Skin XML Structure

Table of Contents

Overview

The layout, structure and tag definitions for the skin files.

Introduction

Unlike web pages, XML files are not forgiving of any mistakes and using the correct case for tag names is very important.

Note: **XML files are CASE SENSITIVE**.

Skin XML Structure

XML Version and Character Encoding

Skin xmls start by defining the XML version and character encoding, usually:

<?xml version="1.0" encoding="utf-8" standalone="yes"?>

<window> element

The first element in a skin file **must** be the <window> tag which contains information that applies to the entire window. Here is a partial skin.xml file for example purposes. Each of the tags is defined below:

```
<window>
  <id>1234</id>
  <defaultcontrol>2</defaultcontrol>
  <allowoverlay>yes</allowoverlay>
  <autohidetopbar>no</autohidetopbar>
  <disabletopbar>no</disabletopbar>
  <define>#header.label:5900</define>
  <define>#header.image:trailers_logo.png</define>
  <define>#header.hover:hover_my trailers.png</define>
  <controls>
    <import>common.window.xml</import>
    <control>
    ...
```

Note: Not all skin xmls contain a Window ID, for example, 'common' files which are imported into 'window' xmls.

Element Name	Data Type	Description
id	Integer	Each skin file must have a unique id and it must match the id in the application or plugin using the skin. This id is used to reference this skin page from a hyperlink in another skin page.
defaultcontrol	Integer	Specifies the id of the default control in this skin file

allowoverlay	Boolean (yes or no)	Show the video/tv/music preview screen in the bottom corner when a media file is playing
autohidetopbar	Boolean (yes or no)	Automatically hide the top bar
disabletopbar	Boolean (yes or now)	Disable top bar completely in the current window
define	String	Specify a value of "#*Named Token*:*Replacement Value*". Any occurrence of the token represented by the define's name will be replaced with the appropriate value in any imported markup. Images are imported from the Media within the current skin directory. Labels (e.g. 5900 in the above example) are imported from the *language* folder in the MediaPortal application folder. This element may be repeated as many times as tokens are required.

<define> tags

Define tags normally appear at the top of a skin xml for easy reference.They allow a basic form of parameter to be used so that imported files can be as generic as possible. A define tag's value is a simple name - value pairing.

- <define> tags can be used in any XML with a window ID
- <define> tags are entered in the window elements control at the top of the xml before the <controls>.
- 'global' <define> tags can be also be used in the references.xml to define global values

Note: As of v 1.3.0 beta, <define> tags are 'promoted' to skin properties to allow them to contain skin expressions, and be added to the window definition through the use of an <include> file. See Defines for further info.

Sample

The following would be a typical example of how this new feature could be used:

```
<window>
  <!-- include the common window features here -->
  <define>#header.label:134</define>
  <define>#header.image:videos_logo.png</define>
  <define>#header.hover:hover_my videos.png</define>
  <controls>
    <!-- include the header logo, text and other common elements -->
    <import>common.window.xml</import>
    <control>
    ...
```

The value 134 in the above example references a text string in the corresponding language file. For english, this would be the ""strings_en.xml"" in the *language* folder in the MediaPortal application folder.

Any occurrence of the token represented by the define's name will be replaced with the appropriate value in any imported markup. The following is an example of how an imported file could look:

```
<window>
  <control>
    <type>image</type>
    <id>1</id>
    <posX>60</posX>
    <posY>20</posY>
    <texture>#header.image</texture>
  </control>
  <control>
    <type>label</type>
    <id>1</id>
    <posX>250</posX>
    <posY>70</posY>
    <label>#header.label</label>
    <font>font16</font>
    <textcolor>ffffffff</textcolor>
  </control>
  ...
```

<controls> element

The controls element is a container for all of the controls on a page.

Element Name	Data Type	Description
import	String	The name of a xml file to import within the same skin folder. Normally used to reference all the common markup into smaller and easier to manage files. You may have multiple import elements
include	String	Same as import - as of v 1
condition	Boolean	As of v 1.3.0 beta, skins may use any valid skin expression or skin setting that evaluates to a boolean
control	Control	A graphical control. There will be as many of these as there are controls on a window

<control> element

All the GUI controls are put between the <controls> and </controls> tag and each one is contained with a <control> tag. Some controls (e.g. group) may allow you to nest subcontrols within them so you may have <control> tags within a parent <control> tag.

```
<window>
  ...
  <controls>
    ...
    <control>
      <id>1</id>
      <description>default button</description>
      <type>button</type>
      ...
```

Every control is derived from a base control (GuiControl) and has at least the following attributes:

Element Name	Data Type	Description
id	Integer	The id of the control. The id will couple the skin file to the code, so if we later on want to check that a user pressed a button, the id will be required and must be unique. For controls that will never be referenced in the code it is safe to set it to "1"
description	String	An optional description of the control for your reference
type	String	The type of control you want. See the list below for valid types of control
posX	Integer	The X-position on the window for this control (left edge = 0, measures positive to the right)
posY	Integer	The Y-position on the window for this control (top edge = 0, measures positive down)
width	Integer	The width of this control
height	Integer	The height of this control
onleft	Integer	The control id to move the focus to when the user moves left. If not specified (or zero) MediaPortal will find the closest control in that direction to move to
onright	Integer	The control id to move the focus to when the user moves right. If not specified (or zero) MediaPortal will find the closest control in that direction to move to
onup	Integer	The control id to move the focus to when the user moves up. If not specified (or zero) MediaPortal will find the closest control in that direction to move to
ondown	Integer	The control id to move the focus to when the user moves down. If not specified (or zero) MediaPortal will find the closest control in that direction to move to
colordiffuse	Long	Allows you to mix a color & a graphics texture. E.g. If you have a graphics texture like a blue button you can mix it with a yellow color diffuse and the end result will be green. Defaults to 0xFFFFFFFF
dimColor	Integer	Color for a control when it is not focused. Defaults to half transparent (0x60ffffff)

<import> or <include> files

Skin files have the ability to import/include files into the skin markup using the <import> or <include> tags.

The concept is to move all the common markup into smaller and easier to manage files to minimize the amount of work required to create new skin files and to make it easier to maintain and modify existing skin files.

The <import> control should be added to the <controls> section of the xml. The elements defined in the imported file will overlay elements which are defined previously in the xml, so the location of the <import> control is significant.

Two of the most commonly imported files are common.window.xml which defines elements common to most or all windows, and common.time.xml which defines the display of the date and time on each window.

See Import or include for further details.