

Global Menu Pro version 1.0.0

ADVANCED COMPONENT
FOR FLASH 8, FLASH CS3/CS4

Overview

The Global Menu Pro is a Flash menu component that represents a menu structure in a pop-up fashion with sliding submenus. It allows you to implement a hierarchical menu into your website navigation system. The common characteristic of this menu is that space-saving submenus slide out on demand, revealing many layers of depth in the menu.

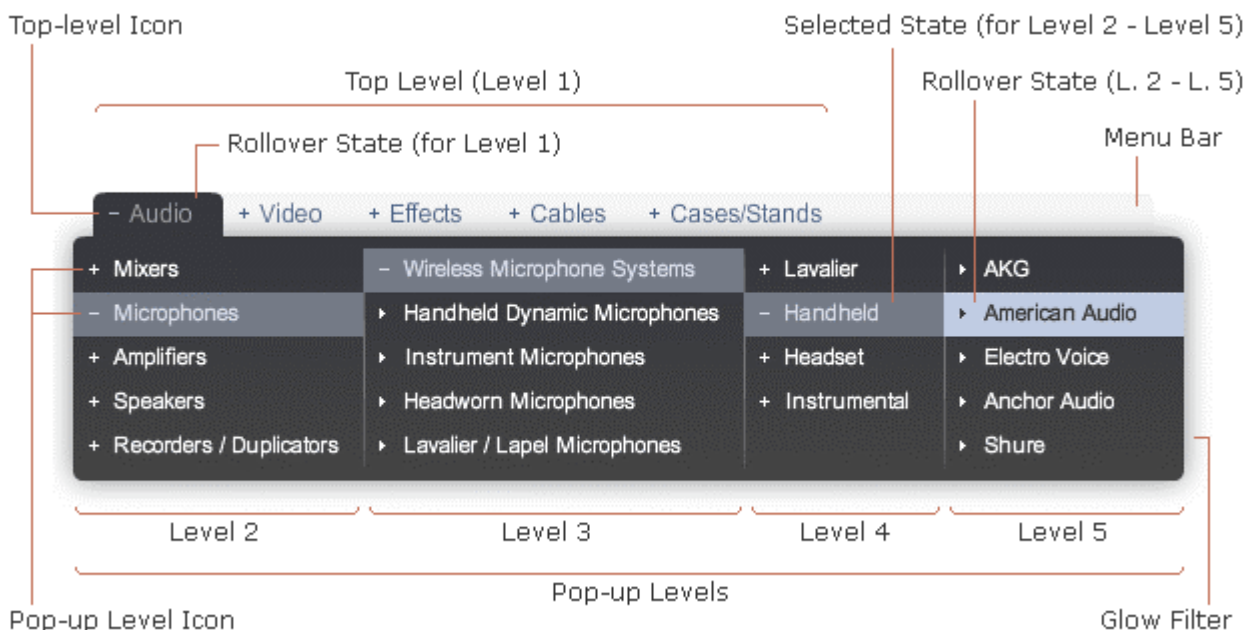
The Global Menu Pro is designed to handle both large professional and small sites. It can therefore easily accommodate menus running into 100's, 10's or 1's of menu items.

The component provides with great usability supporting most complete feature set:

- multilevel structure (up to 5 levels) and unlimited menu items
- menu data is loaded using XML
- component can be completely [configured through the XML file](#)
- (optionally) separate style declaration for each of top-level menu items
- (optionally) setting specific values of width for any submenu block
- all parameters (excepting those corresponding to particular items) can be easily set through the Component Inspector
- alpha transparency support for all backgrounds
- gradient support for background color
- adding URLs or custom Flash functions to any menu click event (through the XML file)

The structure of Global Menu Pro includes following levels:

1. *Menu bar* — a horizontal menu bar.
2. *Top level (static)* — consists of items that are always visible. Move your mouse over a top-level item, and nested second-level submenu (if one exists) pops up. Only one top-level item can be open at a time.
3. *Pop-up levels (dynamic, optional)* — are only visible when a corresponding top-level item is open. Up to 4 nested levels are supported (Level 2 - Level 5). Clicking on an item with "+" or "-" icon respectively reveals or hides nested submenu by sliding the submenu block to the right or to the left.

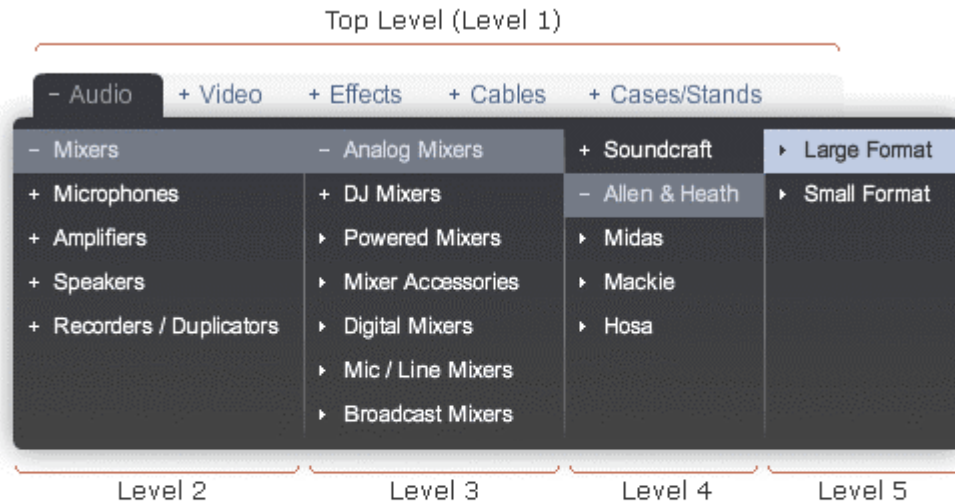


Features

Multilevel structure

- Sliding submenus with up to 4 submenu levels.
- High scalability: can easily manage 100's of menu entries.
- Unlimited menu items can be added at any level in the menu hierarchy.

5-level menu



1-level menu



Separate style declaration for each of top-level menu items

Global Menu Pro supports separate style declaration for each of top-level item and its submenus. You can do the following:

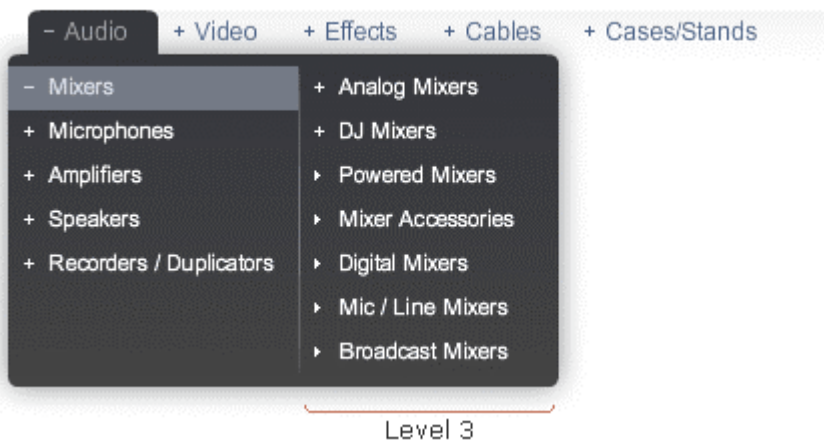
- set individual styles for text color
- set individual styles for background color
- define specific values of some parameters

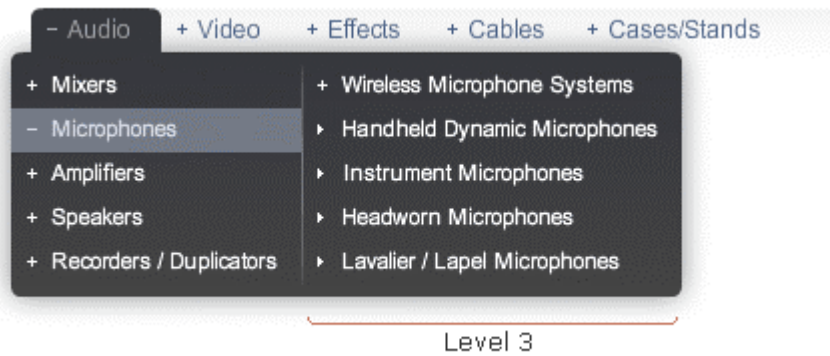
Separate style declaration can be configured through the XML file (see [XML file structure](#) for detailed information).



Setting specific values of width for any submenu block

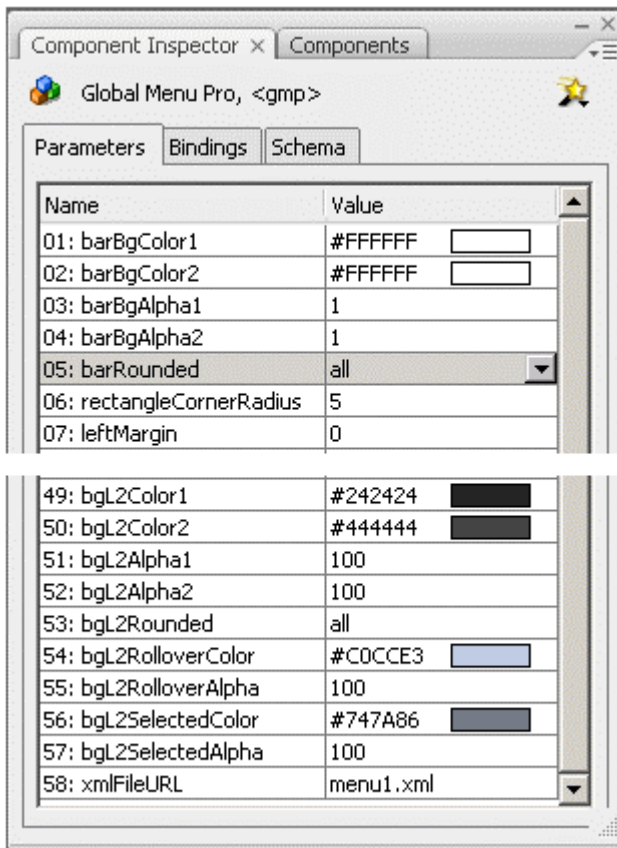
Global Menu Pro enables you to set a specific value of width for any submenu through the attributes of its parent node (see [XML file structure](#) for detailed information).





Setting all common parameters through the component inspector

All common parameters (excepting those corresponding to particular menu items) can be easily set through the Component Inspector (see [Component parameters](#) for more information).



Alpha transparency support for all backgrounds

You can set the alpha transparency for the following elements of Global Menu Pro:

- background of the menu bar
- background of a first-level item in the default state
- background of a first-level item in the rollover state
- background of submenus of pop-up (second, third, fourth, fifth) levels

- background of a pop-up level item in the rollover state
- background of a pop-up level item in the selected state



Gradient support for background color

You can apply a linear gradient fill (vertical) for the following elements of Global Menu Pro:

- background of the menu bar
- background of a first-level item in the default state
- background of submenus of pop-up levels (second, third...)



Using Global Menu Pro

The Global Menu Pro component can be used to create the professionally looking navigation system for websites of any complexity. Using parameter settings in the authoring environment and ActionScript methods, properties and events available for the component, you can build advanced and full-featured navigation systems. Elegant design and excellent dynamics will create unique and attractive appearance for different types of Flash and HTML-based web applications.

Use Global Menu Pro to create any of the following:

- Horizontal pop-up menus
- Dropdown menus
- Sliding menus
- Any combinations of above menus

This is not the complete list of possible usages, just ones that are likely to be used most often. Your creative power and imagination can greatly extend the scope of the Global Menu Pro component.

Global Menu Pro parameters

You can set the following authoring parameters for each Global Menu Pro component instance in the Component inspector:

1. Menu Bar parameters

barBgColor1: The starting (top) color of the gradient fill of the menu bar background. The default value is "#FFFFFF".

barBgColor2: The ending (bottom) color of the gradient fill of the menu bar background. The default value is "#FFFFFF".

barBgAlpha1: The starting (top) alpha transparency of the gradient fill of the menu bar background. Possible values: from 0 to 100. The default value is 0.

barBgAlpha2: The ending (bottom) alpha transparency of the gradient fill of the menu bar background. Possible values: from 0 to 100. The default value is 0.

```
barBgColor1 = "#CCCCCC"  
barBgAlpha1 = 100  
barBgColor2 = "#CCCCCC"  
barBgAlpha2 = 100
```



a) solid fill, no transparency

```
barBgColor1 = "#F5F5F5"  
barBgAlpha1 = 80  
barBgColor2 = "#FFFFFF"  
barBgAlpha2 = 0
```



a) gradient fill, with transparency

Note: If you need to use a solid fill (with no gradient), the value of `barBgColor2` must be equal to the value of `barBgColor1` and the value of `barBgAlpha2` must be equal to the value of `barBgAlpha1`.

barRounded: Specifies which corners of the menu bar to be drawn rounded. This parameter can be one of six predefined values: "all", "left", "right", "top", "bottom", "none". The default value is "all".

The size of the menu bar can be set while authoring or [through the XML file](#). While authoring, select the Global Menu Pro component on the Stage and use the Free Transform tool or any of the Modify > Transform commands.

2. Geometrical and Functional parameters

rectangleCornerRadius: Specifies the corner radius for all background fills (rectangles), in pixels. The default value is 5.

leftMargin: The left margin of the menu, relative to the left edge of the menu bar, in pixels. The default value is 0.

itemHeightL1: The height of each top-level item, in pixels. The default value is 30.

itemHeightL2: The height of each pop-up level item, in pixels. The default value is 25.

itemWidthL1: The width of each top-level item, in pixels. The default value is 100.

itemWidthL2: The width of each second-level item, in pixels. The default value is 150.

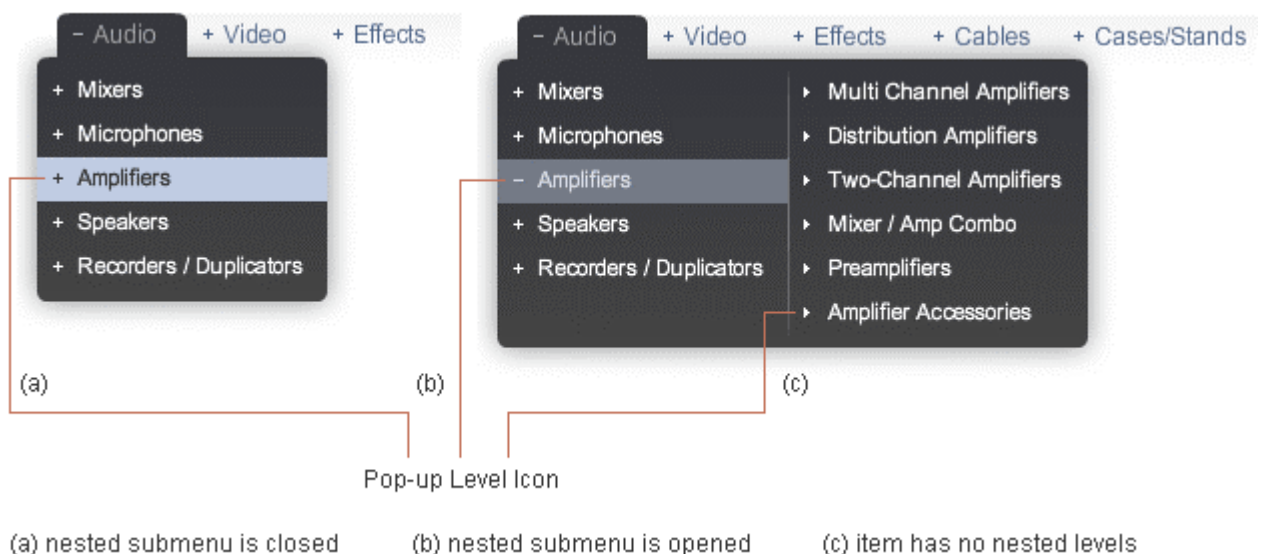
itemWidthL3: The width of each third-level item, in pixels. The default value is 150.

itemWidthL4: The width of each fourth-level item, in pixels. The default value is 150.

itemWidthL5: The width of each fifth-level item, in pixels. The default value is 150.

topLevelIcons: Determines whether to draw the icons for the top-level menu items. The top-level icons are built-in and predefined. The default value is "true".

popupLevelIcons: Determines whether to draw the icons for the menu items of pop-up levels. The pop-up level icons are built-in and predefined. The default value is "true".



Note: In this context, "pop-up level" means "second, third, fourth or fifth level" of menu.

topLevelIconSize: The size of the icon for each top-level menu item, in pixels. The value of this parameter is controlled programmatically and depends on the specified font size and font family. Use

topLevelIconSize optionally if you want to set your custom value for the size of a top-level icon. Possible values: 5, 7, 9, 12, 14. The default value is undefined.

popupLevelIconSize: The size of the icon for each pop-up level item, in pixels. The value of this parameter is controlled programmatically and depends on the specified font size and font family. Use popupLevelIconSize optionally if you want to set your custom value for the size of a pop-up level icon. Possible values: 5, 7, 9, 12, 14. The default value is undefined.

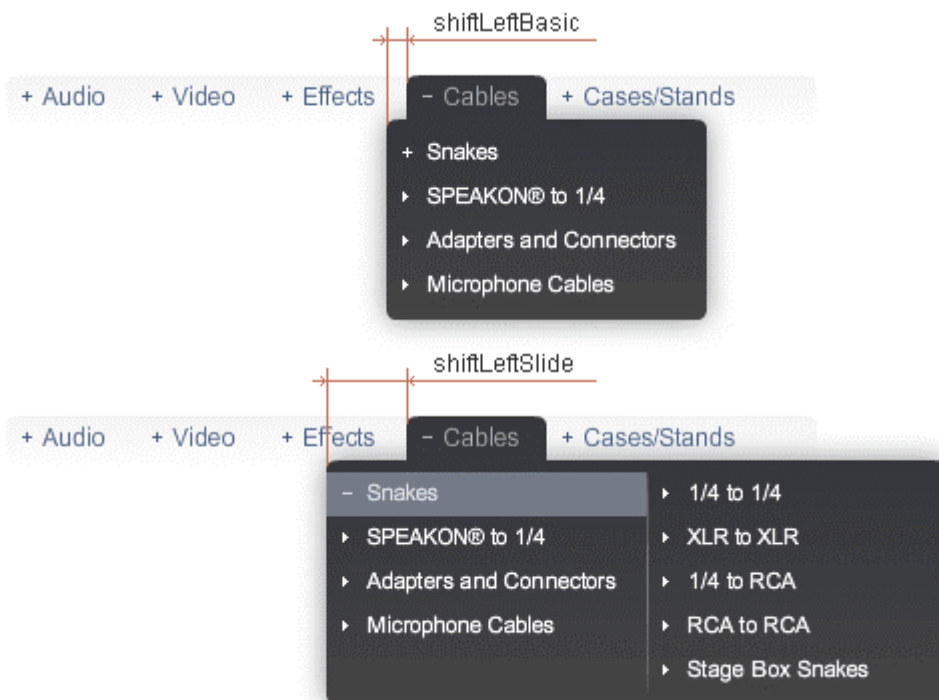
tweenDurationRollover: A number indicating the duration of the tweened animation in seconds. The tweened animation is applied to each top-level item and its nested submenus when fading in and out. This parameter accepts values from 0 to 0.7. The default value is 0.2.

tweenDurationSliding: A number indicating the duration of the tweened animation in seconds. The tweened animation is applied to submenu blocks when sliding to the right or to the left. This parameter accepts values from 0 to 0.5. The default value is 0.2.

fadeOutDelay: Amount of time (in seconds) to wait before fading out the rolled-over top-level item and its nested submenus after the mouse pointer is no longer positioned over the menu. This parameter accepts values from 0 to 3. The default value is 0.4.

shiftLeftBasic: Specifies the distance (in pixels) between the left edge of the top-level item and the left edge of its nested second-level submenu in the state when the third-level submenu is closed. The default value is 10.

shiftLeftSlide: Specifies the distance (in pixels) between the left edge of the top-level item and the left edge of its nested second-level submenu in the state when the third-level submenu is opened. The default value is 40.



glowFilter: Determines whether to apply a glow effect to pop-up submenu blocks. The default value is "true".

itemRollAnimation: Lets you select which type of animation effect to apply to a pop-up level item when it is rolled over or rolled out. This parameter can be one of three predefined values: 1, 2, 3. The default value is 1.

3. Text Style parameters

fontFamily: The font name for text. The default value is "_sans".

embedFonts: A Boolean value that indicates whether the font specified in `fontFamily` is an embedded font. This parameter must be set to "true" if `fontFamily` refers to an embedded font. Otherwise, the embedded font is not used. If this parameter is set to true and `fontFamily` does not refer to an embedded font, no text is displayed. The default value is "false".

fontSizeL1: The point size for the font of a top-level item text. Possible values: from 8 to 96. The default value is 12.

fontSizeL2: The point size for the font of a pop-up level item text. Possible values: from 8 to 96. The default value is 11.

fontColorL1: The text color of a top-level item. The default value is "#6683AF".

fontColorL2: The text color of a pop-up level item. The default value is "#FFFFFF".

fontColorRolloverL1: The color of text when the pointer rolls over a top-level item. The default value is "#AFB2B6".

fontColorRolloverL2: The color of text when the pointer rolls over a pop-up level item. The default value is "#383838".

fontColorSelectedL2: The color of text in the selected pop-up level item. The default value is "#DADEE9".

fontColorDisabledL1: The text color of a top-level item when it is disabled. The default value is "#979797".

fontColorDisabledL2: The text color of a pop-up level item when it is disabled. The default value is "#B1B4B8".

fontWeightL1: The font weight for the text of a top-level item. Possible values: "normal", "bold". The default value is "normal".

fontWeightL2: The font weight for the text of a pop-up level item. Possible values: "normal", "bold". The default value is "normal".

fontStyleL1: The font style for the text of a top-level item. Possible values: "normal", "italic". The default value is "normal".

fontStyleL2: The font style for the text of a pop-up level item. Possible values: "normal", "italic". The default value is "normal".

textDecorationL1: The text decoration of a top-level item. Possible values: "none", "underline". The default value is "none".

textDecorationRolloverL1: The text decoration when the pointer rolls over a top-level item. Possible values: "none", "underline". The default value is "none".

textAlignL1: The text alignment of a top-level item. Possible values: "left", "center", "right". The default value is "left".

textPaddingTopL1: The spacing between the top edge of a top-level item and the text of its label, in pixels. Use `textPaddingTopL1` optionally if you want to set your custom value for this parameter. The default value is undefined.

textPaddingTopL2: The spacing between the top edge of a pop-up level item and the text of its label, in pixels. Use `textPaddingTopL2` optionally if you want to set your custom value for this parameter. The default value is undefined.

textPaddingLeftL1: Specifies the spacing (in pixels) between: (a) the left edge of a top-level item and the text of its label (or the top-level icon, if `topLevelIcons` is set to "true"), (b) the right edge of a top-level item and the text of its label. Use `textPaddingLeftL1` optionally if you want to set your custom value for this parameter. The default value is undefined.

textPaddingLeftL2: The spacing between the left edge of a pop-up level item and the text of its label (or the pop-up level icon, if `popupLevelIcons` is set to "true"), in pixels. Use `textPaddingLeftL2` optionally if you want to set your custom value for this parameter. The default value is undefined.

4. Background Style parameters

bgL1Color1: The starting (top) color of the gradient fill of a top-level item's background. The default value is "#FFFFFF".

bgL1Color2: The ending (bottom) color of the gradient fill of a top-level item's background. The default value is "#FFFFFF".

bgL1Alpha1: The starting (top) alpha transparency of the gradient fill of a top-level item's background. Possible values: from 0 to 100. The default value is 0.

bgL1Alpha2: The ending (bottom) alpha transparency of the gradient fill of a top-level item's background. Possible values: from 0 to 100. The default value is 0.

Note: If you need to use a solid fill (with no gradient), the value of `bgL1Color2` must be equal to the value of `bgL1Color1` and the value of `bgL1Alpha2` must be equal to the value of `bgL1Alpha1`.

bgL1Rounded: Specifies which corners of a top-level item's background to be drawn rounded. This parameter can be one of six predefined values: "all", "left", "right", "top", "bottom", "none". The default value is "top".

bgL1RolloverColor: The background color of a rolled-over top-level item. The default value is "#242424".

bgL1RolloverAlpha: The background transparency of a rolled-over top-level item. Possible values: from 0 to 100. The default value is 100.

bgL2Color1: The starting (top) color of the gradient fill of a pop-up level item's background. The default value is "#242424".

bgL2Color2: The ending (bottom) color of the gradient fill of a pop-up level item's background. The default value is "#444444".

bgL2Alpha1: The starting (top) alpha transparency of the gradient fill of a pop-up level item's background. Possible values: from 0 to 100. The default value is 100.

bgL2Alpha2: The ending (bottom) alpha transparency of the gradient fill of a pop-up level item's background. Possible values: from 0 to 100. The default value is 100.

Note: If you need to use a solid fill (with no gradient), the value of `bgL2Color2` must be equal to the value of `bgL2Color1` and the value of `bgL2Alpha2` must be equal to the value of `bgL2Alpha1`.

bgL2Rounded: Specifies which corners of a pop-up level item's background to be drawn rounded. This parameter can be one of four predefined values: "all", "top", "bottom", "none". The default value is "all".

bgL2RolloverColor: The background color of a rolled-over pop-up level item. The default value is "#C0CCE3".

bgL2RolloverAlpha: The background transparency of a rolled-over pop-up level item. Possible values: from 0 to 100. The default value is 100.

bgL2SelectedColor: The background color of a selected pop-up level item. The default value is "#747A86".

bgL2SelectedAlpha: The background transparency of a selected pop-up level item. Possible values: from 0 to 100. The default value is 100.

5. XML File parameters

xmlFileURL: This parameter is the path to the configuration XML file. The default value is undefined.

Note: Another way to set the path to XML file is to pass the variable `xmlURL` to a SWF (containing Global Menu Pro component instance) through HTML tags (see [example](#) for more information).

XML file structure

The structure of the Global Menu Pro component XML file enables you to configure parameters and styles on the following levels:

1. Inside the `<settings>` element — all common parameters and styles (exactly the same as in the [Component inspector](#)).
2. Inside those `<item>` nodes of the `<items>` element that describe top-level menu items — separate style declaration for each of top-level item and its submenus (overrides the parameters of the same name set inside the `<settings>` element).
3. Inside each `<item>` node of the `<items>` element — some individual parameters concerning a single menu item (overrides the parameters of the same name set inside the `<settings>` element or through the separate style declaration).

Note: Setting a property in the XML file overrides the parameter of the same name set in the Component inspector. If the value of a property in the XML file is undefined (omitted), it will be ignored.

The basic template of the XML file (without separate style declaration):

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
  <settings>
    <menuBar width="405" height="25" color1="F5F5F5" alpha1="100" color2=" " alpha2="" rounded="all"/>
    <rectangleCornerRadius value="6"/>
    <leftMargin value="0"/>
    <itemHeight level1="22" level2="20"/>
    <itemWidth level1="80" level2="120" level3="140" level4="140" level5="100"/>
    <topLevelIcons enabled="true" size=""/>
    <popupLevelIcons enabled="true" size=""/>
    <tweenDuration rollover="0.2" sliding="0.3"/>
    <fadeOut delay="0.4"/>
    <shiftLeft basic="10" slide="40"/>
    <glowFilter enabled="true"/>
    <itemRollAnimation mode="1"/>
    <fontFamily name="arial_embed"/>
    <embedFonts enabled="true"/>
    <fontSize level1="12" level2="11"/>
    <fontColor level1="445C82" level2="FFFFFF"/>
    <fontColorRollover level1="AFB2B6" level2="383838"/>
    <fontColorSelected level2="DADEE9"/>
    <fontColorDisabled level1="979797" level2="B1B4B8"/>
    <fontWeight level1="normal" level2="normal"/>
    <fontStyle level1="normal" level2="normal"/>
    <textDecoration level1="none"/>
    <textDecorationRollover level1="none"/>
    <textAlign level1="left"/>
    <textPaddingLevel1 top="" left=""/>
    <textPaddingLevel2 top="" left=""/>
    <backgroundLevel1 color1="FFFFFF" alpha1="0" color2="FFFFFF" alpha2="0" rounded="top"/>
    <backgroundLevel1Rollover color="37383E" alpha="100"/>
    <backgroundLevel2 color1="37383E" alpha1="100" color2="444444" alpha2="100" rounded="all"/>
    <backgroundLevel2Rollover color="C0CCE3" alpha="100"/>
    <backgroundLevel2Selected color="747A86" alpha="100"/>
  </settings>
  <items>
    <item label="Audio">
      <item label="Mixers">
        <item label="Analog Mixers">
          <item label="Soundcraft">
            <item label="Live" action="getUrl"
              url="/audio/mixers/analog_mixers/soundcraft/live/" target="_self"></item>
          </item>
        </item>
      </item>
    </item>
  </items>
</menu>
```

The advanced sample of the XML file (the fragment showing separate style declaration and individual parameters applied to a single menu item):

```
<items>
  <item
    label="Audio"
    action=""
    url=""
    target=""
    widthL1="65"
    widthL2="145"
    enabled=""
    shiftLeftSlide="10"
    itemL2Opened="3"
    fontColorL1="FFFFFF"
    fontColorL2=""
    fontColorRolloverL1="F5D6D6"
    fontColorRolloverL2=""
    fontColorSelectedL2="FAEBEB"
    fontColorDisabledL1=""
    fontColorDisabledL2="CDBEB4"
    bgL1Color1="EFB4B4"
    bgL1Color2="C62C2B"
    bgL1Alpha1="100"
    bgL1Alpha2="100"
    bgL1Rounded=""
    bgL1RolloverColor="C62C2B"
    bgL1RolloverAlpha="100"
    bgL2Color1="C62C2B"
    bgL2Color2="444444"
    bgL2Alpha1="100"
    bgL2Alpha2="100"
    bgL2Rounded=""
    bgL2RolloverColor="F5D6D6"
    bgL2RolloverAlpha="100"
    bgL2SelectedColor="F5D6D6"
    bgL2SelectedAlpha="50"
  >
  <item label="Mixers" widthL3="150" fontColor="FF9933">
    <item label="Analog Mixers" widthL4="130">
      <item label="Soundcraft" widthL5="120">
        <item label="Live" action="getUrl"
          url="/audio/mixers/analog_mixers/soundcraft/live/" target="_self"></item>
        <item label="Multi-Purpose" enabled="false"></item>
      </item>
    </item>
  </item>
</items>
```

The description of the XML file elements

1. The following table describes the attributes of nodes that the **<settings>** element contains:

Node name	Attribute name	Default	Description
menuBar	width	500	The width of the menu bar, in pixels.
	height	30	The height of the menu bar, in pixels.
	color1	"FFFFFF"	The starting (top) color of the gradient fill of the menu bar background.
	alpha1	0	The starting (top) alpha transparency of the gradient fill of the menu bar background. Possible values: from 0 to 100.
	color2	"FFFFFF"	The ending (bottom) color of the gradient fill of the menu bar background.
	alpha2	0	The ending (bottom) alpha transparency of the gradient fill of the menu bar background. Possible values: from 0 to 100.
	rounded	"all"	Specifies which corners of the menu bar to be drawn rounded. Possible values: "all", "left", "right", "top", "bottom", "none".
rectangleCornerRadius	value	5	Specifies the corner radius for all background fills (rectangles), in pixels.
leftMargin	value	0	The left margin of the menu, relative to the left edge of the menu bar, in pixels.
itemHeight	level1	30	The height of each top-level item, in pixels.
	level2	25	The height of each pop-up level item, in pixels.
itemWidth	level1	100	The width of each top-level item, in pixels.
	level2	150	The width of each second-level item, in pixels.
	level3	150	The width of each third-level item, in pixels.
	level4	150	The width of each fourth-level item, in pixels.
	level5	150	The width of each fifth-level item, in pixels.
topLevelIcons	enabled	"true"	Determines whether to draw the icons for the top-level menu items. The top-level icons are built-in and predefined. Possible values: "true", "false".
	size	undefined	The size of the icon for each top-level menu item, in pixels. The value of this parameter is controlled programmatically and depends on the specified font size and font family. Use this attribute optionally if you want to set your custom value for the size of a top-level icon. Possible values: 5, 7, 9, 12, 14.
popupLevelIcons	enabled	"true"	Determines whether to draw the icons for the menu items of pop-up levels. The pop-up level icons are built-in and predefined. Possible

	size	undefined	values: "true", "false". The size of the icon for each pop-up level menu item, in pixels. The value of this parameter is controlled programmatically and depends on the specified font size and font family. Use this attribute optionally if you want to set your custom value for the size of a pop-up level icon. Possible values: 5, 7, 9, 12, 14.
tweenDuration	rollover	0.2	A number indicating the duration of the tweened animation in seconds. The tweened animation is applied to each top-level item and its nested submenus when fading in and out. Possible values: from 0 to 0.7.
	sliding	0.2	A number indicating the duration of the tweened animation in seconds. The tweened animation is applied to submenu blocks when sliding to the right or to the left. Possible values: from 0 to 0.5.
fadeOut	delay	0.4	Amount of time (in seconds) to wait before fading out the rolled-over top-level item and its nested submenus after the mouse pointer is no longer positioned over the menu. Possible values: from 0 to 3.
shiftLeft	basic	10	Specifies the distance (in pixels) between the left edge of the top-level item and the left edge of its nested second-level submenu in the state when the third-level submenu is closed.
	slide	40	Specifies the distance (in pixels) between the left edge of the top-level item and the left edge of its nested second-level submenu in the state when the third-level submenu is opened.
glowFilter	enabled	"true"	Determines whether to apply a glow effect to pop-up submenu blocks. Possible values: "true", "false".
itemRollAnimation	mode	1	Lets you select which type of animation effect to apply to a pop-up level item when it is rolled over or rolled out. Possible values: 1, 2, 3.
fontFamily	name	"_sans"	The font name for text.
embedFonts	enabled	"false"	A Boolean value that indicates whether the font specified in fontFamily is an embedded font. This parameter must be set to "true" if fontFamily refers to an embedded font. Otherwise, the embedded font is not used. If this parameter is set to true and fontFamily does not refer to an embedded font, no text is displayed. Possible values: "true", "false".
fontSize	level1	12	The point size for the font of a top-level item text. Possible values: from 8 to 96.

	level2	11	The point size for the font of a pop-up level item text. Possible values: from 8 to 96.
fontColor	level1	"6683AF"	The text color of a top-level item.
	level2	"FFFFFF"	The text color of a pop-up level item.
fontColorRollover	level1	"AFB2B6"	The color of text when the pointer rolls over a top-level item.
	level2	"383838"	The color of text when the pointer rolls over a pop-up level item.
fontColorSelected	level2	"DADEE9"	The color of text in the selected pop-up level item.
fontColorDisabled	level1	"979797"	The text color of a top-level item when it is disabled.
	level2	"B1B4B8"	The text color of a pop-up level item when it is disabled.
fontWeight	level1	"normal"	The font weight for the text of a top-level item. Possible values: "normal", "bold".
	level2	"normal"	The font weight for the text of a pop-up level item. Possible values: "normal", "bold".
fontStyle	level1	"normal"	The font style for the text of a top-level item. Possible values: "normal", "italic".
	level2	"normal"	The font style for the text of a pop-up level item. Possible values: "normal", "italic".
textDecoration	level1	"none"	The text decoration of a top-level item. Possible values: "none", "underline".
textDecorationRollover	level1	"none"	The text decoration when the pointer rolls over a top-level item. Possible values: "none", "underline".
textAlign	level1	"left"	The text alignment of a top-level item. Possible values: "left", "center", "right".
textPaddingLevel1	top	undefined	The spacing between the top edge of a top-level item and the text of its label, in pixels. Use this attribute optionally if you want to set your custom value for this parameter.
	left	undefined	Specifies the spacing (in pixels) between: (a) the left edge of a top-level item and the text of its label (or the top-level icon, if topLevelIcons is set to "true"), (b) the right edge of a top-level item and the text of its label. Use this attribute optionally if you want to set your custom value for this parameter.
textPaddingLevel2	top	undefined	The spacing between the top edge of a pop-up level item and the text of its label, in pixels. Use this attribute optionally if you want to set your custom value for this parameter.
	left	undefined	The spacing between the left edge of a pop-up

backgroundLevel1	color1	"FFFFFF"	level item and the text of its label (or the pop-up level icon, if popupLevelIcons is set to "true"), in pixels. Use this attribute optionally if you want to set your custom value for this parameter.
	alpha1	0	The starting (top) alpha transparency of the gradient fill of a top-level item's background.
	color2	"FFFFFF"	The starting (top) color of the gradient fill of a top-level item's background.
	alpha2	0	The starting (top) alpha transparency of the gradient fill of a top-level item's background. Possible values: from 0 to 100.
	rounded	"top"	The ending (bottom) color of the gradient fill of a top-level item's background.
backgroundLevel1Rollover	color	"242424"	The ending (bottom) alpha transparency of the gradient fill of a top-level item's background. Possible values: from 0 to 100.
	alpha	100	Specifies which corners of a top-level item's background to be drawn rounded. Possible values: "all", "left", "right", "top", "bottom", "none".
backgroundLevel2	color	"242424"	The background color of a rolled-over top-level item.
	alpha1	100	The background transparency of a rolled-over top-level item. Possible values: from 0 to 100.
	color1	"242424"	The starting (top) color of the gradient fill of a pop-up level item's background.
	alpha1	100	The starting (top) alpha transparency of the gradient fill of a pop-up level item's background. Possible values: from 0 to 100.
	color2	"444444"	The ending (bottom) color of the gradient fill of a pop-up level item's background.
backgroundLevel2Rollover	alpha2	100	The ending (bottom) alpha transparency of the gradient fill of a pop-up level item's background. Possible values: from 0 to 100.
	rounded	"all"	Specifies which corners of a pop-up level item's background to be drawn rounded. Possible values: "all", "top", "bottom", "none".
	color	"C0CCE3"	The background color of a rolled-over pop-up level item.
backgroundLevel2Selected	alpha	100	The background transparency of a rolled-over pop-up level item. Possible values: from 0 to 100.
	color	"747A86"	The background color of a selected pop-up level item.
	alpha	100	The background transparency of a selected pop-up level item. Possible values: from 0 to 100.

Note: If you need to use a solid fill (with no gradient), the values of color1 and color2 attributes must be equal, and the values of alpha1 and alpha2 attributes must be equal as well.

2. The following table describes the specific attributes you can add to those **<item>** nodes of the **<items>** element that correspond to the top-level menu items. This can be useful if you need to apply separate style declaration for each or certain of top-level item and its submenus.

Note: In this context, "a pop-up level item" means "a second-, third-, fourth- or fifth-level item that belongs to the hierarchy of the current top-level menu item".

Node name	Attribute name	Default	Description
item	widthL1	undefined	The width of the current top-level item, in pixels.
	widthL2	undefined	The width of the nested second-level submenu, in pixels.
	shiftLeftSlide	undefined	The distance (in pixels) between the left edge of the current top-level item and the left edge of its nested second-level submenu in the state when the third-level submenu is opened.
	itemL2Opened	undefined	Lets you set the number of the second-level item which nested submenu you want to be in the open state when the pointer rolls over the current top-level item. This can be useful if you want to attract user's attention to a specific menu item and its content.
	fontColorL1	undefined	The text color of the current top-level item.
	fontColorL2	undefined	The text color of a pop-up level item.
	fontColorRolloverL1	undefined	The color of text when the pointer rolls over the current top-level item.
	fontColorRolloverL2	undefined	The color of text when the pointer rolls over a pop-up level item.
	fontColorSelectedL2	undefined	The color of text in the selected pop-up level item.
	fontColorDisabledL1	undefined	The text color of the current top-level item when it is disabled.
	fontColorDisabledL2	undefined	The text color of a pop-up level item when it is disabled.
	bgL1Color1	undefined	The starting (top) color of the gradient fill of the top-level item's background.
	bgL1Color2	undefined	The ending (bottom) color of the gradient fill of the top-level item's background.
	bgL1Alpha1	undefined	The starting (top) alpha transparency of the gradient fill of the top-level item's background. Possible values: from 0 to 100.
	bgL1Alpha2	undefined	The ending (bottom) alpha transparency of the gradient fill of the top-level item's background. Possible values: from 0 to 100.
	bgL1Rounded	undefined	Specifies which corners of the top-level item's background to be drawn rounded. Possible values:

			"all", "left", "right", "top", "bottom", "none".
bgL1RolloverColor	undefined		The background color of the current top-level item when rolled over.
bgL1RolloverAlpha	undefined		The background transparency of the current top-level item when rolled over. Possible values: from 0 to 100.
bgL2Color1	undefined		The starting (top) color of the gradient fill of a pop-up level item's background.
bgL2Color2	undefined		The ending (bottom) color of the gradient fill of a pop-up level item's background.
bgL2Alpha1	undefined		The starting (top) alpha transparency of the gradient fill of a pop-up level item's background. Possible values: from 0 to 100.
bgL2Alpha2	undefined		The ending (bottom) alpha transparency of the gradient fill of a pop-up level item's background. Possible values: from 0 to 100.
bgL2Rounded	undefined		Specifies which corners of a pop-up level item's background to be drawn rounded. Possible values: "all", "top", "bottom", "none".
bgL2RolloverColor	undefined		The background color of a rolled-over pop-up level item.
bgL2RolloverAlpha	undefined		The background transparency of a rolled-over pop-up level item. Possible values: from 0 to 100.
bgL2SelectedColor	undefined		The background color of a selected pop-up level item.
bgL2SelectedAlpha	undefined		The background transparency of a selected pop-up level item. Possible values: from 0 to 100.

3. The **<items>** element can contain an unlimited number of **<item>** nodes. Each **<item>** node represents a menu item object and should have one required attribute label. You can also add 5 optional attributes in the **<item>** node: action, url, target, enabled, fontColor.

Node name	Attribute name	Default	Description
item	label	undefined	The text that is displayed to represent a menu item.
	action	undefined	(Optional) This attribute can be either "getUrl" or the name of your custom Flash function (e.g., "pageInformation"). In the first case, Flash getURL() method is called — it loads a document from the specified URL into the specified window. In the second case, your custom Flash function is called.
	url	undefined	(Optional) This attribute defines either the URL from which to obtain the document (if action attribute is set to "getUrl") or any parameters to be passed to your function (if action attribute is set to the name of your custom Flash function). You can specify zero or more parameters, separating them by commas.

target	"_self"	(Optional) This attribute can be either a parameter ("_self", "_blank", "_parent", "_top" or your custom value) that specifies the window or HTML frame that the document is loaded into (if action attribute is set to "getUrl") or a target movie clip where your function is placed (if action attribute is set to the name of your custom Flash function).
enabled	"true"	(Optional) Specifies whether a menu item is enabled or disabled. Possible values: "true", "false".
font_color	undefined	(Optional) The text color of a single menu item (overrides the prior color setting for this particular item).

You can also add one more optional attribute in the **<item>** node: widthL3/widthL4/widthL5.

Node name	Attribute name	Default	Description
item	widthL3	undefined	(Optional) This attribute is set for a second-level item and specifies the width (in pixels) of its third-level submenu.
	widthL4	undefined	(Optional) This attribute is set for a third-level item and specifies the width (in pixels) of its fourth-level submenu.
	widthL5	undefined	(Optional) This attribute is set for a fourth-level item and specifies the width (in pixels) of its fifth-level submenu.

Note: All colors should be expressed in RRGGBB format.

Creating an application with Global Menu Pro component

The following procedure explains how to add Global Menu Pro component to an application while authoring.

Example 1

In this example, Global Menu Pro is used to create the navigation system for HTML-based web application.

To create an application with the Global Menu Pro component:

1. Set the Stage size to 570 x 370 pixels and leave white as background color. For frame rate enter the value not less than 30 fps.
2. Drag one Global Menu Pro component from the Components panel to the Stage and set its coordinates to (20, 20).
3. In the Component inspector leave all parameters without changes (with default values).
4. Set dimensions for the component instances to 530 x 25 pixels.
5. Publish the Flash document with appropriate name — for example, "menu.swf".

6. Using your favorite text editor, create a new document and enter the following code:

Note: The full code of this XML file (as well as the source FLA file) is available in the package of the Global Menu Pro paid version (see the Sample 1 files).

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
  <settings>
    <menuBar width="530" height="25" color1="F5F5F5" alpha1="100" color2="FFFFFF" alpha2="70"
      rounded="all"/>
    <rectangleCornerRadius value="6"/>
    <leftMargin value="0"/>
    <itemHeight level1="22" level2="22"/>
    <itemWidth level1="" level2="" level3="" level4="" level5=""/>
    <topLevelIcons enabled="true" size=""/>
    <popupLevelIcons enabled="true" size=""/>
    <tweenDuration rollover="0.2" sliding="0.3"/>
    <fadeOut delay="0.4"/>
    <shiftLeft basic="10" slide="40"/>
    <glowFilter enabled="true"/>
    <itemRollAnimation mode="1"/>
    <fontFamily name="Arial"/>
    <embedFonts enabled="false"/>
    <fontSize level1="12" level2="11"/>
    <fontColor level1="445C82" level2="FFFFFF"/>
    <fontColorRollover level1="AFB2B6" level2="383838"/>
    <fontColorSelected level2="DADEE9"/>
    <fontColorDisabled level1="979797" level2="B1B4B8"/>
    <fontWeight level1="normal" level2="normal"/>
    <fontStyle level1="normal" level2="normal"/>
    <textDecoration level1="none"/>
    <textDecorationRollover level1="none"/>
    <textAlign level1="left"/>
    <textPaddingLevel1 top="" left=""/>
    <textPaddingLevel2 top="" left=""/>
    <backgroundLevel1 color1="" alpha1="0" color2="" alpha2="0" rounded="top"/>
    <backgroundLevel1Rollover color="37383E" alpha="100"/>
    <backgroundLevel2 color1="37383E" alpha1="100" color2="444444" alpha2="100" rounded="all"/>
    <backgroundLevel2Rollover color="C0CCE3" alpha="100"/>
    <backgroundLevel2Selected color="747A86" alpha="100"/>
  </settings>
  <items>
    <item label="Audio" widthL1="65" widthL2="145" shiftLeftSlide="10">
      <item label="Mixers" widthL3="130">
        <item label="Analog Mixers" widthL4="100">
          <item label="Soundcraft" widthL5="170">
            <item label="Live" action="getUrl"
              url="/audio/mixers/analog_mixers/soundcraft/live/" target="_self"></item>
            <item label="Multi-Purpose" action="getUrl"
              url="/audio/mixers/analog_mixers/soundcraft/multi_purpose/"
              target="_self"></item>
            <item label="Recording" action="getUrl"
              url="/audio/mixers/analog_mixers/soundcraft/recording/" target="_self"></item>
          </item>
        <item label="Allen & Heath" widthL5="100">
          <item label="Large Format" action="getUrl"
            url="/audio/mixers/analog_mixers/allen/large_format/" target="_self"></item>
        </item>
      </item>
    </item>
  </items>
</menu>
```

```

        <item label="Small Format" action="getUrl"
            url="/audio/mixers/analog_mixers/allen/small_format/" target="_self"></item>
    </item>
    <item label="Midas" action="getUrl" url="/audio/mixers/analog_mixers/midas/"
        target="_self"></item>
    <item label="Mackie" action="getUrl" url="/audio/mixers/analog_mixers/mackie/"
        target="_self"></item>
    <item label="Hosa" action="getUrl" url="/audio/mixers/analog_mixers/hosa/"
        target="_self"></item>
    .....
</item>
</item>
</items>
</menu>

```

This is your configuration XML document. Save the XML file in preferred location (for example, in the same directory as the SWF file that contains a Global Menu Pro component) and name it, for example, "menu.xml".

7. Create an HTML file and place the following HTML code into it — to embed SWF file (in our case, menu.swf):

```

<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0"
width="570" height="370" id="menu" align="middle">
    <param name="allowScriptAccess" value="sameDomain"/>
    <param name="movie" value="menu.swf"/>
    <param name="FlashVars" value="xmlURL=menu.xml"/>
    <param name="quality" value="high"/>
    <param name="bgcolor" value="#FFFFFF"/>
    <embed src="menu.swf" FlashVars="xmlURL=menu.xml" quality="high" bgcolor="#FFFFFF" width="570"
height="370" name="menu" align="middle" allowScriptAccess="sameDomain" type="application/x-
shockwave-flash" pluginspage="http://www.adobe.com/go/getflashplayer"/>
</object>

```

Note: Global Menu Pro can accept one external variables: xmlURL — the path to configuration XML file.

8. Save the HTML file in preferred location (for example, in the same directory as the SWF file that contains a Global Menu Pro component) and name it, for example, "menu.html".
9. Open menu.html in a web browser and test the behavior of the Global Menu Pro component.

Example 2

In this example, Global Menu Pro is used to create the navigation system for Flash-based web application.

Before creating an application, download the following files required for this example and save it to your hard disk:

http://e-merald.com/pic/components/city_bg.jpg

http://e-merald.com/pic/components/home_icon.gif

To create an application with the Global Menu Pro component:

1. Set the Stage size to 570 x 303 pixels and specify "#FFFFFF" as background color. For frame rate enter the value not less than 30 fps.

2. Import the saved image files to the Stage by selecting File > Import > Import to Stage. Set the coordinates for the imported images: (0, 0) — for city_bg.jpg; (548, 10) — for home_icon.gif.
3. Drag one Global Menu Pro component from the Components panel to the Stage and set its coordinates to (15, 3). In the Property inspector, set the Instance Name value for the component instance to "gmp".

4. In the Component inspector, do the following:

For xmlFileURL parameter, set: "menu.xml"

You can leave the other parameters without changes and configure the component through the XML file, or you can leave the values for the <settings> element attributes blank and configure the component through the Component Inspector panel.

5. Set dimensions for the component instance to 240 x 20 pixels.
6. Using the Text tool, create a dynamic text field on the Stage. With the text field selected, in the Property inspector, set the Variable value to "header_txt". Create three more dynamic text fields and set the Variable value for them accordingly to "category_txt", "title_txt" and "description_txt". Set appropriate values for other text field properties.
7. In case if you want to use an embedded font in your SWF file, do the following:
 - Be sure the font you want to use is installed on your system (in our case, "Arial").
 - Create a font library item (see [Adobe Flash Help on the web](#) for more information). Enter "arial_embed" for the font item in the Name text field.
 - The fontFamily parameter (this can be either the corresponding node of the XML file or the corresponding parameter in the Component Inspector panel) must refer to the name of the embedded font ("arial_embed", in our case), and the embedFonts parameter must be set to "true".

8. Open the Actions panel, select Frame 1 in the main Timeline, and enter the following ActionScript code:

```

menuListener = new Object();
menuListener.onLoadXML = function(eventObj) {
    xmlDecomposing();
}
gmp.addEventListener("onLoadXML", menuListener);

home_btn.onRelease = function() {
    header_txt = "Italy";
    category_txt = "";
    title_txt = "General Information";
    description_txt = gmp.xml.firstChild.childNodes[1].attributes.description;
}

function xmlDecomposing() {
    itemsArray = new Array();
    header_txt = "Italy";
    category_txt = "";
    title_txt = "General Information";
    description_txt = gmp.xml.firstChild.childNodes[1].attributes.description;
    for (var aNode:XMLNode = gmp.xml.firstChild.childNodes[1].firstChild; aNode != null; aNode = aNode.nextSibling) {
        itemsArray.push(aNode.attributes.description);
        if (aNode.hasChildNodes() == true) {
            for (var bNode:XMLNode = aNode.firstChild; bNode != null; bNode = bNode.nextSibling) {

```

```

        itemsArray.push(bNode.attributes.description);
        if (bNode.hasChildNodes() == true) {
            for (var cNode:XMLNode = bNode.firstChild; cNode != null; cNode = cNode.nextSibling){
                itemsArray.push(cNode.attributes.description);
            }
        }
    }
}
}
}
}

function pageInformation(index:Number, city:String, category:String) {
    header_txt = city;
    category_txt = category;
    title_txt = gmp.getItemAt(index).label;
    description_txt = itemsArray[index];
}

```

9. Publish the Flash document with appropriate name — for example, "menu.swf".
10. Using your favorite text editor, create a new document and enter the following code:

Note: The full code of this XML file (as well as the source FLA file) is available in the package of the Global Menu Pro paid version (see the Sample 4 files).

```

<?xml version="1.0" encoding="UTF-8"?>
<menu>
    <settings>
        <menuBar width="240" height="20" color1="FFFFFF" alpha1="100" color2="FFFFFF" alpha2="100"
            rounded="none"/>
        <rectangleCornerRadius value="6"/>
        <leftMargin value="0"/>
        <itemHeight level1="20" level2="20"/>
        <itemWidth level1="" level2="" level3="" level4="" level5=""/>
        <topLevelIcons enabled="false" size=""/>
        <popupLevelIcons enabled="true" size=""/>
        <tweenDuration rollover="0.2" sliding="0.3"/>
        <fadeOut delay="0.4"/>
        <shiftLeft basic="0" slide=""/>
        <glowFilter enabled="true"/>
        <itemRollAnimation mode="2"/>
        <fontFamily name="arial_embed"/>
        <embedFonts enabled="true"/>
        <fontSize level1="13" level2="12"/>
        <fontColor level1="CC3333" level2="F4F4F4"/>
        <fontColorRollover level1="FEDAD3" level2="CC3333"/>
        <fontColorSelected level2="444444"/>
        <fontColorDisabled level1="979797" level2="B1B4B8"/>
        <fontWeight level1="normal" level2="normal"/>
        <fontStyle level1="normal" level2="normal"/>
        <textDecoration level1="none"/>
        <textDecorationRollover level1="none"/>
        <textAlign level1="center"/>
        <textPaddingLevel1 top="" left=""/>
        <textPaddingLevel2 top="" left=""/>
        <backgroundLevel1 color1="" alpha1="" color2="" alpha2="" rounded="none"/>
        <backgroundLevel1Rollover color="CC3333" alpha="100"/>
        <backgroundLevel2 color1="CC3333" alpha1="100" color2="CC3333" alpha2="70"
            rounded="bottom"/>
    
```

```

    <backgroundLevel2Rollover color="F4F4F4" alpha="80"/>
    <backgroundLevel2Selected color="F4F4F4" alpha="50"/>
</settings>
<items description="Italy, officially the Italian Republic, is a country located on the Italian Peninsula in Southern Europe and on the two largest islands in the Mediterranean Sea, Sicily and Sardinia...">
  <item
    label="Rome"
    widthL1="80"
    widthL2="110"
    shiftLeftSlide="0"
    bgL1Color1="CC3333"
    bgL1Color2="CC3333"
    bgL1Alpha1="20"
    bgL1Alpha2="0"
  >
    <item label="History" widthL3="200">
      <item label="Earliest History" action="pageInformation" url="2,Rome,History" target=""
        description="There is geological evidence of human occupation of the Rome area from at
        least 14000 years..."></item>
      <item label="Monarchy, Republic, Empire" action="pageInformation" url="3,Rome,History"
        target="" description="Rome's early history is shrouded in legend. According to Roman
        tradition, the city was founded by the twins Romulus and Remus..."></item>
      .....
    </item>
  </item>
</items>
</menu>

```

Note: In this example, the <action> attribute of each <item> node is equal to the name of Flash function to be called on menu item's click event. The <url> attribute of each <item> node defines 3 parameters separated by commas: 1) the index of a corresponding menu item, 2) the name of the selected city, 3) the name of the current category. Thus, the value of <url> specifies parameters for the pageInformation() function (see the ActionScript code above). We also add our custom attribute "description" to the <items> element and some <item> nodes. This data can be then easily accessible through the component's `xml` property.

This is your configuration XML document. Save the XML file in preferred location (for example, in the same directory as the SWF file that contains an Global Menu Pro component) and name it, in our case, "menu.xml".

11. Open the Flash document (menu.swf).

GlobalMenuPro class

Inheritance MovieClip > UIObject class > GlobalMenuPro class

Setting a property of the GlobalMenuPro class with ActionScript overrides the parameter of the same name set in the Component inspector.

Method summary for the GlobalMenuPro class

The following table lists methods of the GlobalMenuPro class.

Method	Description
GlobalMenuPro.getItemAt()	Gets a reference to a menu item at a specified location.

Methods inherited from the UIObject class

The following table lists the methods the GlobalMenuPro class inherits from the UIObject class. When calling these methods from the GlobalMenuPro object, use the form *GlobalMenuPro.methodName*.

Method	Description
UIObject.getStyle()	Gets the style property from the style declaration or object.
UIObject.move()	Moves the object to the requested position.

Property summary for the GlobalMenuPro class

The following table lists properties of the GlobalMenuPro class.

Property	Description
GlobalMenuPro.xmlFileURL	The path to the configuration XML file.
GlobalMenuPro.xml	Read-only; the XML object with the downloaded XML data.

Properties inherited from the UIObject class

The following table lists the properties the GlobalMenuPro class inherits from the UIObject class. When accessing these properties from the GlobalMenuPro object, use the form *GlobalMenuPro.propertyName*.

Property	Description
UIObject.left	Read-only; the left edge of the object, in pixels.
UIObject.top	Read-only; the position of the top edge of the object, relative to its parent.
UIObject.visible	A Boolean value indicating whether the object is visible (true) or not (false).
UIObject.x	Read-only; the left edge of the object, in pixels.
UIObject.y	Read-only; the top edge of the object, in pixels.

Event summary for the GlobalMenuPro class

The following table lists the event of the GlobalMenuPro class.

Event	Description
GlobalMenuPro.click	Broadcast when a menu item is clicked.
GlobalMenuPro.closing	Broadcast when a top-level item is returning to the default state.
GlobalMenuPro.onLoadXML	Broadcast when the component attempts to load an XML file.
GlobalMenuPro.opening	Broadcast when a top-level item is rolled over.

Events inherited from the UIObject class

The following table lists the events the GlobalMenuPro class inherits from the UIObject class.

Event	Description
UIObject.hide	Broadcast when an object's state changes from visible to invisible.
UIObject.reveal	Broadcast when an object's state changes from invisible to visible.

GlobalMenuPro.click

Usage

Usage 1:

```
listenerObject = new Object();
listenerObject.click = function(eventObject) {
    // Your code here
}
my_menu.addEventListener("click", listenerObject);
```

Usage 2:

```
on (click) {
    // Your code here
}
```

Description

Event; broadcast to all registered listeners when the mouse is clicked (pressed and released) over a menu item.

The first usage example uses a dispatcher/listener event model. A component instance (*my_menu*) dispatches an event (in this case, *click*) and the event is handled by a function, also called a "handler", on a listener object (*listenerObject*) that you create. You define a method with the same name as the event on the listener object; the method is called when the event is triggered. When the event is triggered, it automatically passes an event object (*eventObject*) to the listener object method. The event object has properties that contain information about the event. You can use these properties to write code that handles the event. Finally, you call the `EventDispatcher.addEventListener()` method on the component instance that broadcasts the event to register the listener with the instance. When the instance dispatches the event, the listener is called.

The second usage example uses an `on()` handler and must be attached directly to a menu instance. The keyword `this`, used inside an `on()` handler attached to a component, refers to the component

instance. For example, the following code, attached to a Global Menu Pro instance *my_menu*, sends "_level0.my_menu" to the Output panel:

```
on (click) {
    trace(this);
}
```

Event object

Along with the standard event object properties, the click event has one additional property — an integer that indicates the index position of the clicked menu item.

Example

This example, written on a frame of the Timeline, sets the `selected_item` variable to the value of the clicked menu item's "label" attribute. The target property of an event object is the component that generated the event. You can access instance properties and methods from the target property (in this example, the `getItemAt()` method is accessed).

```
var selected_item:String;
menuListener = new Object();
menuListener.click = function(eventObj) {
    selected_item = eventObj.target.getItemAt(eventObj.data).label;
}
my_menu.addEventListener("click", menuListener);
```

The following code sends a message to the Output panel when a menu item is clicked. The `on()` handler must be attached directly to a menu instance.

```
on (click) {
    trace("menu item was clicked");
}
```

GlobalMenuPro.closing

Usage

Usage 1:

```
listenerObject = new Object();
listenerObject.closing = function(eventObject) {
    // Your code here
}
my_menu.addEventListener("closing", listenerObject);
```

Usage 2:

```
on (closing) {
    // Your code here
}
```

Description

Event; broadcast to all registered listeners when a top-level item of the Global Menu Pro component is returning to its default (not rolled-over) state.

The first usage example uses a dispatcher/listener event model. A component instance (*my_menu*) dispatches an event (in this case, `closing`) and the event is handled by a function, also called a "handler", on a listener object (*listenerObject*) that you create. You define a method with the same name as the event on the listener object; the method is called when the event is triggered. When the event is triggered, it automatically passes an event object (*eventObject*) to the listener object method. The event object has properties that contain information about the event. You can use these properties to write code that handles the event. Finally, you call the

`EventDispatcher.addListener()` method on the component instance that broadcasts the event to register the listener with the instance. When the instance dispatches the event, the listener is called.

The second usage example uses an `on()` handler and must be attached directly to a menu instance. The keyword `this`, used inside an `on()` handler attached to a component, refers to the component instance. For example, the following code, attached to a Global Menu Pro instance `my_menu`, sends `"_level0.my_menu"` to the Output panel:

```
on (closing) {
    trace(this);
}
```

Event object

Along with the standard event object properties, the closing event has one additional property — an integer that indicates the index position of the top-level menu item that is returning to its default state.

Example

This example, written on a frame of the Timeline, sets the `cl_index` variable to the value of the index position of the top-level menu item that is returning to its default state.

```
var cl_index:Number;
menuListener = new Object();
menuListener.closing = function(eventObj) {
    cl_index = eventObj.data;
}
my_menu.addListener("closing", menuListener);
```

The following code sends a message to the Output panel when a top-level item is returning to its default state. The `on()` handler must be attached directly to a menu instance.

```
on (closing) {
    trace("menu item is closing");
}
```

GlobalMenuPro.getItemAt()

Usage

```
my_menu.getItemAt(index)
```

Parameters

index An integer indicating the index of the node in the menu. This is a zero-based index, so 0 retrieves the first item, 1 retrieves the second item, and so on.

Returns

A reference to the specified node.

Description

Method; returns a reference to the specified child node of the menu.

Example

The following example demonstrates how to retrieve the values of attributes available for the item node with index 5:

```
my_menu.getItemAt(5).label;
my_menu.getItemAt(5).action;
my_menu.getItemAt(5).url;
my_menu.getItemAt(5).target;
```

```
my_menu.getItemAt(5).enabled;  
my_menu.getItemAt(5).fontColor;
```

GlobalMenuPro.onLoadXML

Usage

Usage 1:

```
listenerObject = new Object();  
listenerObject.onLoadXML = function(eventObject) {  
    // Your code here  
}  
my_menu.addEventListener("onLoadXML", listenerObject);
```

Description

Event; broadcast when the Global Menu Pro component attempts to load an XML file.

The usage example uses a dispatcher/listener event model. A component instance (*my_menu*) dispatches an event (in this case, *onLoadXML*) and the event is handled by a function, also called a "handler", on a listener object (*listenerObject*) that you create. You define a method with the same name as the event on the listener object; the method is called when the event is triggered. When the event is triggered, it automatically passes an event object (*eventObject*) to the listener object method. The event object has properties that contain information about the event. You can use these properties to write code that handles the event. Finally, you call the `EventDispatcher.addEventListener()` method on the component instance that broadcasts the event to register the listener with the instance. When the instance dispatches the event, the listener is called.

Event object

Contains a Boolean value ("true" or "false") to indicate whether an XML file was found.

Example

This example, written on a frame of the Timeline, uses the event object to set up a conditional statement that checks to see if an XML file was found and loaded.

```
menuListener = new Object();  
menuListener.onLoadXML = function(eventObj) {  
    if (eventObj.data == true) {  
        // show success alert  
    } else {  
        // show error alert  
    }  
}  
my_menu.addEventListener("onLoadXML", menuListener);
```

GlobalMenuPro.opening

Usage

Usage 1:

```
listenerObject = new Object();  
listenerObject.opening = function(eventObject) {  
    // Your code here  
}  
my_menu.addEventListener("opening", listenerObject);
```

Usage 2:

```

on (opening) {
    // Your code here
}

```

Description

Event; broadcast to all registered listeners when a top-level item of the Global Menu Pro component is rolled over.

The first usage example uses a dispatcher/listener event model. A component instance (*my_menu*) dispatches an event (in this case, *opening*) and the event is handled by a function, also called a "handler", on a listener object (*listenerObject*) that you create. You define a method with the same name as the event on the listener object; the method is called when the event is triggered. When the event is triggered, it automatically passes an event object (*eventObject*) to the listener object method. The event object has properties that contain information about the event. You can use these properties to write code that handles the event. Finally, you call the `EventDispatcher.addEventListener()` method on the component instance that broadcasts the event to register the listener with the instance. When the instance dispatches the event, the listener is called.

The second usage example uses an `on()` handler and must be attached directly to a menu instance. The keyword `this`, used inside an `on()` handler attached to a component, refers to the component instance. For example, the following code, attached to a Global Menu Pro instance *my_menu*, sends "`_level0.my_menu`" to the Output panel:

```

on (opening) {
    trace(this);
}

```

Event object

Along with the standard event object properties, the *opening* event has one additional property — an integer that indicates the index position of the rolled-over top-level menu item.

Example

This example, written on a frame of the Timeline, sets the `op_index` variable to the value of the index position of the top-level menu item that is rolled over.

```

var op_index:Number;
menuListener = new Object();
menuListener.opening = function(eventObj) {
    op_index = eventObj.data;
}
my_menu.addEventListener("opening", menuListener);

```

The following code sends a message to the Output panel when a top-level item is rolled over. The `on()` handler must be attached directly to a menu instance.

```

on (opening) {
    trace("menu item is opening");
}

```

GlobalMenuPro.xmlFileURL

Usage

```
my_menu.xmlFileURL
```

Description

Property; the path to the configuration XML file. The default value is undefined.

Example

The following example sets the `xmlFileURL` property to "data/top_menu.xml":

```
my_menu.xmlFileURL = "data/top_menu.xml";
```

GlobalMenuPro.xml

Usage

```
my_menu.xml
```

Description

Property (read-only); returns the XML object with the downloaded XML data.

Example

This is the configuration XML file for a Global Menu Pro instance named "gmp". There is a custom attribute "description" in the `<items>` element and some `<item>` nodes.

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
  <settings>
    .....
  </settings>
  <items description="Italy, officially the Italian Republic, is a country located...">
    <item label="Rome">
      <item label="History">
        <item label="Earliest History" action="pageInformation" url="2" target=""
          description="There is geological evidence..."></item>
        <item label="Monarchy" action="pageInformation" url="3" target="" description="Rome's
          early history..."></item>
        .....
      </item>
    </item>
  </items>
</menu>
```

The code below shows how to access the attribute values looping through the XML data. Each time we click the certain menu item, the custom Flash fuction `pageInformation` is called and the text field displays the corresponding description (if included in the XML file). See also [example](#) for more information.

```
var country_description:String;
var descriptionArray:Array = new Array();

menuListener = new Object();
menuListener.onLoadXML = function(eventObj) {
  xmlDecomposing();
}
gmp.addEventListener("onLoadXML", menuListener);

function xmlDecomposing() {
  country_description = gmp.xml.firstChild.childNodes[1].attributes.description;
  for (var aNode:XMLNode = gmp.xml.firstChild.childNodes[1].firstChild; aNode != null; aNode =
aNode.nextSibling) {
    descriptionArray.push(aNode.attributes.description);
    if (aNode.hasChildNodes() == true) {
      for (var bNode:XMLNode = aNode.firstChild; bNode != null; bNode = bNode.nextSibling) {
        descriptionArray.push(bNode.attributes.description);
        if (bNode.hasChildNodes() == true) {
```

```
        for (var cNode:XMLNode = bNode.firstChild; cNode != null; cNode = cNode.nextSibling){
            descriptionArray.push(cNode.attributes.description);
        }
    }
}

function pageInformation(index:Number) {
    description_textfield = descriptionArray[index];
}
```