DEX 1.1 Skinning Documentation Page

DOCUMENTATION FOR THE XML SKINNING SYSTEM.

There are 2 other documents as well:
Skinning the playlist
Skinning Object Definitions

Basics

To create a skin, you need a folder with the name of your skin, and a .xml file with the description of the skin. The directory will contain all pictures necessary for your skin.
Directory: `c:\Program Files\PCDJ DEX\skins\mySkin\`

XML: `c:\Program Files\PCDJ DEX\skins\mySkin.xml`

The .xml file contains one xml object, named skin, which is started by `<skin>` at the beginning of the file,

and `</skin>` at the end of the file.
All text outside this tag is ignored.

Because with a full skin, the xml can get quite large, which
makes it hard to edit, it is also possible to split the xml in different parts.
All parts that you want to use have to be stored in the skin's directory (PCDJ
DEX\skins\myskin\playerwindow.xml
for example)

You can then include this file in your main skin object by using the include tag
as follows:
`<INCLUDE>playerwindow.xml</INCLUDE>`

At almost any place, you may add a comment field, which is also ignored by
PCDJ DEX.

If you want to share your skin with others, it is nice to have an
installer, which makes it easier for other people to install your skin.
To help you with that, I've created a NSIS template. The only thing you need to
change in the script is the name of your skin.
You can download the template here. (this is a zip. extract and use the .nsi file) To compile it, you
need NSIS, which is a freeware
installer system.

Once your skin is ready, and you have created an installer for it, you can
post it on the forum.
If you still have some questions about creating a skin, feel free to ask them on
the forum as well.

LIVE CHAT
NOT AVAILABLE

Skin helper utility

To help you create your skin’s xml, there is also a little utility available to generate the xml objects by selecting the required fields.
The utility can be found Here.

Skin xml file layout

A skin will look like this

```
<SKIN>
    <INFO>
    …
    </INFO>

    <GENERAL>


    …
    </GENERAL>

    <ELEMENTS>
    …
    </ELEMENTS>

    <COLORS>
    …
    </COLORS>

    <FONTS>
    …
    </FONTS>

    <PLAYERWINDOW>
    …
    </PLAYERWINDOW>

    <MAIN>
    …

    </MAIN>

    <FXWINDOW>
    …
    </FXWINDOW>

    <PLAYLIST>

    …
    </PLAYLIST>

    <WINDOW>
    …
    </WINDOW>

</SKIN>
```

Info Field

```
<INFO>
    <NAME>Skin Name</NAME>
    <AUTHOR>Author Name</AUTHOR>

    <DATE>March 19, 2004</DATE>
    <LAST-UPDATED>February 11, 2006</LAST-UPDATED>
    <URL>http://www.djdecks.be</URL>
    <MINIMALBUILD>3998</MINIMALBUILD>

</INFO>
```

In the future, this information might be shown while selecting the skin.
Minimal build is the lowest DEX version (indicated by build number) that is required to optimally use this skin.
You can find out which build you are using by opening the console and scrolling to the top.

General Field

```
<GENERAL>
    <BACKGROUND>djdecks_background</BACKGROUND>

    <PLAYERS>2</PLAYERS>
    <SHOW-BACKGROUND>yes</SHOW-BACKGROUND>
    <SHOW-EFFECTS>yes</SHOW-EFFECTS>
    <RESOLUTION-WIDTH>1024</RESOLUTION-WIDTH>

    <RESOLUTION-HEIGHT>768</RESOLUTION-HEIGHT>
</GENERAL>
```

Background specifies the background image.
For all images, DEX first looks in the skins\skinName folder, and tries .bmp, .jpg, .gif and .png extensions.
If the image isn’t found, the skins root directory is searched.
DEX_background is the name of the default background that is included with DEX. You don’t need to copy it in your own skin directory.

Players defines the number of players to be shown by default. If you want to make a skin that consists of only one window with all controls on it, use 0 here.
2 is the default, and will result in 2 separate windows for the players.

You can also choose if you want to show an effect panel for every player, and if you want to show a background.

Resolution-Height and **Resolution-Width** specifies the original resolution of the skin.
This helps the automatic resizing to make sure the skin fits the screen on different resolutions.

The default is 1024×768

Elements Field

```
<ELEMENTS>
    <PICTURE>elements</PICTURE>
    <ELEMEMT>
        …

    </ELEMENT>
    <ELEMENT>
        …
    </ELEMENT>
    …
</ELEMENTS>
```

Elements are additional small images that you may need in your skin.
An example is the handle of a slider that shows at which position the slider is.
Elements may also be the pictures of a button when it is pressed or when the mouse hovers over it.

The **Picture** field specifies in which picture all these elements are located.

Each element field looks like this:

```
<ELEMENT>
    <NAME>Sliderbutton_Horizontal</NAME>

    <POSITION>
        <X>10</X>
        <Y>9</Y>
        <WIDTH>275</WIDTH>

        <HEIGHT>19</HEIGHT>
    </POSITION>
</ELEMENT>
```

Name specifies the name of the element, position specifies where the element is located in the elements picture.
To specify the **position**, you must specify the top-left coordinate with the X and Y fields.

You can specify width and height of the element with the WIDTH and HEIGHT fields, or you can specify the bottom-right coordinate with the X2 and Y2 fields.

The elements field has to be specified before the following fields, because they will need this information.

Fonts Field

```
<FONTS>
```

```
        <FONT>
            …
        </FONT>

        <FONT>
            …
        </FONT>
        …
    </FONTS>
```

The fonts field specifies all the different fonts you want to use.

```
<FONT>
    <NAME>myLargeFont</NAME>
    <SIZE>20</SIZE>
    <BOLD>yes</BOLD>

    <FACE>Arial</FACE>
    <ALLOWLARGER>yes</ALLOWLARGER>
    <ALLOWSMALLER>yes</ALLOWSMALLER>
    <CLEARTYPE>no</CLEARTYPE>

    <ANGLE>0</ANGLE>
</FONT>
```

There is one font created by default, which is called ‘default’
If you create a font with the name ‘default’, then this font will
overwrite the default font.
If you later define an object, and you don’t specify a font, then the default
font is used.
The ‘Size’ field specifies the height of the font in pixels. If you rather want

to specify the fontsize in points (as normally used in windows) you should use a <POINTSIZE> field instead of the <SIZE> field.

The allowlarger and allowsmaller fields can force the size to stay the same even when the resolution is increased or decreased.
This is particularly useful for the playlist font, that may not be readable anymore at 800×600 if smaller font size is allowed.
If these fields are omitted, they default to ‘yes’
Cleartype (optional) specifies if you want the font to be anti-aliased normally, or using cleartype anti-aliasing (only supported in windows XP or later). By default, normal anti-aliasing is used.
Angle (optional) specifies the angle to draw the text in in degrees. Use 90 for vertical text

Colors Field

```
<COLORS>
    <COLOR>

        <NAME>myColor</NAME>
        <R>192</R><G>65</G><B>200</B>
    </COLOR>

    <COLOR>
        …
    </COLOR>
    …
</COLORS>
```

The colors field is optional. It might be useful if you want to be able to change one particular color in your skin to another color.

Playerwindow, Main and FX Window Field

The next 3 fields are very similar. They are the playerwindow, the main and the fxwindow field.
They all consist of objects, which are the separate parts on each window.

```
<MAIN>
    <BACKGROUND>main</BACKGROUND>
    <SELECTED>main_selected</SELECTED>

    <MOUSEOVER>main_mouseover</MOUSEOVER>
    <ACTIVE>main_active</ACTIVE>
    <STARTPOSITION_1>
        <X>5</X>

        <Y>5</Y>
    </STARTPOSITION_1>
    <OBJECT>
        …
    </OBJECT>

    <OBJECT>
        …
    </OBJECT>
    …
</MAIN>
<PLAYERWINDOW>
    <BACKGROUND>player</BACKGROUND>

    <SELECTED>player_selected</SELECTED>
    <MOUSEOVER>player_mouseover</MOUSEOVER>
    <ACTIVE>player_active</ACTIVE>
    <STARTPOSITION_1>

        <X>5</X>
        <Y>5</Y>
    </STARTPOSITION_1>
    <OBJECT>
        …

    </OBJECT>
    …
</PLAYERWINDOW>
<FXWINDOW>
    …
</FXWINDOW>
```

Background specifies the background image for the player and the main window.

Selected specifies an image with the same size as the background image, but with all buttons pressed
MouseOver specifies an image with the same size, but with all buttons selected (when the mouse pointer hovers over them).
Active Some buttons might display a state (for example if loop is on or off). These buttons might also have a different picture when activated.
You can choose whether you want to use a clicked and mouseover picture, or whether you want to use elements for that.
StartPosition_1 specifies the default starting position of this window on the screen. Note that coordinates are set for 1024×768 resolutions, or for skinresize setting of 1. If the user has a different scaling because his resolution is different, then this values will also be automatically adjusted to the correct size.

It is strongly recommended to use this field, because it makes the user
experience better when he switches to your skin.
Note that this only sets the default starting position when the user changes his
skin. The user can still move the windows to a different place if he likes.
For <PLAYERWINDOW> and <FXWINDOW> you can also specify startposition_2 and startposition_3
for
the other 2 decks.
DrawMode: This is an optional advanced setting. Possible values are:
Normal: Default. All objects are redrawn regularly unless the Faster Graphics debug option is selected
by the user

ForceNormal: All objects are redrawn regularly even if Faster Graphics is enabled. Useful for skin
windows that
don’t draw correctly with faster graphics enabled.
FasterGraphics: Objects are only redrawn if their value changes. CPU usage is lower in this mode, but
it doesn’t work well with overlapping objects
Minimal: Default for the effect windows. Objects are only redrawn every 2 seconds or when the user
uses the control with the mouse.

Object Field

An Object looks like this:

```
<OBJECT>

<TYPE>text</TYPE>

    <VALUE>title</VALUE>
    <VALUESPECIFIER>string</VALUESPECIFIER>
    <LEFT-ACTION></LEFT-ACTION>
    <CENTER-ACTION></CENTER-ACTION>
    <RIGHT-ACTION></RIGHT-ACTION>
    <DOUBLECLICK-ACTION></DOUBLECLICK-ACTION>
    <PLAYER>auto</PLAYER>
    <POSITION>
        <X>10</X>

        <Y>9</Y>
        <WIDTH>275</WIDTH>
        <HEIGHT>19</HEIGHT>
    </POSITION>

    <DRAWTYPE>0</DRAWTYPE>
    <COLOR>
        <R>10</R>
        <G>20</G>

        <b>30</b>
    </COLOR>
    <FONT>myLargeFont</FONT>
    <ADDITIONALPIC_1>
        <REF>selected</REF>

    </ADDITIONALPIC_1>
</OBJECT>
```

Type specifies what type of object it is, which defines how the object is drawn. Some examples are text, button and slider.

Value specifies what the value of the object is, for instance the volume of a player, main volume, title, ...

ValueSpecifier specifies more specifically what value should be used.

At the bottom of this page, the values along with their possible valuespecifiers are listed and described in more detail.

The **left, center and right action** specify what action is taken when the user uses the mouse buttons on the object.
You can enter default to use the default action. For instance, for a slider showing the volume of a player, the default action is to change the volume depending on where the user clicked.
It is possible to have another mouse button have a different action, such as resetting the value to it's default.

Player specifies for which player the object is shown. For values such as the main volume, this field has no effect. In the player field, you normally want to use auto for this field. This means that the first player window will affect the properties of the first player, and the second player window will affect the properties of the second player.
You can use this field in the main field, to specify that one slider is used for player 1, and another slider is used for player 2 for instance.

Position defines the position of the object on the window.

Drawtype defines how the object is drawn. For instance for a slider, you might want a handle to drag it or not. How the number is used depends on the type of object.

Color defines the color for the object. For text this will be the font color, for some objects this might be ignored.
Instead of defining the color by it's red, green and blue value, you can also specify it by using <name>myColor</name>, where myColor is the name of a color specified in the colors field.

You can also specify color2, color3 and color4. It depends on the object's type what these colors mean.

Font defines the font to be used. It is the name of a font that you have specified earlier.
You don't have to specify this field. If you don't, a default font is used.

Finally, you can specify up to 4 **additionalpic** fields, labeled additionalpic_1 to 4
You have 3 choices on how to define this. If you use a name field inside it, then the additionalpic is taken from the elements list. The element with the name that you specified is used.

```
    <ADDITIONALPIC_1>

        <NAME>pitchfader_knob</NAME>
    </ADDITIONALPIC_1>
```

You can also use a ref tag, which can be 'background', 'selected', 'mouseover' or 'active'
If this is used, then the picture is taken from the picture that
you specified at the same position. This is especially useful for buttons, where you can just create 3 copies of the background, and change the buttons to how they should look when pressed, active or when the mouse moves over them.

```
    <ADDITIONALPIC_1>
        <REF>mouseover</REF>

    </ADDITIONALPIC_1>
```

Finally, you can also use a file tag, if you want to get the additional pic directly from a file. No extension or path should be specified, the picture will automatically be searched for in the skin's directory.

```
<ADDITIONALPIC_1>
     <FILE>myFile</FILE>
```

```
</ADDITIONALPIC_1>
```

Tabs

Tabs are a quite advanced feature of DEX to allow one part of the screen
to be used for different purposes. A good example is the effect panel, where the
same sliders can be used to control different effect parameters depending on
which effect you choose.
If you are just started skinning, it is best to skip this part and first get
familiar with the basics.

The general outline is as follows, and can be placed inside any window tags (
<PLAYERWINDOW>, <MAIN> or <FXWINDOW>)

```
<TABS>
    <TAB>

        <NAME>Tab’s Name</NAME>
        <DEFAULT>yes</DEFAULT>
        <OBJECT>
        …
        </OBJECT>

        <OBJECT>
        …
        </OBJECT>
        …
    </TAB>
    <TAB>

        …
    </TAB>
    …
</TABS>
```

Since v0.82, you can have multiple tabs sections, each containing as many tab fields as you like.
Each tab has a name, which you can use on a Text or
Button object with Value ‘Button_TabSelect’. Each tab also has a default field,
which indicates with which tab DEX should start as default.
Inside each tab, you can add objects, which will only be active when that tab
is selected.

To select a tab, the user can press a button with Value ‘Button_TabSelect’.
The ValueSpecifier specifies the number of the tab, starting from 1. This button
object can be defined both inside or outside the tabs field.

Playlist field

This field specifies what the playlist/filebrowser will look like.
Since it works slightly different than the other windows, the details are
explained in a separate part of the documentation.

Skinning the playlist

If the playlist field is omitted, a default style is loaded.
Note that if you don’t want to change anything to the default playlist, it is
recommended that you copy the playlist.xml from the Chiron skin to your own
skin and add the <include> for it to your main skin file.

Custom windows

As of v0.87, it is possible to add additional windows if you need more.
This is useful to add a sampler window, or maybe a custom window to show some other info.

A window field looks like this:

```
<WINDOW>
    <NAME>windowname</NAME>
    … objects etc…

</WINDOW>
```

Except for a name field all fields in a window object are the same as
for other windows.
Custom windows are hidden by default, so you will need to add a startup script
or a button on one of the other windows to show your new window.
The following line can be stored in a file **startup.djscript** in your
skin’s folder to open one window.

```
ifnot ShowWindow,windowname|0 click ShowWindow,windowname|0
```

This will open the window called windowname. The zero at the end indicates that
it is for the first deck. Optionally you can use 1, 2 or 3 to open the same window
multiple times, but for different players.

Objects

The different types of objects you can create, and the value and valuespecifier
fields that can be used with them are explained in a separate document:

Skin Objects

PCDJ DEX 3 - Sidelist-Singers List Preview Video: http://t.co/6k1cMAjUHk via @YouTube about 29 mins ago

PCDJ @PCDJSoftware

Follow @PCDJSoftware

Digital 1 DJ
611 S. Ft. Harrison Ave., #317
Clearwater, FL 33756
Toll Free: 1-877-999-7235