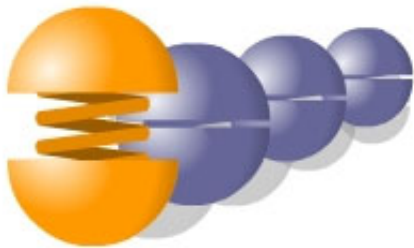


Dynamic



MenuButtons

User Guide

(component version 1.0.0)

Introduction

The DynamicMenuButtons is a set of two components: DynamicButton and DynamicRadioButton. Each of these components behaves similarly to corresponding Macromedia Flash components – Button and RadioButton. Actually DynamicButton and DynamicRadioButton are essentially different from standard Flash components. It is most clearly expressed in the following features. First, the components are “dynamic”: mouse interaction invokes prebuilt ActionScript dynamic effects. Second, the components are multi-purpose and can be easily used in building various combinations for navigation system of a web site or Flash application. Third, the components are extremely customizable.

DynamicButton component

The DynamicButton component is a resizable user interface button with prebuilt dynamic effects and precompiled icon sets. You can add custom icons to a dynamic button. You can also change the behavior of a dynamic button from push to toggle. A toggle dynamic button stays selected when clicked and returns to its unselected state when clicked again.

A dynamic button can be enabled or disabled in an application. In the disabled state, a dynamic button doesn't receive mouse or keyboard input. An enabled dynamic button receives focus if you click it or tab to it. When a DynamicButton instance has focus, you can use the following keys to control it:

Key	Description
Shift+Tab	Moves focus to the previous object.
Tab	Moves focus to the next object.

For more information about controlling focus, see "Creating custom focus navigation" or "FocusManager class" in *Macromedia Flash MX 2004 Using Components Help*.

A live preview of each DynamicButton instance reflects changes made to parameters in the Property inspector or Component inspector during authoring. However, in the live preview a custom icon is represented on the Stage by a gray square.

If a dynamic button is enabled, it displays its rollover unselected state when the pointer moves over it. The dynamic button retains its rollover state when it's pressed. The dynamic button loses its rollover state and receives it again when the mouse is released. If the pointer moves off the dynamic button while the mouse is pressed, the dynamic radio button loses rollover state. If the toggle parameter is set to true, the state of the dynamic button changes (from selected to unselected and vice versa) when the mouse is released over it.

If a button is disabled, it displays its disabled state, regardless of user interaction.

Using the DynamicButton component

A button is a fundamental part of any form or web application. You can use buttons wherever you want a user to initiate an event. The DynamicButton can be easily used in different combinations where appearance of a component is a matter of great importance. It allows creating nonordinary attractive design for your web application, presentation etc.

DynamicButton parameters

You can set the following authoring parameters for each DynamicButton component instance in the Component inspector:

animationEffect sets the type of animation effect applied to the dynamic button. This parameter can be one of five values: Circulation, Heartbeat, Pulsar, Rotation or Vibration; the default value is Circulation. For more information, see [DynamicMenuButtons animation](#).

animationIconSet specifies the set of icons (for selected and unselected state) applied to the dynamic button. This parameter can be one of five values: set1, set2, set3, set4 or set5; the default value is set1. For more information, see [DynamicMenuButtons animation](#).

customIcons is an object type optional parameter that contains two values to be set: `falseIcon` and `trueIcon`. This parameter adds custom icons to the dynamic button. The values are the linkage identifiers of movie clips or graphic symbols in the library. The default values are undefined.

Note: Setting the custom icons disables the action of `animationIconSet`, `iconColor` and `iconColorSelected` parameters. For more information, see [DynamicMenuButtons animation](#).

label sets the value of the text on the dynamic button; the default value is `DynamicButton`.

labelPlacement orients the label text on the dynamic button in relation to the icon. This parameter can be one of three values: `left`, `right` or `bottom`; the default value is `bottom`. For more information, see [DynamicButton.labelPlacement](#).

selected sets the initial value of the dynamic button to `selected` (`true`) or `unselected` (`false`), if the `toggle` parameter is `true`. A selected dynamic button differs from unselected by the icon view. The default value is `false`.

sound is an object type optional parameter that contains two values to be set: `rollover_sound` and `release_sound`. This parameter attaches the sound(s) specified in the Values dialog box to the dynamic button when rolled over (`rollover_sound`) or released (`release_sound`). The sound(s) must be in the library of the current SWF file and specified for export in the Linkage Properties dialog box. The default values are undefined.

toggle turns the dynamic button into a toggle switch. If `true`, the state of the dynamic button changes (from `selected` to `unselected` and vice versa) when clicked. If `false`, the dynamic button behaves like a normal push button; the default value is `false`.

URL is an object type optional parameter that contains three values to be set: `url`, `window`, `delay`. This parameter specifies a document from a specific URL to be load into a window when the dynamic button is clicked.

url. The URL from which to obtain the document. The default value is undefined.

window. (Optional) Specifies the window or HTML frame into which the document should load. You can enter the name of a specific window or select from the following reserved target names: `_self` (the current frame in the current window), `_blank` (a new window), `_parent` (the parent of the current frame), `_top` (the top-level frame in the current window). The default value is `_blank`.

delay. Sets the time delay, in milliseconds, between `onClick` event and the beginning of a specified URL loading. The default value is 500.

To test this function, make sure the file to be loaded is at the specified location.

For other parameters that customize the color and text of `DynamicButton` component, see [Customizing the DynamicButton component](#).

You can write ActionScript to set additional options for `DynamicButton` instances using the methods, properties and events of the `DynamicButton` class. For more information, see [DynamicButton class](#).

Creating an application with the DynamicButton component

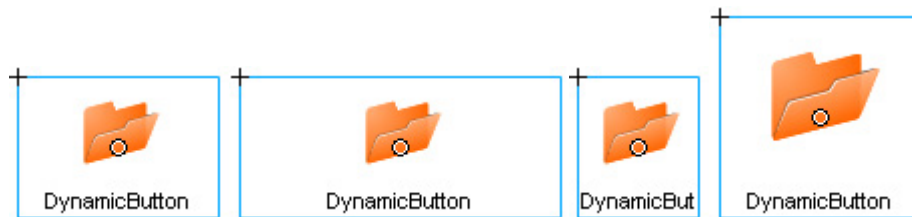
Because the `DynamicButton` component is developed rather as intermediate for the `DynamicRadioButton` component and parameters of these components are much the same, for more information on creating an application with the `DynamicButton` component, see [Creating an application with the DynamicRadioButton component](#).

Customizing the DynamicButton component

You can transform a DynamicButton component horizontally and vertically while authoring and at runtime. While authoring, select the component on the Stage and use the Free Transform tool or any of the Modify > Transform commands. At runtime, use the `setSize()` method.

The bounding box of a DynamicButton component is invisible and also designates the hit area for the component. If you increase the size of the component, you also increase the size of the hit area. The default value for a component's size is 100 x 70 pixels. If the component's bounding box is too small to fit the component label, the label is clipped to fit.

Transforming a component instance vertically results in proportional change of an icon scale. Transforming a component instance horizontally reduces or increases its width and does not resize an icon. Change the component's width to fit the component label if necessary. Modifying the component does not change the height of a label text. For this purpose, you should use the `fontSize` parameter while authoring or `DynamicButton.fontSize` property at runtime.



A DynamicButton component uses the following styles:

Style	Description
fontFamily	The font name for text. The default value is "_sans". <i>ActionScript usage example:</i> <code>myDynamicButton.setStyle("fontFamily", "Courier New");</code>
fontSize	The point size for the font. The default value is 11. <i>ActionScript usage example:</i> <code>myDynamicButton.setStyle("fontSize", 12);</code>
fontStyle	The font style: either "normal" or "italic". The default value is "normal". <i>ActionScript usage example:</i> <code>myDynamicButton.setStyle("fontStyle", "italic");</code>
fontNormalStyle	A set of styles (Object) for the label when a dynamic button is unselected: <i>fontColor.</i> The text color. The default value is 0x000000 (black). <i>fontWeight.</i> The font weight: either "none" or "bold". The default value is "none". <i>textDecoration.</i> The text decoration: either "none" or "underline". The default value is "none". <i>backgroundColor.</i> The color of the text field background. The default value is undefined. <i>borderColor.</i> The color of the text field border. The default value is

	<p>undefined.</p> <p><i>fontColorDisabled</i>. The text color when the component is disabled. The default color is undefined.</p> <p>For information on ActionScript usage, see DynamicButton.fontNormalStyle.</p>
fontOverStyle	<p>A set of styles (Object) for the label when a dynamic button is rolled over:</p> <p><i>fontColor</i>. The text color. The default value is 0x000000 (black).</p> <p><i>fontWeight</i>. The font weight: either "none" or "bold". The default value is "none".</p> <p><i>textDecoration</i>. The text decoration: either "none" or "underline". The default value is "none".</p> <p><i>text</i>. The value of the text on the dynamic button. The default value is undefined.</p> <p><i>backgroundColor</i>. The color of the text field background. The default value is undefined.</p> <p><i>borderColor</i>. The color of the text field border. The default value is undefined.</p> <p>For information on ActionScript usage, see DynamicButton.fontOverStyle.</p>
fontSelectedStyle	<p>A set of styles (Object) for the label when a dynamic button is selected:</p> <p><i>fontColor</i>. The text color. The default value is 0x000000 (black).</p> <p><i>fontWeight</i>. The font weight: either "none" or "bold". The default value is "none".</p> <p><i>textDecoration</i>. The text decoration: either "none" or "underline". The default value is "none".</p> <p><i>text</i>. The value of the text on the dynamic button. The default value is undefined.</p> <p><i>backgroundColor</i>. The color of the text field background. The default value is undefined.</p> <p><i>borderColor</i>. The color of the text field border. The default value is undefined.</p> <p>For information on ActionScript usage, see DynamicButton.fontSelectedStyle.</p>
iconColor	<p>The color of the icon. The default value is 0xFF6600 (orange).</p> <p><i>ActionScript usage example:</i></p> <pre>myDynamicButton.setStyle("iconColor", 0x0066CC);</pre>
iconColorSelected	<p>The color of the icon when the dynamic button is selected. The default value is 0xFF6600 (orange).</p> <p><i>ActionScript usage example:</i></p> <pre>myDynamicButton.setStyle("iconColorSelected", 0x000099);</pre>

Setting a style with ActionScript overrides the parameter of the same name set in the Property inspector or Component inspector.

DynamicButton class

Inheritance MovieClip > UIObject class > UIComponent class > DynamicButton class

Setting a property of the DynamicButton class with ActionScript overrides the parameter of the same name set in the Property inspector or Component inspector. The DynamicButton component uses the Focus Manager to override the default Flash Player focus rectangle and draw a custom focus rectangle with rounded corners.

Method summary for the DynamicButton class

There are no methods exclusive to the DynamicButton class.

Methods inherited from the UIObject class

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

Method	Description
UIObject.destroyObject()	Destroys a component instance.
UIObject.getStyle()	Gets the style property from the style declaration or object.
UIObject.move()	Moves the object to the requested position.
UIObject.setSize()	Resizes the object to the requested size.
UIObject.setStyle()	Sets the style property on the style declaration or object.

Methods inherited from the UIComponent class

The following table lists the methods the DynamicButton class inherits from the UIComponent class.

Method	Description
UIComponent.getFocus()	Returns a reference to the object that has focus.
UIComponent.setFocus()	Sets focus to the component instance.

Property summary for the DynamicButton class

The following table lists properties of the DynamicButton class.

Property	Description
DynamicButton.animationEffect	The animation effect applied to a dynamic button.
DynamicButton.animationIconSet	A set of prebuilt icons applied to a dynamic button.
DynamicButton.customIcons	An object that contains the linkage identifiers for custom icons from the library.
DynamicButton.embedFonts	A Boolean value indicating whether the font specified in <code>fontFamily</code> is an embedded font.
DynamicButton.fontFamily	The font name for text of a dynamic button label.
DynamicButton.fontSize	The point size for the font of a dynamic button label.
DynamicButton.fontStyle	The font style for the font of a dynamic button label.

DynamicButton.fontNormalStyle	An object that contains a set of styles for the label when a dynamic button is unselected.
DynamicButton.fontOverStyle	An object that contains a set of styles for the label when a dynamic button is rolled over.
DynamicButton.fontSelectedStyle	An object that contains a set of styles for the label when a dynamic button is selected.
DynamicButton.iconColor	The color of the icon when a dynamic button is unselected.
DynamicButton.iconColorSelected	The color of the icon when a dynamic button is selected.
DynamicButton.label	The text that appears next to the icon of a dynamic button.
DynamicButton.labelPlacement	The orientation of the label text in relation to the icon of a dynamic button.
DynamicButton.selected	A Boolean value indicating whether the dynamic button is selected (true) or not (false).
DynamicButton.sound	An object that contains the linkage identifier(s) for the sound(s) from the library to be attached to a dynamic button.
DynamicButton.toggle	A Boolean value indicating whether a dynamic button is in a toggle switch.
DynamicButton.URL	An object that contains parameters of a document from a specific URL to be load into a window when a dynamic button is clicked.

Properties inherited from the UIObject class

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

Property	Description
UIObject.bottom	The position of the bottom edge of the object, relative to the bottom edge of its parent. Read-only.
UIObject.height	The height of the object, in pixels. Read-only.
UIObject.left	The left edge of the object, in pixels. Read-only.
UIObject.right	The position of the right edge of the object, relative to the right edge of its parent. Read-only.
UIObject.scaleX	A number indicating the scaling factor in the x direction of the object, relative to its parent.
UIObject.scaleY	A number indicating the scaling factor in the y direction of the object, relative to its parent.
UIObject.top	The position of the top edge of the object, relative to its parent. Read-only.
UIObject.visible	A Boolean value indicating whether the object is visible (true) or not (false).
UIObject.width	The width of the object, in pixels. Read-only.
UIObject.x	The left edge of the object, in pixels. Read-only.

UIObject.y

The top edge of the object, in pixels. Read-only.

Properties inherited from the UIComponent class

The following table lists the properties the DynamicButton class inherits from the UIComponent class.

Property	Description
UIComponent.enabled	Indicates whether the component can receive focus and input.
UIComponent.tabIndex	A number indicating the tab order for a component in a document.

Event summary for the DynamicButton class

The following table lists the event of the DynamicButton class.

Event	Description
DynamicButton.click	Triggered when the mouse is clicked over a dynamic button.
DynamicButton.rollover	Triggered when the mouse is rolled over a dynamic button.
DynamicButton.rollout	Triggered when the mouse is rolled out of a dynamic button.

Events inherited from the UIObject class

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

Event	Description
UIObject.draw	Broadcast when an object is about to draw its graphics.
UIObject.hide	Broadcast when an object's state changes from visible to invisible.
UIObject.load	Broadcast when subobjects are being created.
UIObject.move	Broadcast when the object has moved.
UIObject.resize	Broadcast when an object has been resized.
UIObject.reveal	Broadcast when an object's state changes from invisible to visible.
UIObject.unload	Broadcast when the subobjects are being unloaded.

Events inherited from the UIComponent class

The following table lists the events the DynamicButton class inherits from the UIComponent class.

Event	Description
UIObject.focusIn	Broadcast when an object receives focus.
UIObject.focusOut	Broadcast when an object loses focus.
UIObject.keyDown	Broadcast when a key is pressed.
UIObject.keyUp	Broadcast when a key is released.

DynamicButton.animationEffect

Usage

```
DynamicButtonInstance.animationEffect
```

Description

Property; sets the type of animation effect applied to a dynamic button. You can use this property to get or set the animation effect for a dynamic button instance. Calling this method overrides the animationEffect parameter value set during authoring. This property can be one of five values: "Circulation", "Heartbeat", "Pulsar", "Rotation" or "Vibration"; the default value is "Circulation".

Example

The following example sets the animationEffect property of the instance dynamicButton to "Rotation":

```
dynamicButton.animationEffect = "Rotation";  
trace(dynamicButton.animationEffect);
```

DynamicButton.animationIconSet

Usage

```
DynamicButtonInstance.animationIconSet
```

Description

Property; specifies the set of icons (for selected and unselected state) applied to a dynamic button. You can use this property to get or set the animationIconSet parameter for a dynamic button instance. Calling this method overrides the animationIconSet parameter value set during authoring. This property can be one of five values: "set1", "set2", "set3", "set4" or "set5"; the default value is "set1".

Example

The following example sets the animationIconSet property of the instance dynamicButton to "set3":

```
dynamicButton.animationIconSet = "set3";
```

DynamicButton.click

Usage

Usage 1:

```
on(click) {  
    ...  
}
```

Usage 2:

```
listenerObject = new Object();  
listenerObject.click = function(eventObject) {  
    ...  
}  
DynamicButtonInstance.addEventListener("click", listenerObject);
```

Description

Event; broadcast to all registered listeners when the mouse is clicked (pressed and released) over the dynamic button.

The first usage example uses an `on()` handler and must be attached directly to a `DynamicButton` 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 the dynamic button `dynButton`, sends “_level0.dynButton” to the Output panel:

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

The second usage example uses a dispatcher/listener event model. A component instance (*DynamicButtonInstance*) 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.

For more information, see “EventDispatcher class” in *Macromedia Flash MX 2004 Using Components Help*.

Example

This example, written on a frame of the Timeline, sends a message to the Output panel when the instance `dynamicButton` is clicked. The first line of code creates a listener object called `buttonListener`. The second line defines a function for the `click` event on the listener object. Inside the function is a `trace()` statement that uses the event object (`eventObj`) that is automatically passed to the function to generate a message. The `target` property of an event object is the component that generated the event. You can access instance properties from the `target` property (in this example, the `DynamicButton.label` property is accessed). The last line calls `EventDispatcher.addEventListener()` from the `dynamicButton` instance and passes it the `click` event and the `buttonListener` object as parameters.

```
buttonListener = new Object();  
buttonListener.click = function(eventObj) {  
    trace("The selected instance is " + eventObj.target.label);  
}  
dynamicButton.addEventListener("click", buttonListener);
```

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

```
on(click) {  
    trace("dynamic button was clicked");  
}
```

DynamicButton.customIcons

Usage

Usage 1 (set property):

```
DynamicButtonInstance.setStyle("customIcons", {falseIcon: String, trueIcon: String});
```

Usage 2 (set property):

```
customIconsObject = new Object();  
customIconsObject.falseIcon = String;
```

```
customIconsObject.trueIcon = String;  
DynamicButtonInstance.setStyle("customIcons", customIconsObject);
```

Usage 3 (get property):

```
customIconsObject = new Object(DynamicButtonInstance.customIcons);  
var1:String = customIconsObject["falseIcon"];  
var2:String = customIconsObject["trueIcon"];
```

Parameters

falseIcon A string that is the linkage identifier of a symbol in the library; this symbol is used as an icon that designates unselected state of a dynamic button. The default value is undefined.

trueIcon A string that is the linkage identifier of a symbol in the library; this symbol is used as an icon that designates selected state of a dynamic button. The default value is undefined.

Description

Property; an object that contains the linkage identifiers of movie clips or graphic symbols in the library for custom icons. You can use this property to get or set custom icons for a dynamic button instance. Calling this method overrides the customIcons parameter value set during authoring.

Example

The following example sets the customIcons property of the instances dynamicButton1, dynamicButton2 and dynamicButton3 to "item_false" and "item_true":

```
dynamicButton1.setStyle("customIcons", {falseIcon: "item_false", trueIcon: "item_true"});  
dynamicButton2.setStyle("customIcons", {falseIcon: "item_false", trueIcon: "item_true"});  
dynamicButton3.setStyle("customIcons", {falseIcon: "item_false", trueIcon: "item_true"});
```

or

```
customIconsObject = new Object();  
customIconsObject.falseIcon = "item_false";  
customIconsObject.trueIcon = "item_true";  
dynamicButton1.setStyle("customIcons", customIconsObject);  
dynamicButton2.setStyle("customIcons", customIconsObject);  
dynamicButton3.setStyle("customIcons", customIconsObject);
```

You can access the customIcons property values of dynamicButton1 instance by writing the following code:

```
customIconsObject = new Object(dynamicButton1.customIcons);  
var1 = customIconsObject["falseIcon"];  
var2 = customIconsObject["trueIcon"];  
trace("false icon: " + var1 + ", true icon: " + var2);
```

DynamicButton.embedFonts

Usage

```
DynamicButtonInstance.embedFonts
```

Description

Property; a Boolean value that indicates whether the font specified in fontFamily is an embedded font. This property must be set to true if fontFamily refers to an embedded font. Otherwise, the embedded font will not be used. If this style is set to true and fontFamily does not refer to an embedded font, no text will be displayed. The default value is false.

Example

The following example sets the embedFonts property of the instance dynamicButton to true:
dynamicButton.embedFonts = true;

DynamicButton.fontFamily

Usage

DynamicButtonInstance.fontFamily

Description

Property; sets the font name for text of a dynamic button label. You can use this property to get or set the font name for the label of a dynamic button instance. Calling this method overrides the fontFamily parameter value set during authoring. The default value is "_sans".

Example

The following example sets the fontFamily property of the instance dynamicButton to "Century":
dynamicButton.fontFamily = "Century";

DynamicButton.fontSize

Usage

DynamicButtonInstance.fontSize

Description

Property; sets the point size for the font of a dynamic button label. You can use this property to get or set the font size for the label of a dynamic button instance. Calling this method overrides the fontSize parameter value set during authoring. The default value is 11.

Example

The following example sets the fontSize property of the instance dynamicButton to 12:
dynamicButton.fontSize = 12;

DynamicButton.fontStyle

Usage

DynamicButtonInstance.fontStyle

Description

Property; sets the font style for text of a dynamic button label. You can use this property to get or set the font style for the label of a dynamic button instance. Calling this method overrides the fontStyle parameter value set during authoring. This property can be either "normal" or "italic"; the default value is "normal".

Example

The following example sets the fontStyle property of the instance dynamicButton to "italic":
dynamicButton.fontStyle = "italic";

DynamicButton.fontNormalStyle

Usage

Usage 1 (set property):

```
DynamicButtonInstance.setStyle("fontNormalStyle", {[fontColor: Color[, fontWeight: String[, textDecoration: String[, backgroundColor: Color[, borderColor: Color[, fontColorDisabled: Color]]]]]]});
```

Usage 2 (get property):

```
fontNormalStyleObject = new Object(DynamicButtonInstance.fontNormalStyle);  
var1:Color = fontNormalStyleObject["fontColor"];  
var2:String = fontNormalStyleObject["fontWeight"];
```

```

var3:String = fontNormalStyleObject["textDecoration"];
var4:Color = fontNormalStyleObject["backgroundColor"];
var5:Color = fontNormalStyleObject["borderColor"];
var6:Color = fontNormalStyleObject["fontColorDisabled"];

```

Parameters

<i>fontColor</i>	The text color. The default value is 0x000000 (black).
<i>fontWeight</i>	The font weight: either "none" or "bold". The default value is "none".
<i>textDecoration</i>	The text decoration: either "none" or "underline". The default value is "none".
<i>backgroundColor</i>	The color of the text field background. The default value is undefined.
<i>borderColor</i>	The color of the text field border. The default value is undefined.
<i>fontColorDisabled</i>	The text color when the component is disabled. The default color is undefined.

Description

Property; an object that contains a set of styles for the label when a dynamic button is unselected. You can use this property to get or set the values of text field styles for the label of a dynamic button instance. Calling this method overrides the `fontNormalStyle` parameter value set during authoring.

Example

The following example sets `fontColor` and `textDecoration` parameters of the `fontNormalStyle` property for the instances `dynamicButton1`, `dynamicButton2` and `dynamicButton3`:

```

dynamicButton1.setStyle("fontNormalStyle", {fontColor: 0x336633, textDecoration: "underline"});
dynamicButton2.setStyle("fontNormalStyle", {fontColor: 0x336633, textDecoration: "underline"});
dynamicButton3.setStyle("fontNormalStyle", {fontColor: 0x336633, textDecoration: "underline"});

```

You can access the `fontNormalStyle` property values of `dynamicButton1` instance by writing the following code:

```

fontNormalStyleObject = new Object(dynamicButton1.fontNormalStyle);
var1 = fontNormalStyleObject["fontColor"];
var2 = fontNormalStyleObject["textDecoration"];
trace("font color: " + var1 + ", text decoration: " + var2);

```

DynamicButton.fontOverStyle

Usage

Usage 1 (set property):

```

DynamicButtonInstance.setStyle("fontOverStyle", {[fontColor: Color[, fontWeight: String[, textDecoration: String[, text: String[, backgroundColor: Color[, borderColor: Color]]]]]]]);

```

Usage 2 (set property):

```

fontOverStyleObject = new Object();
fontOverStyleObject.fontColor = Color;
fontOverStyleObject.fontWeight = String;
fontOverStyleObject.textDecoration = String;
fontOverStyleObject.text = String;
fontOverStyleObject.backgroundColor = Color;
fontOverStyleObject.borderColor = Color;
DynamicButtonInstance.setStyle("fontOverStyle", fontOverStyleObject);

```

Usage 3 (get property):

```

fontOverStyleObject = new Object(DynamicButtonInstance.fontOverStyle);
var1:Color = fontOverStyleObject["fontColor"];
var2:String = fontOverStyleObject["fontWeight"];
var3:String = fontOverStyleObject["textDecoration"];
var4:String = fontOverStyleObject["text"];

```

```
var5:Color = fontOverStyleObject["backgroundColor"];
var6:Color = fontOverStyleObject["borderColor"];
```

Parameters

<i>fontColor</i>	The text color. The default value is 0x000000 (black).
<i>fontWeight</i>	The font weight: either "none" or "bold". The default value is "none".
<i>textDecoration</i>	The text decoration: either "none" or "underline". The default value is "none".
<i>text</i>	The value of the text on the dynamic button. The default value is undefined.
<i>backgroundColor</i>	The color of the text field background. The default value is undefined.
<i>borderColor</i>	The color of the text field border. The default value is undefined.

Description

Property; an object that contains a set of styles for the label when a dynamic button is rolled over. You can use this property to get or set the values of text field styles for the label of a dynamic button instance. Calling this method overrides the `fontOverStyle` parameter value set during authoring.

Example

The following example sets `fontColor`, `fontWeight`, `text`, `backgroundColor` and `boderColor` parameters of the `fontOverStyle` property for the instances `dynamicButton1`, `dynamicButton2` and `dynamicButton3`:

```
dynamicButton1.setStyle("fontOverStyle", {fontColor: 0x3366CC, fontWeight: "bold", backgroundColor: 0xFFFFFFFF, borderColor: 0x3366CC, text: "click"});
dynamicButton2.setStyle("fontOverStyle", {fontColor: 0x3366CC, fontWeight: "bold", backgroundColor: 0xFFFFFFFF, borderColor: 0x3366CC, text: "click"});
dynamicButton3.setStyle("fontOverStyle", {fontColor: 0x3366CC, fontWeight: "bold", backgroundColor: 0xFFFFFFFF, borderColor: 0x3366CC, text: "click"});
```

or

```
fontOverStyleObject = new Object();
fontOverStyleObject.fontColor = 0x3366CC;
fontOverStyleObject.fontWeight = "bold";
fontOverStyleObject.backgroundColor = 0xFFFFFFFF;
fontOverStyleObject.borderColor = 0x3366CC;
fontOverStyleObject.text = "click";
dynamicButton1.setStyle("fontOverStyle", fontOverStyleObject);
dynamicButton2.setStyle("fontOverStyle ", fontOverStyleObject);
dynamicButton3.setStyle("fontOverStyle ", fontOverStyleObject);
```

You can access the `fontOverStyle` property values of `dynamicButton1` instance by writing the following code:

```
fontOverStyleObject = new Object(dynamicButton1.fontOverStyle);
var1 = fontOverStyleObject["fontColor"];
var2 = fontOverStyleObject["fontWeight"];
var3 = fontOverStyleObject["backgroundColor"];
var4 = fontOverStyleObject["borderColor"];
var5 = fontOverStyleObject["text"];
trace("font color: " + var1 + ", font weight: " + var2 + ", background color: " + var3 + ", border color: " + var4 + ", text: " + var5);
```

DynamicButton.fontSelectedStyle

Usage

Usage 1 (set property):

```
DynamicButtonInstance.setStyle("fontSelectedStyle", {[fontColor: Color[, fontWeight: String[, textDecoration: String[, text: String[, backgroundColor: Color[, borderColor: Color]]]]]]]);
```

Usage 2 (set property):

```
fontSelectedStyleObject = new Object();
fontSelectedStyleObject.fontColor = Color;
fontSelectedStyleObject.fontWeight = String;
fontSelectedStyleObject.textDecoration = String;
fontSelectedStyleObject.text = String;
fontSelectedStyleObject.backgroundColor = Color;
fontSelectedStyleObject.borderColor = Color;
DynamicButtonInstance.setStyle("fontSelectedStyle", fontSelectedStyleObject);
```

Usage 3 (get property):

```
fontSelectedStyleObject = new Object(DynamicButtonInstance.fontSelectedStyle);
var1:Color = fontSelectedStyleObject["fontColor"];
var2:String = fontSelectedStyleObject["fontWeight"];
var3:String = fontSelectedStyleObject["textDecoration"];
var4:String = fontSelectedStyleObject["text"];
var5:Color = fontSelectedStyleObject["backgroundColor"];
var6:Color = fontSelectedStyleObject["borderColor"];
```

Parameters

<i>fontColor</i>	The text color. The default value is 0x000000 (black).
<i>fontWeight</i>	The font weight: either "none" or "bold". The default value is "none".
<i>textDecoration</i>	The text decoration: either "none" or "underline". The default value is "none".
<i>text</i>	The value of the text on the dynamic button. The default value is undefined.
<i>backgroundColor</i>	The color of the text field background. The default value is undefined.
<i>borderColor</i>	The color of the text field border. The default value is undefined.

Description

Property; an object that contains a set of styles for the label when a dynamic button is selected. You can use this property to get or set the values of text field styles for the label of a dynamic button instance. Calling this method overrides the `fontSelectedStyle` parameter value set during authoring.

Example

The following example sets `fontColor`, `fontWeight`, `text`, `backgroundColor` and `boderColor` parameters of the `fontSelectedStyle` property for the instances `dynamicButton1`, `dynamicButton2` and `dynamicButton3`:

```
dynamicButton1.setStyle("fontSelectedStyle", {fontColor: 0x3366CC, fontWeight: "bold", backgroundColor: 0xFFFF, borderColor: 0x3366CC, text: "click"});
dynamicButton2.setStyle("fontSelectedStyle", {fontColor: 0x3366CC, fontWeight: "bold", backgroundColor: 0xFFFF, borderColor: 0x3366CC, text: "click"});
dynamicButton3.setStyle("fontSelectedStyle", {fontColor: 0x3366CC, fontWeight: "bold", backgroundColor: 0xFFFF, borderColor: 0x3366CC, text: "click"});
```

or

```
fontSelectedStyleObject = new Object();
fontSelectedStyleObject.fontColor = 0x3366CC;
fontSelectedStyleObject.fontWeight = "bold";
fontSelectedStyleObject.backgroundColor = 0xFFFF;
fontSelectedStyleObject.borderColor = 0x3366CC;
fontSelectedStyleObject.text = "click";
dynamicButton1.setStyle("fontSelectedStyle", fontSelectedStyleObject);
dynamicButton2.setStyle("fontSelectedStyle", fontSelectedStyleObject);
dynamicButton3.setStyle("fontSelectedStyle", fontSelectedStyleObject);
```

You can access the `fontSelectedStyle` property values of `dynamicButton1` instance by writing the following code:

```
fontSelectedStyleObject = new Object(dynamicButton1. fontSelectedStyle);
var1 = fontSelectedStyleObject["fontColor"];
var2 = fontSelectedStyleObject["fontWeight"];
var3 = fontSelectedStyleObject["backgroundColor"];
var4 = fontSelectedStyleObject["borderColor"];
var5 = fontSelectedStyleObject["text"];
trace("font color: " + var1 + ", font weight: " + var2 + ", background color: " + var3 + ", border color: " +
var4 + ", text: " + var5);
```

DynamicButton.iconColor

Usage

DynamicButtonInstance.iconColor

Description

Property; sets the color of icon when a dynamic button is unselected. You can use this property to get or set the color of icon for a dynamic button instance. Calling this method overrides the `iconColor` parameter value set during authoring. The default value is `0xFF6600` (orange).

Example

The following example sets the `iconColor` property of the instance `dynamicButton` to `0x0066CC`:

```
dynamicButton.iconColor = 0x0066CC;
```

DynamicButton.iconColorSelected

Usage

DynamicButtonInstance.iconColorSelected

Description

Property; sets the color of icon when a dynamic button is selected. You can use this property to get or set the color of icon for a dynamic button instance. Calling this method overrides the `iconColorSelected` parameter value set during authoring. The default value is `0xFF6600` (orange).

Example

The following example sets the `iconColorSelected` property of the instance `dynamicButton` to `0x000099`:

```
dynamicButton.iconColorSelected = 0x000099;
```

DynamicButton.label

Usage

DynamicButtonInstance.label

Description

Property; specifies the text label for a dynamic button instance. You can use this property to get or set the text label for a dynamic button instance. By default, the label appears center-aligned on the dynamic button. Calling this method overrides the label authoring parameter specified in the Property inspector or the Component inspector. The default value is "DynamicButton".

Example

The following code sets the label of the instance `dynamicButton` to "Bicycles":

```
dynamicButton.label = "Bicycles";
```

DynamicButton.labelPlacement

Usage

DynamicButtonInstance.labelPlacement

Description

Property; sets the position of the label in relation to the icon of a dynamic button. The default value is "bottom". The following are the three possible values:

- *right* The label is set to the right of the icon.
- *left* The label is set to the left of the icon.
- *bottom* The label is set below the icon.

Example

The following code sets the label of the instance *dynamicButton* to the left of the icon. The second line of the code sends the value of the *labelPlacement* property to the Output panel:

```
dynamicButton.labelPlacement = "left";  
trace(dynamicButton.labelPlacement);
```

DynamicButton.rollover

Usage

Usage 1:

```
on(rollover) {  
    ...  
}
```

Usage 2:

```
listenerObject = new Object();  
listenerObject.rollover = function(eventObject) {  
    ...  
}  
DynamicButtonInstance.addEventListener("rollover", listenerObject);
```

Description

Event; broadcast to all registered listeners when the mouse pointer rolls over the dynamic button.

The first usage example uses an `on()` handler and must be attached directly to a *DynamicButton* 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 the dynamic button `dynButton`, sends "`_level0.dynButton`" to the Output panel:

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

The second usage example uses a dispatcher/listener event model. A component instance (*DynamicButtonInstance*) dispatches an event (in this case, `rollover`) 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.

For more information, see “EventDispatcher class” in *Macromedia Flash MX 2004 Using Components Help*.

Example

This example, written on a frame of the Timeline, sends a message to the Output panel when the instance `dynamicButton` is rolled over. The first line of code creates a listener object called `buttonListener`. The second line defines a function for the rollover event on the listener object. Inside the function is a `trace()` statement that uses the event object (`eventObj`) that is automatically passed to the function to generate a message. The target property of an event object is the component that generated the event. You can access instance properties from the target property (in this example, the `DynamicButton.label` property is accessed). The last line calls `EventDispatcher.addEventListener()` from the `dynamicButton` instance and passes it the rollover event and the `buttonListener` object as parameters.

```
buttonListener = new Object();
buttonListener.rollover = function(eventObj) {
    trace("The rolled over instance is " + eventObj.target.label);
}
dynamicButton.addEventListener("rollover", buttonListener);
```

The following code also sends a message to the Output panel when a dynamic button is rolled over. The `on()` handler must be attached directly to a dynamic button instance.

```
on(rollover) {
    trace("dynamic button was rolled over");
}
```

DynamicButton.rollout

Usage

Usage 1:

```
on(rollout) {
    ...
}
```

Usage 2:

```
listenerObject = new Object();
listenerObject.rollout = function(eventObject) {
    ...
}
DynamicButtonInstance.addEventListener("rollout", listenerObject);
```

Description

Event; broadcast to all registered listeners when the mouse pointer rolls out of the dynamic button.

The first usage example uses an `on()` handler and must be attached directly to a `DynamicButton` 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 the dynamic button `dynButton`, sends “`_level0.dynButton`” to the Output panel:

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

The second usage example uses a dispatcher/listener event model. A component instance (*DynamicButtonInstance*) dispatches an event (in this case, rollout) 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.

For more information, see “EventDispatcher class” in *Macromedia Flash MX 2004 Using Components Help*.

Example

This example, written on a frame of the Timeline, sends a message to the Output panel when the instance `dynamicButton` is rolled out. The first line of code creates a listener object called `buttonListener`. The second line defines a function for the rollout event on the listener object. Inside the function is a `trace()` statement that uses the event object (`eventObj`) that is automatically passed to the function to generate a message. The target property of an event object is the component that generated the event. You can access instance properties from the target property (in this example, the `DynamicButton.label` property is accessed). The last line calls `EventDispatcher.addEventListener()` from the `dynamicButton` instance and passes it the rollout event and the `buttonListener` object as parameters.

```
buttonListener = new Object();
buttonListener.rollout = function(eventObj) {
    trace("The rolled out instance is " + eventObj.target.label);
}
dynamicButton.addEventListener("rollout", buttonListener);
```

The following code also sends a message to the Output panel when a dynamic button is rolled out. The `on()` handler must be attached directly to a dynamic button instance.

```
on(rollout) {
    trace("dynamic button was rolled out");
}
```

DynamicButton.selected

Usage

DynamicButtonInstance.selected

Description

Property; a Boolean value that sets the state of a dynamic button to selected (true) or unselected (false). You can use this property to get or set the state of a dynamic button instance. Calling this method overrides the selected parameter value set during authoring. The default value is false.

Example

The first line of code sets the selected property of the `dynamicButton` instance to true. The second line of code returns the value of the selected property:

```
dynamicButton.selected = true;
trace(dynamicButton.selected);
```

DynamicButton.sound

Usage

Usage 1 (set property):

```
DynamicButtonInstance.setStyle("sound", {rollover_sound: String, release_sound: String});
```

Usage 2 (set property):

```
soundObject = new Object();  
soundObject.rollover_sound = String;  
soundObject.release_sound = String;  
DynamicButtonInstance.setStyle("sound", soundObject);
```

Usage 3 (get property):

```
soundObject = new Object(DynamicButtonInstance.sound);  
var1:String = soundObject["rollover_sound"];  
var2:String = soundObject["release_sound"];
```

Parameters

rollover_sound A string that is the linkage identifier of a sound file in the library; this parameter attaches the sound to a dynamic button when rolled over. The default value is undefined.

release_sound A string that is the linkage identifier of a sound file in the library; this parameter attaches the sound to a dynamic button when released. The default value is undefined.

Description

Property; an object that contains the linkage identifier(s) of a sound file (sound files) in the library. This parameter attaches the sound(s) specified in the Values dialog box to a dynamic button when it is rolled over or released. The sound(s) must be in the library of the current SWF file and specified for export in the Linkage Properties dialog box. You can use this property to get or set linkages identifier(s) of sound(s) attached to a dynamic button instance. Calling this method overrides the sound parameter value set during authoring.

Example

The following example sets the sound property values of the instance dynamicButton to "squeak" and "crash":

```
dynamicButton.setStyle("sound", {rollover_sound: "squeak", release_sound: "crash"});
```

or

```
soundObject = new Object();  
soundObject.rollover_sound = "squeak";  
soundObject.release_sound = "crash";  
dynamicButton.setStyle("sound", soundObject);
```

You can access the sound property values of dynamicButton instance by writing the following code:

```
soundObject = new Object(dynamicButton.sound);  
var1 = soundObject["rollover_sound"];  
var2 = soundObject["release_sound"];  
trace("Rollover sound: " + var1 + ", release sound: " + var2);
```

DynamicButton.toggle

Usage

```
DynamicButtonInstance.toggle
```

Description

Property; a Boolean value that turns a dynamic button into a toggle switch. If true, the state of the dynamic button changes (from selected to unselected and vice versa) when clicked. If false, the dynamic button behaves like a normal push button. Calling this method overrides the toggle parameter value set during authoring. The default value is false.

Example

The following example sets the toggle property of the dynamicButton instance to true:

```
dynamicButton.toggle = true;
```

DynamicButton.URL

Usage

Usage 1 (set property):

```
DynamicButtonInstance.setStyle("URL", {url: String, window: String, delay: Number});
```

Usage 2 (set property):

```
URLObject = new Object();  
URLObject.url = String;  
URLObject.window = String;  
URLObject.delay = Number;  
DynamicButtonInstance.setStyle("URL", URLObject);
```

Usage 3 (get property):

```
URLObject = new Object(DynamicButtonInstance.URL);  
var1:String = URLObject["url"];  
var2:String = URLObject["window"];  
var3:Number = URLObject["delay"];
```

Parameters

url A string that is the URL from which to obtain the document; the default value is undefined.

window A string that specifies the window or HTML frame into which the document should load. You can enter the name of a specific window or select from the following reserved target names (the default value is "_blank"):

- *_self* The current frame in the current window.
- *_blank* A new window
- *_parent* The parent of the current frame.
- *_top* The top-level frame in the current window.

delay A number that sets the time delay, in milliseconds, between onClick event and the beginning of a specified URL loading. The default value is 500.

Description

Property; an object that contains parameters of a document from a specific URL to be load into a window when a dynamic button is clicked. Calling this method overrides the URL parameter value set during authoring.

Example

The following example sets the URL property values of the instance dynamicButton:

```
dynamicButton.setStyle("URL", {url: "http://www.macromedia.com", window: "_self", delay: 1000});
```

or

```
URLObject = new Object();
```

```
URLObject.url = "http://www.macromedia.com";  
URLObject.window = "_self";  
URLObject.delay = 1000;  
dynamicButton.setStyle("URL", URLObject);
```

You can access the URL property values of dynamicButton instance by writing the following code:

```
URLObject = new Object(dynamicButton.URL);  
var1 = URLObject["url"];  
var2 = URLObject["window"];  
var3 = URLObject["delay"];  
trace("URL: " + var1 + ", window: " + var2 + ", delay: " + var3);
```

DynamicRadioButton component

The DynamicRadioButton component lets you force a user to make a single choice within a set of choices. This component must be used in a group of at least two DynamicRadioButton instances. Only one member of the group can be selected at any given time. Selecting one dynamic radio button in a group deselects the currently selected dynamic radio button in the group. You set the groupName parameter to indicate which group a dynamic radio button belongs to.

A dynamic radio button can be enabled or disabled. A disabled dynamic radio button doesn't receive mouse or keyboard input. When the user clicks or tabs into a DynamicRadioButton component group, only the selected dynamic radio button receives focus. The user can then use the following keys control it:

Key	Description
Up Arrow / Right Arrow	The selection moves to the previous dynamic radio button within the dynamic radio button group.
Down Arrow / Left Arrow	The selection moves to the next dynamic radio button within the dynamic radio button group.
Tab	Moves focus from the dynamic radio button group to the next component.

For more information about controlling focus, see "Creating custom focus navigation" or "FocusManager class" in *Macromedia Flash MX 2004 Using Components Help*.

A live preview of each DynamicRadioButton instance on the Stage reflects changes made to parameters in the Property inspector or Component inspector during authoring. However, the mutual exclusion of selection does not display in the live preview. If you set the selected parameter to true for two dynamic radio buttons in the same group, they both appear selected even though only the last instance created will appear selected at runtime.

If a dynamic radio button is enabled, it displays its rollover state when a user moves the pointer over it. When a user clicks an unselected dynamic radio button, the dynamic radio button retains its rollover unselected state. When a user releases the mouse, the dynamic radio button displays its rollover selected state and the previously selected dynamic radio button in the group returns to its unselected state. If a user moves the pointer off a dynamic radio button while pressing the mouse, the dynamic radio button loses rollover dynamic state but it retains its current static state (selected or unselected).

If a dynamic radio button or dynamic radio button group is disabled, it displays its disabled state, regardless of user interaction.

Using the DynamicRadioButton component

The DynamicRadioButton is a universal component that combines its individual properties with the properties of Macromedia RadioButton component. As a result, there are two common uses for DynamicRadioButton:

1. Creating the main animated menu for a Flash application that consists of three and more items. Both horizontal and vertical arrangement of menu items (icons) is possible.
2. Creating the form that forces a user to make a single choice within a set of choices (similarly to Macromedia RadioButton).

The first use of the component is preferred because it was originally designed for this purpose.

DynamicRadioButton parameters

You can set the following authoring parameters for each DynamicRadioButton component instance in the Component inspector:

animationEffect sets the type of animation effect applied to the dynamic radio button. This parameter can be one of five values: Circulation, Heartbeat, Pulsar, Rotation or Vibration; the default value is Circulation. For more information, see [DynamicMenuButtons animation](#).

animationIconSet specifies the set of icons (for selected and unselected state) applied to the dynamic radio button. This parameter can be one of five values: set1, set2, set3, set4 or set5; the default value is set1. For more information, see [DynamicMenuButtons animation](#).

customIcons is an object type optional parameter that contains two values to be set: falseIcon and trueIcon. This parameter adds custom icons to the dynamic radio button. The values are the linkage identifiers of movie clips or graphic symbols in the library. The default values are undefined.

Note: Setting the custom icons disables the action of animationIconSet, iconColor and iconColorSelected parameters. For more information, see [DynamicMenuButtons animation](#).

groupName is the group name of the dynamic radio button. The default value is menuGroup.

label sets the value of the text on the dynamic radio button; the default value is DynRadioButton.

labelPlacement orients the label text on the dynamic radio button in relation to the icon. This parameter can be one of three values: left, right or bottom; the default value is bottom. For more information, see [DynamicButton.labelPlacement](#).

selected sets the initial value of the dynamic radio button to selected (true) or unselected (false). A selected dynamic radio button differs from unselected by the icon view. Only one dynamic radio button in a group can have a selected value of true. If more than one dynamic radio button in a group is set to true, the dynamic radio button that is instantiated last is selected. The default value is false.

sound is an object type optional parameter that contains two values to be set: rollover_sound and release_sound. This parameter attaches the sound(s) specified in the Values dialog box to the dynamic radio button when rolled over (rollover_sound) or released (release_sound). The sound(s) must be in the library of the current SWF file and specified for export in the Linkage Properties dialog box. The default values are undefined.

URL is an object type optional parameter that contains three values to be set: url, window, delay. This parameter specifies a document from a specific URL to be load into a window when the dynamic radio button is clicked.

url. The URL from which to obtain the document. The default value is undefined.

window. (Optional) Specifies the window or HTML frame into which the document should load. You can enter the name of a specific window or select from the following reserved target names: `_self` (the current frame in the current window), `_blank` (a new window), `_parent` (the parent of the current frame), `_top` (the top-level frame in the current window). The default value is `_blank`.

delay. Sets the time delay, in milliseconds, between onClick event and the beginning of a specified URL loading. The default value is 500.

To test this function, make sure the file to be loaded is at the specified location.

For other parameters that customize the color and text of DynamicRadioButton component, see [Customizing the DynamicRadioButton component](#).

You can write ActionScript to set additional options for DynamicRadioButton instances using the methods, properties and events of the DynamicRadioButton class. For more information, see [DynamicRadioButton class](#).

Creating an application with the DynamicRadioButton component

The following procedure explains how to add DynamicRadioButton components to an application while authoring.

Example 1

In this example, the dynamic radio buttons are used to create the main animated menu for a Flash application.

To create an application with the DynamicRadioButton component:

1. Set the Stage size to 500 x 300 pixels and leave white as background color. For frame rate enter any appropriate value – for example, 30 fps.
2. Drag one DynamicRadioButton component from the Components panel to the Stage and set its coordinates to (10, 35).

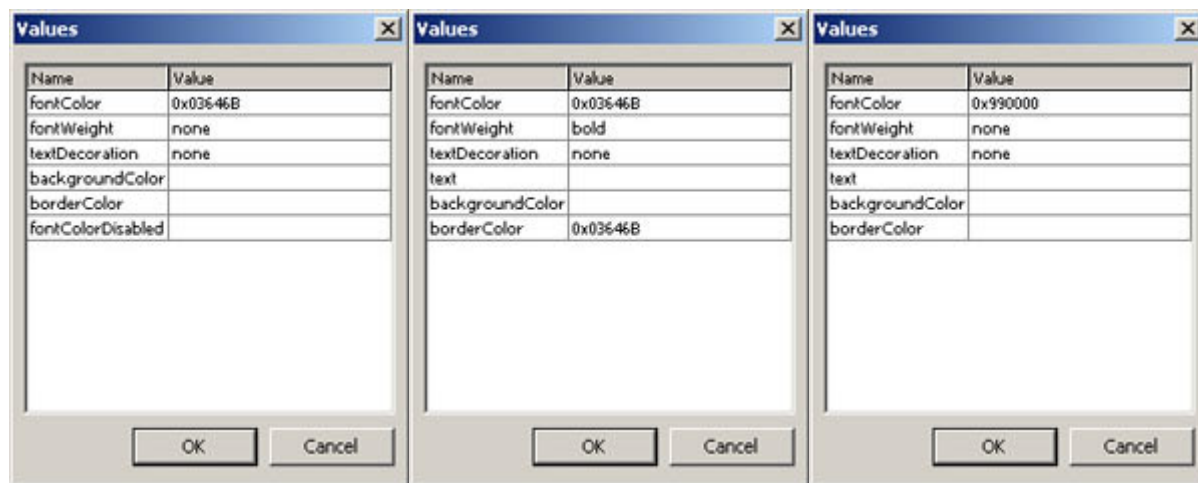
Note: Although the default size for a DynamicRadioButton instance is 100 x 70 pixels, the visual area actually increases vertically up to 150% while effect duration.

3. In the Component inspector, do the following:

For the icon color parameters, set:

iconColor #669999
 iconColorSelected #FF9900

For the fontNormalStyle, fontOverStyle and fontSelectedStyle parameters as the objects, set the following values (from left to right accordingly):



Leave the other parameters without changes.

4. Make three copies of the current DynamicRadioButton instance and set their coordinates to (110, 35), (210, 35) and (310, 35) accordingly.
5. Set the Label parameter for the component instances accordingly to "Home", "Products", "Support", "Contact".

6. Select one of the dynamic radio buttons on the Stage – for example, "Home" – and set the selected parameter to true.
7. Select Control > Test Movie.

Example 2

In this example, the dynamic radio buttons are used to create the main Flash menu as SWF file embedded in HTML pages.

To create an application with the DynamicRadioButton component:

1. Set the Stage size to 500 x 300 pixels and leave white as background color. For frame rate enter any appropriate value – for example, 30 fps.
2. Drag one DynamicRadioButton component from the Components panel to the Stage and set its coordinates to (10, 40).
3. In the Component inspector, do the following:

For the animationEffect parameter, set Circulation
 For the animationIconSet parameter, set.....set2
 For the fontNormalStyle parameters, set:
 fontColor..... 0x000000
 For the fontOverStyle and fontSelectedStyle parameters, set:
 fontColor..... 0x990000
 textDecoration..... underline
 For the icon color parameters, set:
 iconColor #FF6600
 iconColorSelected #666666
 For the groupName parameter, set..... topMenu
 For the URL parameters, set:
 window _self
 delay..... 800

Leave the other parameters without changes.

4. Make three copies of the current DynamicRadioButton instance and set their coordinates to (110, 40), (210, 40) and (310, 40) accordingly.
5. Set the Label parameter for the component instances accordingly to "Home", "Documentation", "Projects", "Archive".
6. In the Property inspector, enter values m1, m2, m3 and m4 for the Instance Name property of the component instances.
7. Select one of the dynamic radio buttons on the Stage – for example, "Home" – and set the selected parameter to true.
8. For the URL parameters in the Component inspector, set the following values of the url property:
 for m1 instance home.html
 for m2 instance documentation.html
 for m3 instance projects.html
 for m4 instance archive.html
9. Place the following code in the first frame of the movie:

```
topMenu.selection = eval(menu_item);
```

10. Publish the Flash document with appropriate name – for example, top_menu.swf.
11. Create four HTML pages with the file names: home.html, documentation.html, projects.html and archive.html and place them to the same folder that contains top_menu.swf.
12. Place the following HTML-code sample into home.html to embed SWF file (in our case, top_menu.swf):

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,0,0"
width="500" height="300" id="top_menu" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value=" top_menu.swf?menu_item=m1" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src=" top_menu.swf?menu_item=m1" quality="high" bgcolor="#ffffff" width="500" height="300"
name=" top_menu" align="middle" allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

For the files documentation.html, projects.html and archive.html, the values of menu_item variable must be m2, m3 and m4 accordingly.

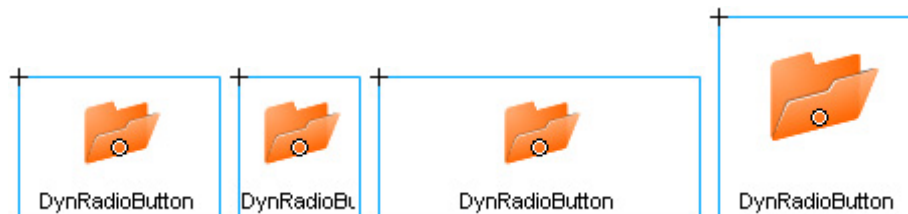
13. Open one of created HTML files in a web browser and test the behavior of the DynamicRadioButton component.

Customizing the DynamicRadioButton component

You can transform a DynamicRadioButton component horizontally and vertically while authoring and at runtime. While authoring, select the component on the Stage and use the Free Transform tool or any of the Modify > Transform commands. At runtime, use the setSize() method.

The bounding box of a DynamicRadioButton component is invisible and also designates the hit area for the component. If you increase the size of the component, you also increase the size of the hit area. The default value for a component's size is 100 x 70 pixels. If the component's bounding box is too small to fit the component label, the label is clipped to fit.

Transforming a component instance vertically results in proportional change of an icon scale. Transforming a component instance horizontally reduces or increases its width and does not resize an icon. Change the component's width to fit the component label if necessary. Modifying the component does not change the height of a label text. For this purpose, you should use the `fontSize` parameter while authoring or `DynamicButton.fontSize` property at runtime.



All the styles used by the DynamicRadioButton component are inherited from the DynamicButton class. For information about using styles with the DynamicRadioButton component, see [Customizing the DynamicButton component](#).

DynamicRadioButton class

Inheritance MovieClip > UIObject class > UIComponent class > DynamicButton class > DynamicRadioButton class

Setting a property of the DynamicRadioButton class with ActionScript overrides the parameter of the same name set in the Property inspector or Component inspector. The DynamicRadioButton component uses the Focus Manager to override the default Flash Player focus rectangle and draw a custom focus rectangle with rounded corners.

Method summary for the DynamicRadioButton class

There are no methods exclusive to the DynamicRadioButton class.

Methods inherited from the UIObject class

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

Method	Description
UIObject.destroyObject()	Destroys a component instance.
UIObject.getStyle()	Gets the style property from the style declaration or object.
UIObject.move()	Moves the object to the requested position.
UIObject.setSize()	Resizes the object to the requested size.
UIObject.setStyle()	Sets the style property on the style declaration or object.

Methods inherited from the UIComponent class

The following table lists the methods the DynamicRadioButton class inherits from the UIComponent class.

Method	Description
UIComponent.getFocus()	Returns a reference to the object that has focus.
UIComponent.setFocus()	Sets focus to the component instance.

Property summary for the DynamicRadioButton class

The following table lists properties of the DynamicRadioButton class.

Property	Description
DynamicRadioButton.groupName	The group name for a dynamic radio button instance.
DynamicRadioButton.selected	Selects the dynamic radio button, and deselects the previously selected dynamic radio button.
DynamicRadioButton.selection	A reference to the currently selected dynamic radio button in a radio button group.

Properties inherited from the DynamicButton class

The following table lists the properties the DynamicRadioButton class inherits from the DynamicButton class. When accessing these properties from the DynamicRadioButton object, use the form *DynamicRadioButtonInstance.propertyName*.

Property	Description
DynamicButton.animationEffect	The animation effect applied to a dynamic button.
DynamicButton.animationIconSet	A set of prebuilt icons applied to a dynamic button.
DynamicButton.customIcons	An object that contains the linkage identifiers for custom icons from the library.
DynamicButton.embedFonts	A Boolean value indicating whether the font specified in <code>fontFamily</code> is an embedded font.
DynamicButton.fontFamily	The font name for text of a dynamic button label.
DynamicButton.fontSize	The point size for the font of a dynamic button label.
DynamicButton.fontStyle	The font style for the font of a dynamic button label.
DynamicButton.fontNormalStyle	An object that contains a set of styles for the label when a dynamic button is unselected.
DynamicButton.fontOverStyle	An object that contains a set of styles for the label when a dynamic button is rolled over.
DynamicButton.fontSelectedStyle	An object that contains a set of styles for the label when a dynamic button is selected.
DynamicButton.iconColor	The color of the icon when a dynamic button is unselected.
DynamicButton.iconColorSelected	The color of the icon when a dynamic button is selected.
DynamicButton.label	The text that appears next to the icon of a dynamic button.
DynamicButton.labelPlacement	The orientation of the label text in relation to the icon of a dynamic button.
DynamicButton.sound	An object that contains the linkage identifier(s) for the sound(s) from the library to be attached to a dynamic button.
DynamicButton.URL	An object that contains parameters of a document from a specific URL to be load into a window when a dynamic button is clicked.

Properties inherited from the UIObject class

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

Property	Description
<code>UIObject.bottom</code>	The position of the bottom edge of the object, relative to the bottom edge of its parent. Read-only.
<code>UIObject.height</code>	The height of the object, in pixels. Read-only.
<code>UIObject.left</code>	The left edge of the object, in pixels. Read-only.

UIObject.right	The position of the right edge of the object, relative to the right edge of its parent. Read-only.
UIObject.scaleX	A number indicating the scaling factor in the x direction of the object, relative to its parent.
UIObject.scaleY	A number indicating the scaling factor in the y direction of the object, relative to its parent.
UIObject.top	The position of the top edge of the object, relative to its parent. Read-only.
UIObject.visible	A Boolean value indicating whether the object is visible (true) or not (false).
UIObject.width	The width of the object, in pixels. Read-only.
UIObject.x	The left edge of the object, in pixels. Read-only.
UIObject.y	The top edge of the object, in pixels. Read-only.

Properties inherited from the UIComponent class

The following table lists the properties the DynamicRadioButton class inherits from the UIComponent class.

Property	Description
UIComponent.enabled	Indicates whether the component can receive focus and input.
UIComponent.tabIndex	A number indicating the tab order for a component in a document.

Event summary for the DynamicRadioButton class

The following table lists the event of the DynamicRadioButton class.

Event	Description
DynamicRadioButton.click	Triggered when the mouse is clicked over a dynamic radio button or dynamic radio button group.
DynamicRadioButton.rollover	Triggered when the mouse is rolled over a dynamic radio button or dynamic radio button group.
DynamicRadioButton.rollout	Triggered when the mouse is rolled out of a dynamic radio button or dynamic radio button group.

Events inherited from the UIObject class

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

Event	Description
UIObject.draw	Broadcast when an object is about to draw its graphics.
UIObject.hide	Broadcast when an object's state changes from visible to invisible.
UIObject.load	Broadcast when subobjects are being created.

UIObject.move	Broadcast when the object has moved.
UIObject.resize	Broadcast when an object has been resized.
UIObject.reveal	Broadcast when an object's state changes from invisible to visible.
UIObject.unload	Broadcast when the subobjects are being unloaded.

Events inherited from the UIComponent class

The following table lists the events the DynamicRadioButton class inherits from the UIComponent class.

Event	Description
UIObject.focusIn	Broadcast when an object receives focus.
UIObject.focusOut	Broadcast when an object loses focus.
UIObject.keyDown	Broadcast when a key is pressed.
UIObject.keyUp	Broadcast when a key is released.

DynamicRadioButton.click

Usage

Usage 1:

```
on(click) {
    ...
}
```

Usage 2:

```
listenerObject = new Object();
listenerObject.click = function(eventObject) {
    ...
}
DynamicRadioButtonGroup.addEventListener("click", listenerObject);
```

Description

Event; broadcast to all registered listeners when the mouse is clicked (pressed and released) over the dynamic radio button or if the dynamic radio button is selected by means of the arrow keys.

The first usage example uses an `on()` handler and must be attached directly to a `DynamicRadioButton` 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 the dynamic radio button `dynRadioButton`, sends “`_level0.dynRadioButton`” to the Output panel:

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

The second usage example uses a dispatcher/listener event model. A component instance (*DynamicRadioButtonInstance*) 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.

For more information, see “EventDispatcher class” in *Macromedia Flash MX 2004 Using Components Help*.

Example

This example, written on a frame of the Timeline, sends a message to the Output panel when a dynamic radio button in `menuGroup` is clicked. The first line of code creates a listener object called `menuListener`. The second line defines a function for the click event on the listener object. Inside the function is a `trace()` statement that uses the event object (`eventObj`) that is automatically passed to the function to generate a message. The target property of an event object is the component that generated the event. You can access instance properties from the target property (in this example, the `DynamicRadioButton.selection` property is accessed). The last line calls `EventDispatcher.addEventListener()` from `menuGroup` and passes it the click event and the `menuListener` object as parameters.

```
menuListener = new Object();
menuListener.click = function(eventObj) {
    trace("The selected instance is " + eventObj.target.selection);
}
menuGroup.addEventListener("click", menuListener);
```

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

```
on(click) {
    trace("dynamic radio button was clicked");
}
```

DynamicRadioButton.groupName

Usage

```
DynamicRadioButtonInstance.groupName
DynamicRadioButtonsGroup.groupName
```

Description

Property; sets the group name for a dynamic radio button instance or group. You can use this property to get or set a group name for a dynamic radio button instance or for a dynamic radio button group. Calling this method overrides the `groupName` parameter value set during authoring. The default value is “menuGroup”.

Example

The following example sets the `groupName` of the instances `dynamicRadioButton1`, `dynamicRadioButton2` and `dynamicRadioButton3` to “colorChoice”:

```
dynamicRadioButton1.groupName = "colorChoice";
dynamicRadioButton2.groupName = "colorChoice";
dynamicRadioButton3.groupName = "colorChoice";
```

DynamicRadioButton.rollover

Usage

```
Usage 1:
on(rollover) {
```

```
    ...  
}
```

Usage 2:

```
listenerObject = new Object();  
listenerObject.rollover = function(eventObject) {  
    ...  
}  
DynamicRadioButtonInstance.addEventListener("rollover", listenerObject);
```

Description

Event; broadcast to all registered listeners when the mouse pointer rolls over the dynamic radio button or if the dynamic radio button is selected by means of the arrow keys.

The first usage example uses an `on()` handler and must be attached directly to a `DynamicRadioButton` 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 the dynamic radio button `dynRadioButton`, sends “_level0.dynRadioButton” to the Output panel:

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

The second usage example uses a dispatcher/listener event model. A component instance (*DynamicRadioButtonInstance*) dispatches an event (in this case, `rollover`) 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.

For more information, see “EventDispatcher class” in *Macromedia Flash MX 2004 Using Components Help*.

Example

This example, written on a frame of the Timeline, sends a message to the Output panel when one of the dynamic radio buttons in the group is rolled over. The first line of code creates a listener object called `menuListener`. The second line defines a function for the `rollover` event on the listener object. Inside the function is a `trace()` statement that uses the event object (`eventObj`) that is automatically passed to the function to generate a message. The target property of an event object is the component that generated the event. You can access instance properties from the target property (in this example, the `DynamicButton.label` property is accessed). The last line calls `EventDispatcher.addEventListener()` from the component instance and passes it the `rollover` event and the `menuListener` object as parameters.

```
menuListener = new Object();  
menuListener.rollover = function(eventObj) {  
    trace("The rolled over menu item is " + eventObj.target.label);  
}  
menuItem1.addEventListener("rollover", menuListener);  
menuItem2.addEventListener("rollover", menuListener);  
menuItem3.addEventListener("rollover", menuListener);  
menuItem4.addEventListener("rollover", menuListener);
```

The following code also sends a message to the Output panel when a dynamic radio button is rolled over. The `on()` handler must be attached directly to a dynamic radio button instance.

```
on(rollover) {
    trace("dynamic radio button was rolled over");
}
```

DynamicRadioButton.rollout

Usage

Usage 1:

```
on(rollout) {
    ...
}
```

Usage 2:

```
listenerObject = new Object();
listenerObject.rollout = function(eventObject) {
    ...
}
DynamicRadioButtonInstance.addEventListener("rollout", listenerObject);
```

Description

Event; broadcast to all registered listeners when the mouse pointer rolls out of the dynamic radio button or if the dynamic radio button is deselected by means of the arrow keys.

The first usage example uses an `on()` handler and must be attached directly to a `DynamicRadioButton` 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 the dynamic radio button `dynRadioButton`, sends “`_level0.dynRadioButton`” to the Output panel:

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

The second usage example uses a dispatcher/listener event model. A component instance (*DynamicRadioButtonInstance*) dispatches an event (in this case, `rollout`) 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.

For more information, see “`EventDispatcher` class” in *Macromedia Flash MX 2004 Using Components Help*.

Example

This example, written on a frame of the Timeline, sends a message to the Output panel when one of the dynamic radio buttons in the group is rolled out. The first line of code creates a listener object called `menuListener`. The second line defines a function for the `rollout` event on the listener object. Inside the function is a `trace()` statement that uses the event object (`eventObj`) that is automatically passed to the function to generate a message. The `target` property of an event object is the component that generated the event. You can access instance properties from the `target` property (in

this example, the `DynamicButton.label` property is accessed). The last line calls `EventDispatcher.addEventListener()` from the component instance and passes it the rollout event and the `menuListener` object as parameters.

```
menuListener = new Object();
menuListener.rollout = function(eventObj) {
    trace("The rolled out menu item is " + eventObj.target.label);
}
menuItem1.addEventListener("rollout", menuListener);
menuItem2.addEventListener("rollout", menuListener);
menuItem3.addEventListener("rollout", menuListener);
menuItem4.addEventListener("rollout", menuListener);
```

The following code also sends a message to the Output panel when a dynamic radio button is rolled out. The `on()` handler must be attached directly to a dynamic radio button instance.

```
on(rollout) {
    trace("dynamic radio button was rolled out");
}
```

DynamicRadioButton.selected

Usage

DynamicRadioButtonInstance.selected

Description

Property; a Boolean value that sets the state of the dynamic radio button to selected (true) and deselects the previously selected radio button, or sets the radio button to unselected (false). You can use this property to get or set the state of a dynamic radio button instance. Calling this method overrides the selected parameter value set during authoring.

Example

The following example sets the selected property of the instance `dynamicRadioButton` to true:

```
dynamicRadioButton.selected = true;
```

DynamicRadioButton.selection

Usage

DynamicRadioButtonInstance.selection
DynamicRadioButtonGroup.selection

Description

Property; behaves differently depending on whether you get or set the property. If you get the property, it returns the object reference of the currently selected dynamic radio button in a dynamic radio button group. If you set the property, it selects the specified dynamic radio button (passed as an object reference) in a dynamic radio button group and deselects the previously selected dynamic radio button.

Example

The following example selects the dynamic radio button with the instance name `menu_item1` and sends its instance name to the Output panel:

```
menuGroup.selection = menu_item1;
trace(menuGroup.selection._name);
```

DynamicMenuButtons animation

The DynamicRadioButton component and the DynamicButton component use the same animation; therefore, all of the animation control parameters and properties that described in this chapter equally refer to both components.





















You can control the animation of DynamicRadioButton or DynamicButton components through the following three parameters in the Property inspector or in the Component inspector (see [DynamicRadioButton parameters](#) or [DynamicButton parameters](#)):

animationEffect defines the type of prebuilt dynamic effect. This parameter is applied when using both prebuilt and custom icons;



animationIconSet defines one of five prebuilt icon sets. This parameter becomes inactive if the user specifies the custom icons through the customIcons parameter;

customIcons is an optional parameter that allows you to choose two custom icons (that designate selected and unselected states of a dynamic button or a dynamic radio button) in the library and apply them to the component instance. Setting this parameter disables the action of animationIconSet parameter.














Each of five dynamic effects belongs to one of two groups: centric and non-centric. Division into groups is defined by dynamics of movement during the animation. For each of these groups of effects there are five sets (pairs) of icons displayed in the table below.

animationEffect	animationIconSet					icon state
	set1	set2	set3	set4	set5	
<i>Non-centric effects:</i> Circulation Heartbeat Vibration						falseIcon
						trueIcon
<i>Centric effects:</i> Pulsar Rotation						falseIcon
						trueIcon

Bear in mind the following tips when using some prebuilt sets of icons:

animationIconSet	Samples	Tips
set3 (non-centric group)		To make these icons look natural choose the shades of red when setting iconColor or iconColorSelected parameters.
set2 (centric group)		These icons look best on a dark background.

All prebuilt sets of icons are developed in such a way that can be applied to any dynamic effect in corresponding group (non-centric or centric). Nevertheless, certain sets of icons were originally designed for specific effects. These combinations therefore are preferred to use. They are displayed in the following table.

animationEffect	Preferred animation icon sets (in decreasing order)	
	names of sets	icons
Circulation	set1 ⇒ set2	 
Heartbeat	set3 ⇒ set1 ⇒ set2	  
Vibration	set4 ⇒ set5 ⇒ set3	  
Pulsar	set1	
Rotation	set3 ⇒ set4 ⇒ set2 ⇒ set5	   

Each of five dynamic effects has its own parameters for icon location and its individual zone of scaling. Therefore, keep in mind these features when creating your custom icons for the specified dynamic effect. The following table contents the maximum and recommended sizes for a custom icon depending on specified effect. This will help you in designing icons for the dynamic effect you want to use.

animationEffect	Height, pixels		Width, pixels	
	max	recommended	max	recommended
Circulation	31	≤ 31	36	≤ 32
Heartbeat	34	≤ 31	46	≤ 36
Vibration	33	≤ 31	46	≤ 36
Pulsar	40	≤ 31	70	≤ 54
Rotation	33	≤ 27	44	≤ 34

Note: 1) The location of a symbol registration point when creating your custom icons must be: upper left corner for *Non-centric effects* and central for *Centric effects*. 2) Both of two custom icons (falseIcon and trueIcon) must be of the same size.

Bear in mind the following tips when using some prebuilt dynamic effects.

animationEffect	Tips
Circulation	<p>Although using custom icons disables the action of the <code>iconColor</code> and <code>iconColorSelected</code> parameters, it does not apply to the prebuilt animation item (rings) for this effect. Thus, you can specify the color of this item by setting the <code>iconColor</code> and <code>iconColorSelected</code> parameters for a component instance.</p>
Pulsar	<p>Because this effect belongs to the centric group of <code>DynamicMenuButtons</code> animation effects, it is better to use centric round-shaped icons when specifying the custom icons. Also note that you should not use the white background for this effect. Otherwise, the effect becomes invisible.</p>
Rotation	<p>Because this effect belongs to the centric group of <code>DynamicMenuButtons</code> animation effects, it is better to use centric round-shaped icons when specifying the custom icons. Also, using custom icons with this effect you can tune the rotation speed by setting the <code>animationIconSet</code> parameter to <code>set1</code>, <code>set2</code> or <code>set3</code>.</p>