

ModuleMenuNode

Inherits From:	Object
Declared In:	"ModuleMenuNode.h"

Class Description

This object stores menu information to be passed to Resound from modules. Future versions will eliminate this node (I hope), when IB is able to generate menus without an application object present.

The ModuleMenuNode is a tree node. The root (top) node of the tree is your Module's *TheModuleMenuNode*, and should be set with a blank Name. This node contains all the menu items to be added directly to Resound's Modules menu. Each menu item is represented by a child ModuleMenuNode with a name. If a child is meant to be a simple menu item, it's given a target receiver (typically your module) and a selector. If a child is meant to be a submenu, it's not given a receiver or selector, but is instead given children nodes.

Let's say you want to make two menu options in the Modules menu. One is just a standard menu option, called "Tweak". The other displays a submenu, called "Twist", with submenu items "Bend", "Break", and "Bust". You'd generate a tree as a collection of nodes that would look something like this:

(Root Node)

Name	""	<i>empty string</i>
Receiver	NULL	<i>indicates this node has submenus</i>
Message		<i>not set</i>
Length	2	<i>for Tweak and Twist</i>
Submenu[0]		points to the node Tweak
Submenu[1]		points to the node Twist

Tweak

Name	"Tweak"
Receiver	points to the object that will receive the Tweak message
Message	selector to the message sent to Tweak's receiver
Length	0
Submenu...	<i>not set</i>

Twist

Name	"Twist"	
Receiver	NULL	<i>indicates this node has submenus</i>
Message		<i>not set</i>
Length	3	<i>for Bend, Break, and Bust</i>
Submenu[0]		points to the node Bend
Submenu[1]		points to the node Break
Submenu[2]		points to the node Bust

Bend

Name	"Bend"
Receiver	points to the object that will receive the Bend message
Message	selector to the message sent to Bend's receiver
Length	0
Submenu...	<i>not set</i>

Break

Name	"Break"
Receiver	points to the object that will receive the Break message
Message	selector to the message sent to Break's receiver
Length	0
Submenu...	<i>not set</i>

Bust

Name	"Bust"
Receiver	points to the object that will receive the Bust message
Message	selector to the message sent to Bust's receiver
Length	0
Submenu...	<i>not set</i>

Code for this would be in the **init** method of your module, and might look like:

```
- init
{
    id returnVal=[super init];    // ABSOLUTELY CALL [super init] FIRST!!!

    ModuleMenuNode* tweak=[[ModuleMenuNode alloc] init];
    ModuleMenuNode* twist=[[ModuleMenuNode alloc] init];
    ModuleMenuNode* bend=[[ModuleMenuNode alloc] init];
    ModuleMenuNode* break=[[ModuleMenuNode alloc] init];
    ModuleMenuNode* bust=[[ModuleMenuNode alloc] init];

    [TheModuleMenuNode setReceiver:NULL];
    [TheModuleMenuNode setLength:2];
    [TheModuleMenuNode setSubmenu: 0 : tweak];
    [TheModuleMenuNode setSubmenu: 1 : twist];

    [tweak setName: "Tweak"];
```

```

[tweak setReceiver: self];    // module is target of menu's action method...
[tweak setMessage:@selector(tweakMe:)];
[tweak setLength:0];

[twist setReceiver:NULL];
[twist setLength:3];
[twist setSubmenu: 0 : bend];
[twist setSubmenu: 1 : break];
[twist setSubmenu: 1 : bust];

[bend setName: "Bend"];
[bend setReceiver: self];    // module is target of menu's action method...
[bend setMessage:@selector(bendMe:)];
[bend setLength:0];

[break setName: "Break"];
[break setReceiver: self];    // module is target of menu's action method...
[break setMessage:@selector(breakMe:)];
[break setLength:0];

[bust setName: "Bust"];
[bust setReceiver: self];    // module is target of menu's action method...
[bust setMessage:@selector(bustMe:)];
[bust setLength:0];

return returnVal;
}

```

As a result, Resound would create menus in its Modules menu as such:

paste.tiff ↪

Resound attaches the appropriate receiver to as the target of Tweak, Bend, Break, and Bust, and sets them to call the appropriate message. In the example above, the receivers of all menus have been set to your own module, and the modules will call, respectively, **tweakMe:**, **bendMe:**, **breakMe:**, or **bustMe:**.

You don't need to free the nodes—that's taken care of for you by Module's **free** method.

What if I don't want a menu option at all?

Good question. In this case, you don't need to do anything special at all in your init method (other than calling **[super init]** first). As long as TheModuleMenuNode's length has been set to 0 (which it's initialized to), it won't look for submenu nodes to add.

Defined Constants

MODULE_MENU_NODE_ARRAY_MAX	256
MODULE_MENU_NODE_STRING_MAX	256

Instance Variables

char	Name [MODULE_MENU_NODE_STRING_MAX];
id	Receiver ;
SEL	Message ;
int	Length ;
id	Submenu [MODULE_MENU_NODE_ARRAY_MAX];

Name	Name to be put in the menu. If Name is empty, then menu is the list of initial menu choices. TheModuleMenuNode should <i>not</i> have a name, but all of its submenu nodes <i>should</i> have a name.
------	---

Receiver	Receiver of the method. If Receiver is NULL, the menu is assumed to have submenus.
----------	--

Message	The selector to be sent to Receiver.
---------	--------------------------------------

Length	Number of submenus. If Length is 0, then there are no submenus. If Receiver is NULL, then Length should <i>not</i> be 0.
Submenu	The menu's submenus, if any.

Method Types

Creating and freeing instances	- init ± free
Querying the object	- getName - getReceiver - setMessage - getLength - getSubmenu:
Setting data	- setName: - setReceiver: - setMessage: - setLength: - setSubmenu::

Instance Methods

free
- free

Frees the ModuleMenuNode and all of its subnodes.

getLength

- (int) **getLength**

Returns the number of submenus of the ModuleMenuNode.

getMessage

- (SEL) **getMessage**

Returns the ModuleMenuNode's message selector.

getName

- (const char*) **getName**

Returns the ModuleMenuNode's name, "" if none.

getReceiver

- **getReceiver**

Returns the ModuleMenuNode's receiver.

init

- **init**

Initializes the ModuleMenuNode.

setLength:

- **setLength:** (int)*this_length*

Sets the number of submenus of the ModuleMenuNode.

setMessage:

- **setMessage:** (SEL)*this_message*

Sets the message sent to the ModuleMenuNode's receiver.

setName:

- **setName:** (const char*)*this_name*

Sets the name of the ModuleMenuNode.

setReceiver:

- **setReceiver:** *this_receiver*

Sets the ModuleMenuNode's receiver.

setSubmenu:

- **setSubmenu:** (int) *this_index* : *this_node*

Sets submenu # *this_index* of the ModuleMenuNode to *this_node*. *this_node* should be a child ModuleMenuNode.