubber\_progress\_default
* for the buffering part – the part that indicates the media's load progress: Scrubber\_buffer\_default

The asset names for the scrubber thumbnail states are:

* Scrubber\_thumbUp
* Scrubber\_thumbOver
* Scrubber\_thumbDown
* Scrubber\_thumbDisabled

In case you don’t need a visual thumbnail, use a 0.01 transparent asset, so the user will be able to drag.

## Timer skinning

Timers are actually labels. You need to add a styleName attribute to the Timer tag, and in the skin file use the same value as the asset's class definition. Just a reminder – the asset should contain one (and only one) textfield with the design attributes (size, color, text align, padding etc’).

The following tag defines a timer whose appearance matches a MovieClip with the class definition "timerProgress":

<Timer.. styleName="timerProgress" format="mm:ss" />