

How to create a skin for JMediaPlayer

JMediaPlayer skin documentation.

Based on JMediaPlayer 1.60. Future versions of JMediaPlayer will extend the current skin format maintaining full compatibility for skins based on this and older specifications.

Introduction

A JMediaPlayer skin is made of several images of one or more formats packed in a zip archive along with a file that defines the image name, the location, the size, the action and/or additional properties of every functional element of the user interface. This file must be called `main.xml` and its structure is described in the next sections of this document.

Although the format of the images can be one of the following

- .png
- .bmp
- .gif
- .jpg or .jpeg
- .tif or .tiff

You will probably want to use a lossless format such as .png or .bmp to avoid ruining your work with bad compression.

The official skin file extension is `.jmpskin`, but JMediaPlayer will recognise as skins also `.zip` files. This is useful for testing purposes; before distributing your skin please rename it to `SkinName.jmpskin`

main.xml

`main.xml` is a standard extensible markup language (XML) document. It is derived from the schema available at <http://www.jockersoft.com/jenius/jmpskin.xsd> (see [Troubleshooting](#) section for more details and download links).

Every control (button, label, trackbar, other) can be represented as an `element` on the xml document. The properties of the control are `attributes` of the `element`.

Elements starts with `<` followed by the control name, its `attributes&values` and end with `>`. They can span in multiple lines

The `attribute` name is followed by `=` and its `value` is enclosed by double quotes `" "`

```
<controlname attribute1="value1" attribute2="value2" attribute3="value3" />
```

For example, valid controls declarations are:

```
<label id="lblStatus" color="17; 68; 166" font="Trebuchet; 8pt; style=Bold" x="185" y="32" />
<imgbutton id="Pause" file="bpause.bmp" x="500" y="98" />
<trackbar id="barPosition" file="seekb.bmp" x="120" y="65" w="226" h="5"
    backCl="111; 119; 111" borderCl="White" fillCl="255; 0; 255"
    horizontal="true" mode="unhide" />
```

In the section ['Skin Elements'](#) you will find all you need to know to define controls in your skin and in section ['Putting it together'](#) you will find more details in skin structure.

Transparency

JMediaPlayer *supports alpha transparency* in the background image and supports a single transparent color in the images of the various controls. This means that the buttons/bars can have non-rectangular shapes and that you can create elaborated skins with a semi transparent background that let partially show what's behind it!!

To have transparency in your skin, you have two options:

- use an image format that supports it: these are png (32bit images with Alpha, Red, Green, Blue channels or 24bit with 1 transparent color) or Gif (8bit palette based images with only one transparent color and 256 colors max).
- if for any weird reason you hate png images, you can choose one color to be used as transparent. You'll have to set its name or RGB value to the `transparentColor` attribute of the `background` element. You will find an explanation on how to do this in the next section.

Transparency issues:

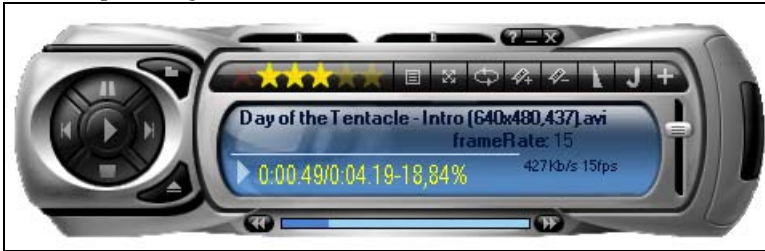
Only if you chose the 32bit png semi-transparency approach there are a few things to keep in mind:

- you have to tell JMediaPlayer that your skin uses alpha transparency in the main background image, setting to `true` the `alphaTransparency` attribute in the `background` element. You will find an explanation on how to do this in the next section.
- alpha transparency is supported only on Windows 2000, Windows XP or higher operating systems; on older systems JMediaPlayer will automatically set to opaque every pixel of your main skin background that has partial transparency.

- every control must be and be placed over opaque regions. This is especially true for labels and trackbars with transparent background and imagebuttons with non-rectangular shapes. For technical reasons, these controls must be drawn over a partial copy of the main background.

You'll need to create another .png image (`maskImage` attribute) of the same size of the main background image that will be completely transparent except of the areas under these controls that must be completely opaque.

For example, using this skin:



The mask image will be something like this:



Failing to provide a mask image will result in:



Automatic conversion tool from BSplayer's skins

If you have already created a skin for BSplayer (version 1.0) and you want to port it to JMediaPlayer, you can use the import tool integrated in JeniuS (**Tools** menu, **Settings**, **JMediaPlayer** -> **Skin** tab). This will make a main.xml file and will pack it with every image of the old skin in a .jmpskin file (a .zip file, renamed) in the default JeniuS skin directory.

The imported skin will probably need some tweaks in labels size and position. There may be some BSplayer functions your skin use that JMediaPlayer currently doesn't support: you may wish to replace them with one or more JMediaPlayer specific functions.

See appendix '[skin conversion from BSplayer](#)' for the complete list and a conversion table and more details on BSplayer skin conversion.

Skin Elements

Skin Background

The `background` element has the following attributes:

- `mainImage` : file name of the background image of your skin.
- `alphaTransparency` : if set to `true`, JMediaPlayer will render the background image using alpha transparency. If the `mainImage` is not a 32bit png file, an error is shown.
Optional. Default is `false`
- `maskImage` : file name of the mask image.
Optional. Must be present if `alphaTransparency` is set to `true`; if `alphaTransparency` is set to `false` it is ignored.
- `transparentColor` : in non 32bit png background images, use this attribute to choose a color to be treated as transparent.
Optional. Default is `Transparent`.

See [Valid color values](#) section for allowed values and examples

The general format of the background element is this (in square brackets [] are enclosed optional attributes. see above for default values):

```
<background mainImage="FILE" [alphaTransparency="true|false"] [maskImage="FILE"]
           [transparentColor="COLOR"] />
```

For example, 2 valid background declarations may be:

```
<background mainImage="main.png" alphaTransparency="true" maskImage="main_Mask.png" />
```

```
<background mainImage="mybackground.png" transparentColor="Magenta" />
```

Image Buttons



An image button (`imgbtn` element) is a special button that can assume one of the following 5 states depending on several factors:

- Normal state
- Mouse Up
- Mouse Down (click)
- Disabled state
- Checked state

Only the first 3 (normal, mouseup, mousedown) must exist for every button. The checked state is used only by 3 buttons and the disabled state is currently unused.

So, for each button you want to insert, you'll need to provide 3 images of the same format and size. The file name of each image must follow this convention:

`filenameSTATECODE.extension`

where:

`filename` is common to every image of that button and can be whatever you want (IE: `btnplay`)

`STATECODE` is a code that distinguishes one of the 5 states: normal= `n`; mouseup= `u`; mousedown= `d`; disabled= `dis`; checked= `c`

`extension` is common to every image of that button and is one of the allowed images extension (IE: `png`)

For example, the image for the Play button in it's normal state can be: `btnplayn.png`

continuing with the example, the mouseup state will be `btnplayu.png`, the mousedown state will be `btnplayd.png`, the disabled state will be `btnplaydis.png` and the checked state will be `btnplayc.png`.

Filenames and state codes are not case sensitive.

Button functions

The function of an image button is determined by it's `id` (identifier) attribute. The complete list of valid `ids` currently supported is the following (in **bold** are represented the buttons that should always be in every skin, but you are not forced to insert them):

```
Play
Pause
Stop
PreviousTrack
NextTrack

VolumeUp
VolumeDown
Mute

LoadJenius
AddFileToJenius
Volume

Fullscreen
ShowPlaylist

SubtitlesOnOff
SubFontPlus
SubFontMinus
MoveSubtitlesUp
MoveSubtitlesDown

Settings
Skins

FastForwards
FastBackwards
SeekForwards
SeekBackwards
SeekForwardsLarge
SeekBackwardsLarge
JumpToTime

Zoom50
Zoom100
Zoom200
AspectCycle
AspectOriginal
Aspect169
```

```

Aspect43
VideoInfo
PanIn
PanOut
ZoomIn
ZoomOut
CustomPanScan
CustomPosition

Minimize
Exit
Credits
VideoAlwaysOnTop

Open
OpenAudio
LoadSubtitles
CloseFile

AddBookmark
EditBookmark

IncrementPlaybackSpeed
DecrementPlaybackSpeed
IncrementPlaybackSpeedSmall
DecrementPlaybackSpeedSmall

Random
SwitchStatusDisplayMode

```

Most of them are self explanatory and can be found also in other programs, but for the other ones here is the description:

LoadJenius: opens the main JeniuS interface. This buttons must also have the disabled state.

AddFileToJenius: adds the current file playing to JeniuS' library

Volume: opens an external mixer/equalizer set by the user. The default is Windows' mixer.

SeekForwards and **SeekBackwards**: when pressed they jump to the next/previous <n> seconds of the file

SeekForwardsLarge and **SeekBackwardsLarge**: as **SeekForwards** and **SeekBackwards**, but the jump is of more seconds

FastForwards and **FastBackwards**: while kept pressed, they act as **SeekForwards** and **SeekBackwards**

CustomPosition: for video files, opens a window that let the user select where the video coordinates in the draw surface

CustomPanScan: for video files, opens a window that let the user select the scale factor

Duplicated items

You can insert in the skin more than one button with the same name (same function), but only the first one will be used for special actions (like the SubtitlesOnOff button, when a subtitles file is loaded it is displayed in its checked state)

Syntax in main.xml

To insert a button in your skin, add the following line in the body of main.xml:

```
<imgbutton id="OneOfTheValidIDs" file="FILENAME.EXTENSION" x="INT" y="INT" />
```

Note:

- FILENAME is the base filename of the image, without the statecode
- INT is an integer number > 0
- you must specify the x,y location of the button and you cannot specify it's width or height: it's size will be taken from the normal state image.

For example, 3 valid button declarations may be:

```

<imgbutton id="Play" file="bplay.png" x="45" y="53" />
<imgbutton id="Pause" file="bpause.bmp" x="36" y="30" />
<imgbutton id="Minimize" file="Btn2.png" x="316" y="2" />

```

Track Bars



A track bar (`trackbar` element) is a bit more complex than a button. I hope you have an idea of what a track bar is because I'm not able to find a good definition that isn't in contrast with what JMediaPlayer's track bars can be :-)

Here are the main characteristics:

- **shape**: rectangular. It is defined using 4 attributes: **x**, **y**, **w** (width), **h** (height)
- **orientation**: defined using the **horizontal** attribute.
Valid values are **true** or **false**.
See [Valid boolean values](#) section for allowed values
- **colors**: are optional and can be omitted (in this case it is used the **Transparent** color). Just remember to use an image (see later).
Color attributes are **backCl** (background color), **borderCl** (border color. border width = 1pixel), **fillCl** (the area that represents the value will be paint using this color)
See [Valid color values](#) section for allowed values and examples
- **image**: the image to be used as background or as 'handle'. See track bar type.
The image is defined using the **file** attribute and is optional if you specified the **backCl** and **fillCl** color attributes.
- **track bar type**: the attribute **mode** can assume these values
 - **centered**: the image will be drawn as an 'handle' that can be dragged to change the value of the track bar.
 - **unhide**: the image will be partially drawn in the background. The higher the value of the trackbar is, the more the image is shown.
Must be set if an image has been provided. If no image is set, it is ignored.
- **show center**: if the **center** attribute is set to **true**, then in the center of the trackbar will be drawn a small line of the same color of **borderCl**
Optional. Default is **false**.
See [Valid boolean values](#) section for allowed values
- **function**: is determined by the **id** attribute.
The complete list of valid **ids** currently supported is:
 - **barBalance**: balancement of left/right audio channels
 - **barSpeed**: playback speed of the media
 - **barPosition**: current position compared with media length
 - **barVolume**: adjust audio volume

Duplicated items

No duplicated items are allowed

Syntax in main.xml

The general format of a track bar is this (in square brackets [] are enclosed optional attributes. see above for default values):

```
<trackbar id="OneOfTheValidIDs" [file="FILE"] x="INT" y="INT" w="INT" h="INT"
        [backCl="COLOR"] [borderCl="COLOR"] [fillCl="COLOR"]
        horizontal="true|false" [center="true|false"] [mode="centered|unhide"] />
```

For example, 2 valid button declarations may be:

```
<trackbar id="barBalance" file="smallhandle.png" x="148" y="5" w="58" h="7"
        backCl="Transparent" borderCl="Transparent" fillCl="Transparent"
        horizontal="true" center="true" mode="centered" />

<trackbar id="barPosition" x="166" y="121" w="156" h="7"
        backCl="154; 210; 246" borderCl="Black" fillCl="74; 130; 214"
        horizontal="true" mode="unhide" />
```

Labels

Jenius supports two kinds of labels (**label** element): 'normal' labels and active labels. The first one only displays text, while the last one is also sensitive to mouse clicks. (active label in normal state: **duration**:4.32 ; active label that can be clicked: **artist**: Green Day)

The 'normal' labels can have interesting properties:

- its font size can be stretched using **vertiStretch** and **horizStretch** attributes (IE: **0:00.03/0:04.19-1,26%** is stretched vertically by 1,32)
- it can scroll it's text if the **scrollable** attribute is set to **true**

Label attributes

- **id**: a label can have one of these ids:
 - **lblStatus**: typical texts this label has are: "Welcome to JMediaPlayer", "1:23:45 / 2:34:56 54%"
 - **lblProperties**: this should be an 'active label'. Every few seconds it will show different properties of the media file. Typical texts are: "album: AlbumName", "Played: <n>", "Genre: genre"...
 - **lblFile**: this should be a scrollable label. It shows the filename that is currently playing and the meaning of the various buttons in the interface (when mouse is over them)
 - **lblAuthor**: this only shows author and album properties, if any
 - **lblBitrate**: this shows technical info about the file. Typical texts are: "128Kb/s 44100Hz", "800Kb/s 25fps", ...
- **color**: the font color.
See [Valid color values](#) section for allowed values and examples
- **boldColor**: used only for active labels. It is the color of the font used to display the property name that is being shown.
Optional

See [Valid color values](#) section for allowed values and examples

- **font** : the font of the label.

See [Valid font values](#) section for allowed values and examples

- **horizStretch** : Optional. If set and if their value is different from **1**, the label will be drawn on the horizontal axis multiplying the font size by the value of this attribute. Can be any positive decimal number (greater than **0**). Default: **1**
- **vertiStretch** : as **horizStretch** but the affected axis is the Vertical one
- **scrollable** : If set to **true**, the text of the label will keep scrolling if it is too large to be fully displayed in the label size. Optional. Default is **false**. See [Valid boolean values](#) section for allowed values
- **x, y, w, h** : as before

Duplicated items

No duplicated items are allowed

Syntax in main.xml

The general format of a label is this (in square brackets [] are enclosed optional attributes. see above for default values):

```
<label id="OneOfTheValidIDs" color="COLOR" [boldColor="COLOR"] font="FONT"
      [horizStretch="FLOAT"] [vertiStretch="FLOAT"] [scrollable="true|false"]
      x="INT" y="INT" w="INT" h="INT" />
```

For example, 3 valid label declarations may be:

```
<label id="lblStatus" color="Yellow" font="Microsoft Sans Serif; 9pt"
      horizStretch="1" vertiStretch="1,32"
      x="150" y="84" w="164" h="20" />
<label id="lblProperties" color="2; 19; 54" boldColor="2; 19; 54"
      font="Microsoft Sans Serif; 8,25pt" x="271" y="65" w="132" h="15" />
<label id="lblFile" color="2; 19; 54" font="Microsoft Sans Serif; 8pt; style=Bold"
      scrollable="true" x="138" y="52" w="258" h="14" />
```

Rating box:

A rating box ([rating](#) element) is a special control that lets the user rate the file JMediaPlayer is currently playing.

The rate can be one of the following: corrupt (a red cross) or good (1, 2, 3, 4 or 5 stars).

If you want to use custom cross and star images, you need to set the **fileOK** and **fileBad** attributes and provide two copies of every image: one for the normal state, the other for the mouseup state. This follows the same naming convention as with image buttons ('N' for normal state, 'U' for mouse up state).

Rating box attributes:

- **id** : it's value must always be **rating**
- **x, y, w, h** : as before
- **fileOK** : the base name + extension of the file image that represent a good rating. Only one element is necessary: JMediaPlayer will repeat it for you. Generally this is a bright star.
Optional if **fileBad** is not present.
- **fileBad** : the base name + extension of the file image that represent a bad rating. Generally this is a red cross.
Optional if **fileOK** is not present.

Duplicated items

No duplicated items are allowed

Syntax in main.xml

The general format of a label is this (in square brackets [] are enclosed optional attributes. see above for default values):

```
<rating id="rating" x="INT" y="INT" w="INT" h="INT"
      fileOK="FILENAME.EXTENSION" fileBad="FILENAME.EXTENSION" />
```

Where FILENAME is the base filename of the image, without the statecode.

For example, a valid rating box declarations may be:

```
<rating id="rating" x="135" y="24" w="96" h="16" fileOK="star.png" fileBad="cross.png" />
```

Putting it together

Now that you know which controls you can put in your skin, let's see how to create a skin template.

Open your favourite text-only editor (notepad is OK, Microsoft Word isn't), paste the following lines:

```
<Skin xmlns="http://www.jockersoft.com/schemas/jmpskin.xsd">
</Skin>
```

and save them in a file called `main.xml`

Now insert a few info about the skin and yourself, if you wish: insert it inside the blank space between the first and the last line.

```
<!-- Skin details -->
<skinInfo>
  <name>Skin name</name>
  <description>Skin description</description>
  <author>Your nick</author>
  <authorEmail>Your email</authorEmail>
  <authorUrl>Your website url</authorUrl>
</skinInfo>
```

If you don't want to fill-in a field, simply delete that line.

`main.xml` now will look something like

```
<Skin xmlns="http://www.jockersoft.com/schemas/jmpskin.xsd">

<!-- Skin details -->
<skinInfo>
  <name>Skin name</name>
  <description>Skin description</description>
  <author>Your nick</author>
  <authorEmail>Your email</authorEmail>
  <authorUrl>Your website url</authorUrl>
</skinInfo>
</Skin>
```

Insert the background element, as described in the first part of this document. Example:

```
<!-- Main settings -->
<background mainImage="main.png" alphaTransparency="true"
  maskImage="main_Mask.png" transparentColor="Magenta" />
```

Insert the various image buttons, trackbars and labels as described previously

```
<!-- Image buttons -->
<imgbutton id="Play" file="bplay.png" x="45" y="53" />
<imgbutton id="Pause" file="bpause.png" x="36" y="30" />

<!-- Track bars -->
<trackbar id="barPosition" x="166" y="121" w="156" h="7" backCl="154; 210; 246"
  borderCl="Black" fillCl="74; 130; 214" horizontal="true" mode="unhide" />

<!-- Labels -->
<label id="lblStatus" color="Yellow" font="Microsoft Sans Serif; 9pt" horizStretch="1"
  vertiStretch="1,32" x="150" y="84" w="164" h="20" />
<label id="lblFile" color="2; 19; 54" font="Microsoft Sans Serif; 8pt; style=Bold"
  x="138" y="52" w="258" h="14" scrollable="true" />

<!-- Rating box -->
<rating id="rating" x="135" y="24" w="96" h="16" fileOK="star.png" fileBad="cross.png" />
```

Once the skin template is ready, put into a .zip archive the `main.xml` file and every image used in your skin. Call this archive as you prefer. Put this file in JeniuS skins directory, open JeniuS, select your new skin from the settings window. Open JMediaPlayer to test your skin. When the skin is finished, rename the archive replacing `.zip` extension with `.jmpskin` and insert your skin in the [JMediaPlayer skin collection](#).

Troubleshooting

If you receive an error message when you are trying to load your new skin in JMediaPlayer, probably you have made some mistakes in the "main.xml" file.

To find and fix them you can validate the skin template (the "main.xml" file) using a xml validator service such as this [XSD Schema Validator](#) by gotdotnet.com

Simply insert the main.xml file of your skin and the latest skin schema file you find at

<http://www.jockersoft.com/schemas/jmpskin.xsd> ([download schema - .zip file](#))

This will tell you exactly what's wrong with your skin template.

If you have any problem or question, do not hesitate to ask in the forums

Appendix

Valid values format

Valid color values

Valid colors values are:

- named colors. Any of the standard [SVG 1.1 color keywords](#) plus `Transparent` (IE: `Transparent`, `Black`, `Red`, `Blue`, `Yellow`, `Magenta`)
- RGB representation in this format: `RRR; GGG; BBB` (IE: `255; 069; 000` is `OrangeRed`, `050; 205; 050` is `LimeGreen`, ...)

Valid font values

Valid font values must have one of these formats:

- `FONTNAME; SIZEpt`
- `FONTNAME; SIZEpt; style=STYLE`

Where FONTNAME is the font name (IE: `Microsoft Sans Serif` or `Arial` or `Verdana`...)

SIZE is the size of the font, expressed in points (IE `8` or `6,75` or `11,5`). Must be followed by `pt` (points)

STYLE is optional (default is `Regular`). Must be one or more of the following:

- `Regular`
- `Bold`
- `Italic`
- `Strikeout`
- `Underline`

Some examples of valid font values are: `"Microsoft Sans Serif; 8pt; style=Bold"`, `"Microsoft Sans Serif; 8,25pt"` `"Trebuchet; 8pt; style=Bold"`

Valid boolean values

boolean attributes can assume only the `true` or `false` values.

Remember that `True` and `tRue` are considered different from `true` and so they are not valid.

Skin conversion from BSplayer

When converting a BSplayer skin to JMediaPlayer, there are a few things to keep in mind:

- due to the different way the two programs handle skin elements and the presence/lack of some functions, the conversion result will probably need some additional edits to:
 - refine labels position and size
 - solve some problems in trackbars (maybe the converted trackbar will have a dark background or dark border set)
 - replace any button not supported in JMediaPlayer with something else
 - add controls supported only by JMediaPlayer (consider adding the `rating` box, the `AddFileToJenius` and `LoadJenius` buttons)
- You can import only BSplayer 1.x skins. If your skin is for the previous vesion (0.8x) you can use [BSP SkinMaker](#) to convert it to the newer format before importing it with JeniuS
- BSplayer supports only one color as transparent for the background image. Instead with JMediaPlayer you can have a background with partial transparency. This let you create skins with shades, antialiased borders and so on. If you can, replace the main background image with a 32bit png one which has all this nice stuff (and set `partialTransparency` to `true` in the `background` element, and provide a mask Image, as described [here](#) and [here](#))

Functions not supported in BSplayer, new in JMediaPlayer:

Buttons:

Volume

AddFileToJenius

LoadJenius

CustomPosition

see [button functions](#) for an explanation of what they do

Trackbars:

barBalance

barSpeed

Other:

rating

For example here is how I modified phantomlord's BSPlayer Showtime 1.4 skin for JMediaPlayer:



Conversion table from BSPlayer button actions to JMediaPlayer button ids

BSPlayer version: 1.37 (b826).

In future JMediaPlayer releases, support for more functions will be provided.

This list is here for your convenience only. You are encouraged to use Jenius' skin import tool.

"" means that this action isn't currently supported

```

Action# (Meaning) -> "JMediaPlayerButtonName"
-----
0 (BSP_ExitFSscreen) -> "Fullscreen"
1 (BSP_VolUp) -> "VolumeUp"
2 (BSP_VolDown) -> "VolumeDown"
3 (BSP_DeDynUp //DeDynamic Filter Increase Amplification) -> ""
4 (BSP_DeDynPreUp //DeDynamic Filter Increase Pre-Amplification) -> ""
5 (BSP_DeDynDown) -> ""
6 (BSP_DeDynPreDown) -> ""
7 (BSP_Preferences) -> "Settings"
8 (BSP_FrmCapture //Capture Frame - Original image size) -> ""
9 (BSP_Frm2 //Capture Frame - "What you see") -> ""
10 (BSP_FS_Switch) -> "Fullscreen"
11 (BSP_SubSEndI) -> "SubtitlesOnOff"
12 (BSP_Skins) -> "Skins"
13 (BSP_AStrmVolCyc //Audio Stream Volume Cycle) -> ""
14 (BSP_Rew) -> "SeekBackwards"
15 (BSP_Forw) -> "SeekForwards"
16 (BSP_SubCorInc //Subtitles Time Correction + / VobSub Increase Delay) -> ""
17 (BSP_SubCorDec) -> ""
18 (BSP_SubCorIncS //Subtitles Time Correction+(Small Steps)/VobSub Increase Speed) -> ""
19 (BSP_SubCorDecS) -> ""
20 (BSP_Play) -> "Play"
21 (BSP_Pause) -> "Pause"
22 (BSP_Stop) -> "Stop"
23 (BSP_ViewChp //Chapter Viewer) -> ""
24 (BSP_VBlankSwitch //Switch wait for vertical-blank) -> ""
25 (BSP_Prev) -> "PreviousTrack"
26 (BSP_PrevCh //Previous Chapter) -> ""
27 (BSP_PrevCD) -> ""
28 (BSP_Next) -> "NextTrack"
29 (BSP_NextCh) -> ""
30 (BSP_NextCD) -> ""
31 (BSP_ATop) -> "VideoAlwaysOnTop"
32 (BSP_OvrTop //Always on Top Overlay Mode) -> ""
33 (BSP_AspCyc) -> "AspectCycle"
34 (BSP_PlayList) -> "ShowPlaylist"
35 (BSP_Mute) -> "Mute"
36 (BSP_JumpToTime) -> "JumpToTime"
37 (BSP_Zoom50) -> "Zoom50"
38 (BSP_Zoom100) -> "Zoom100"
39 (BSP_Zoom200) -> "Zoom200"

```

```
40 (BSP_AspOrg) -> "AspectOriginal"
41 (BSP_Asp169) -> "Aspect169"
42 (BSP_Asp43) -> "Aspect43"
43 (BSP_FSSW640 //Fullscreen Switch Resolution 640x480) -> ""
44 (BSP_FSSW800 //Fullscreen Switch Resolution 800x600) -> ""
45 (BSP_VInf) -> "VideoInfo"
46 (BSP_PanIn) -> "PanIn"
47 (BSP_PanOut) -> "PanOut"
48 (BSP_ZoomIn) -> "ZoomIn"
49 (BSP_ZoomOut) -> "ZoomOut"
50 (BSP_MoveLeft // Move Picture Left (in fullscreen)) -> ""
51 (BSP_MoveRight) -> ""
52 (BSP_MoveUp) -> ""
53 (BSP_MoveDown) -> ""
54 (BSP_FRSizeLeft //Free Resize Left (in fullscreen)) -> ""
55 (BSP_FRSizeRight) -> ""
56 (BSP_FRSizeUp) -> ""
57 (BSP_FRSizeDown) -> ""
58 (BSP_ResetMov) -> ""
59 (BSP_HideCtrl) -> "Minimize"
60 (BSP_EQ) -> ""
61 (BSP_OpenAud) -> "OpenAudio"
62 (BSP_OpenSub) -> "LoadSubtitles"
63 (BSP_OpenMov) -> "Open"
64 (BSP_PanScan //Pan-Scan (Auto)) -> ""
65 (BSP_CusPanScan) -> "CustomPanScan"
66 (BSP_DeskMode //Desktop Mode) -> ""
67 (BSP_AddBk) -> "AddBookmark"
68 (BSP_EditBK) -> "EditBookmark"
69 (BSP_SkinRefr) -> ""
70 (BSP_About) -> "Credits"
71 (BSP_CycleAS //Cycle Audio Stream) -> "Random"
72 (BSP_CycleSub //Cycle Subtitles) -> ""
73 (BSP_IncPBRate) -> "IncrementPlaybackSpeed"
74 (BSP_DecPBRate) -> "DecrementPlaybackSpeed"
75 (BSP_IncPP //Increase Post-Processing) -> ""
76 (BSP_DecPP) -> ""
77 (BSP_Exit) -> "Exit"
78 (BSP_CloseM) -> "CloseFile"
79 (BSP_JumpF) -> "SeekForwardsLarge"
80 (BSP_JumpB) -> "SeekBackwardsLarge"
81 (BSP_ChBordEx//Select Chapter) -> ""
82 (BSP_CycleVid//Cycle Video Stream) -> ""
83 (BSP_IncFnt) -> "SubFontPlus"
84 (BSP_DecFnt) -> "SubFontMinus"
85 (BSP_IncBri //Increase Brightness) -> ""
86 (BSP_DecBri) -> ""
87 (BSP_MovSubUp) -> "MoveSubtitlesUp"
88 (BSP_MovSubDown) -> "MoveSubtitlesDown"
89 (BSP_SHTime //Show / Hide Movie Time (OSD)) -> ""
90 (BSP_IncBriHW) -> ""
91 (BSP_DecBriHW) -> ""
92 (BSP_IncConHW) -> ""
93 (BSP_DecConHW) -> ""
94 (BSP_IncHueHW) -> ""
95 (BSP_DecHueHW) -> ""
96 (BSP_IncSatHW) -> ""
97 (BSP_DecSatHW) -> ""
98 (BSP_ShowHWClr //Show Hardware Color Controls Dialog) -> ""
99 (BSP_IncMovWin //Resize Movie Window +) -> ""
100 (BSP_DecMovWin //Resize Movie Window -) -> ""
101 (BSP_IncPBRatel) -> "IncrementPlaybackSpeedSmall"
102 (BSP_DecPBRatel) -> "DecrementPlaybackSpeedSmall"
103 (BSP_SWRepeat //Switch Repeat Modes) -> "Random"
104 (BSP_SWDispFmt //Switch Time / Frames) -> "SwitchStatusDisplayMode"
105 (BSP_FastForw) -> "FastForwards"
106 (BSP_FastRew) -> "FastBackwards"
107 (BSP_OpenURL) -> ""
```