NSFontManager

Inherits From: NSObject

Conforms To: NSObject (NSObject)

Declared In: AppKit/NSFontManager.h

Class Description

NSFontManager is the center of activity for the font conversion system. It records the currently selected font, updates the Font Panel and Font menu to reflect the selected font, initiates font changes, and converts fonts in response to requests from text-bearing objects. In a more prosaic role, NSFontManager can be queried for the fonts available to the application, and for the particular attributes of a font, such as whether it's condensed or extended.

You normally set up a font manager and the Font Menu using Interface Builder. However, you can also do so programmatically by getting the shared font manager instance and having it create the standard Font menu at run time:

```
NSFontManager *fontManager = [NSFontManager sharedFontManager];
NSMenu *fontMenu = [fontManager fontMenu:YES];
```

You can then add the Font menu to your application's main menu. Once the Font menu is installed, your application automatically gains the functionality of both the Font menu and the Font Panel.

Recording the Selected Font

Any object that records fonts that the user can change should tell the font manager what the font of its selection is whenever it becomes the first responder and whenever its selection changes while it's the first responder. The object does so by sending the shared font manager a **setSelectedFont:isMultiple:** message. It should pass in the first font of the selection, along with a flag indicating whether there's more than one font.

The font manager uses this information to update the Font Panel and Font menu to reflect the selected font. For example, suppose the selected font is set as Helvetica Oblique 12.0 point. In this case the Font Panel selects that font and displays its name; the Font menu changes its Italic command to Unitalic; if there's no Bold variant of Helvetica available, the Bold menu item is disabled; and so on.

Initiating Font Changes

The user normally changes the font of the selection by manipulating the Font Panel and Font menu. These objects initiate the intended change by sending an action message to the font manager. There are four font-changing action methods:

– addFontTrait:
– removeFontTrait:
– modifyFont:
– modifyFontViaPanel:

The first three cause the font manager to query the sender of the message in order to determine which trait to add or remove, or how to modify the font. The last causes the font manager to use the settings in the Font Panel to modify the font. The font manager records this information and uses it in later requests to convert fonts, as described under "Responding to Font Changes."

When the font manager receives an **addFontTrait:** or **removeFontTrait:** message, it queries the sender with a **tag** message, interpreting the return value as a trait mask for use with **convertFont:toHaveTrait:** or **convertFont:toNotHaveTrait:**, as described below under "Converting Fonts Manually." The Italic and Bold Font menu commands, for example, have tags of NSItalicFontMask and NSBoldFontMask, respectively. See **convertFont:toHaveTrait:** for a list of trait mask values.

When the font manager receives a **modifyFont:** message, it queries the sender with a **tag** message and interprets the return value as a particular kind of conversion to perform, via the various conversion methods described under "Converting Fonts Manually." For example, a button whose tag value is NSSizeUpFontAction causes the font manager's **convertFont:** method to increase the size of the NSFont passed as the argument. See **modifyFont:** for a list of conversion tag values.

For **modifyFontViaPanel:**, the font manager sends the application's Font Panel a **panelConvertFont:** message. The Font Panel in turn uses the font manager to convert the font provided according to the user's choices. For example, if the user selects only the font family in the Font Panel (perhaps to Helvetica), then whatever fonts are provided to **panelConvertFont:**, only the family is changed: Courier Medium 10.0 point becomes Helvetica Medium 10.0 point, while Times Italic 12.0 point becomes Helvetica Oblique 12.0 point.

Responding to Font Changes

The font manager responds to a font-changing action method by sending a **changeFont:** action message up the responder chain. A text-bearing object that receives this message should have the font manager convert the fonts in its selection by invoking **convertFont:** for each font and using the NSFont object returned. **convertFont:** uses the information recorded by the font-changing action method, such as **addFontTrait:**, modifying the font provided appropriately. (There's no way to explicitly set the font-changing action or trait; instead, you use the methods described under "Converting Fonts Manually.")

This simple example assumes there's only one font in the selection:

```
- (void)changeFont:(id)sender
{
    NSFont *oldFont = [self selectionFont];
    NSFont *newFont = [sender convertFont:oldFont];
    [self setSelectionFont:newFont];
    return;
}
```

Most text-bearing objects will have to scan the selection for ranges with different fonts, and invoke **convertFont:** for each one.

Font Trait Masks

NSFontManager categorizes fonts according to a small set of traits. You can convert fonts by adding and removing individual traits, and you can get a font with a specific combination of traits. The traits defined and available for your use are:

NSItalicFontMask	NSCondensedFontMask
NSBoldFontMask	NSExpandedFontMask
NSNarrowFontMask	NSCompressedFontMask
NSFixedPitchFontMask	NSSmallCapsFontMask
NSPosterFontMask	

NSCondensedFontMask and NSExpandedFontMask are mutually exclusive. In addition to these, NSFontManager defines the masks NSUnitalicFontMask, NSUnboldFontMask, and NSNonStandardCharacterSetFontMask for its own internal use. Though they're defined in the header file, you shouldn't use these masks.

Converting Fonts Manually

NSFontManager defines a number of methods for explicitly converting particular traits and characteristics of a font. These methods are:

- convertFont:toFace:
- convertFont:toFamily:
- convertFont:toHaveTrait:
- convertFont:toNotHaveTrait:
- convertFont:toSize:
- convertWeight:ofFont:

Each returns a transformed version of the font provided, or the original font if it can't be converted. **convertFont:toFace:** and **convertFont:toFamily:** both alter the basic design of the font provided. The first method requires a fully-specified typeface name, such as "Times-Roman" or "Helvetica-BoldOblique", while the second expects only a family name, such as "Times" or "Helvetica".

convertFont:toHaveTrait: and **convertFont:toNotHaveTrait:** use trait masks to add or remove a single trait such as Italic, Bold, Condensed, or Extended.

convertFont:toSize: returns a font of the requested size, with all other characteristics the same as those of the original font.

convertWeight:ofFont: either increases or decreases the weight of the font provided, according to a boolean flag. Font weights are typically indicated by a series of names, which can vary from font to font. Some go from Light to Medium to Bold, while others have Book, SemiBold, Bold, and Black. This method offers a uniform way of incrementing and decrementing any font's weight.

The default implementation of font conversion is very conservative, making a change only if no other trait or aspect is affected. For example, if you try to convert Helvetica Oblique 12.0 point by adding the Bold trait, and only Helvetica Bold is available, the font isn't converted. You can create a subclass of NSFontManager and override the conversion methods to perform less conservative conversion, perhaps using Helvetica Bold in this case and losing the Oblique trait.

In addition to the font-conversion methods, NSFontManager defines **fontWithFamily:traits:weight:size:** to construct a font with a given set of characteristics. If you don't care to make a subclass of NSFontManager, you can use this method to perform approximate font conversions yourself.

Examining Fonts

In addition to converting fonts, NSFontManager provides information on which fonts are available to the application, and on the characteristics of any given font. **availableFonts** returns an array of the names of all fonts available. **availableFontNamesWithTraits:** filters the available fonts based on a font trait mask.

There are three methods for examining individual fonts. **fontNamed:hasTraits:** returns YES if the font matches the trait mask provided. **traitsOfFont:** returns a trait mask for a given font. **weightOfFont:** returns an approximate ranking of a font's weight on a scale of 0–15, where 0 is the lightest possible weight, 5 is Normal or Book weight, 9 is the equivalent of Bold, and 15 is the heaviest possible (often called Black or Ultra Black).

Customizing the Font Conversion System

If you need to customize the font conversion system by creating subclasses of NSFontManager or NSFontPanel, you must inform the NSFontManager class of this change with a **setFontManagerFactory:** or **setFontPanelFactory:** message, before either the shared font manager or shared font panel is created. These methods record your class as the one to instantiate the first time the font manager or Font Panel is requested.

You may be able to avoid using subclasses if all you need is to add some custom controls to the Font Panel. In this case, you can invoke NSFontPanel's **setAccessoryView:** method to add an NSView below its font browser. This allows you to add controls for such things as font color, kerning, and so on.

If you provide your own Font menu, you should register it with the font manager using the **setFontMenu:** method. The font manager is responsible for validating Font menu items and changing their titles and tags according to the selected font. For example, when the selected font is Italic the font manager changes the Italic Font menu item to Unitalic, and changes its tag to NSUnitalicFontMask. Your Font menu's items should use the appropriate action methods and tags. Here are some examples:

Font Menu Item	Action	Tag
Italic	**addFontTrait:**	NSItalicFontMask
Bold	**addFontTrait:**	NSBoldFontMask
Heavier	**modifyFont:**	NSHeavierFontAction
Larger	**modifyFont:**	NSSizeUpFontAction

Normally, the application's Font Panel displays all the standard fonts available on the system. If this isn't appropriate for your application—for example, if only fixed-pitch fonts should be used—you can assign a delegate to the font manager object to filter the available fonts. Before the Font Panel displays a particular font family or face, it asks the font manager to confirm it. The font manager in turn queries its delegate with a **fontManager:willIncludeFont:** message. If the delegate returns YES (or doesn't implement this method), the font is displayed in the Font Panel. If the delegate returns NO, the font isn't listed.

Method Types

Getting the shared font manager	+ sharedFontManager
Changing the default font conversion classes	+ setFontManagerFactory: + setFontPanelFactory:
Getting available fonts	– availableFonts – availableFontNamesWithTraits:
Setting and examining the selected font	– setSelectedFont:isMultiple: – selectedFont – isMultiple – sendAction
Action methods	– addFontTrait: – removeFontTrait: – modifyFont: – modifyFontViaPanel:
Converting fonts automatically	– convertFont:

Converting fonts manually	– convertFont:toFace: – convertFont:toFamily: – convertFont:toHaveTrait: – convertFont:toNotHaveTrait: – convertFont:toSize: – convertWeight:ofFont:
Getting a particular font	– fontWithFamily:traits:weight:size:
Examining fonts	– traitsOfFont: – fontNamed:hasTraits: – weightOfFont:
Enabling the Font Panel and Font menu	– setEnabled: – isEnabled
Setting the Font menu	– setFontMenu: – fontMenu:
Getting the Font Panel	– fontPanel: – orderFrontFontPanel:
Setting the delegate	– setDelegate: – delegate
Setting the action method	– setAction: – action

Class Methods

setFontManagerFactory:

+ (void)**setFontManagerFactory:**(Class)*aClass*

Sets the class object used to create the font manager to *aClass*, which should be a subclass of NSFontManager. When the NSFontManager class object receives a **sharedFontManager** message, it creates an instance of *aClass*, if no instance already exists. Your font manager class should implement **init** as its designated initializer. The default font manager factory is NSFontManager.

This method must be invoked before your application's main nib file is loaded, such as in the application delegate's **applicationWillFinishLaunching:** method.

See also: + **setFontPanelFactory:**

setFontPanelFactory:

+ (void)**setFontPanelFactory:**(Class)*factoryId*

Sets the class used to create the Font Panel to *aClass*, which should be a subclass of NSFontPanel. Invoke this method before accessing the Font Panel in any wa, such as in the application delegate's **applicationWillFinishLaunching:** method.

See also: + **setFontManagerFactory:**

sharedFontManager

+ (NSFontManager *)**sharedFontManager**

Returns the singleton instance of the font manager factory for the application, creating it if necessary.

See also: + **setFontManagerFactory:**

Instance Methods

action

– (SEL)**action**

Returns the action that's sent to the first responder when the user selects a new font from the Font panel or chooses a command from the Font menu. The default action is **changeFont:**.

See also: – **setAction:**

addFontTrait:

– (void)**addFontTrait:**(id)*sender*

This action method causes the receiver to send its action message (**changeFont:** by default) up the responder chain. When a responder replies by providing a font to convert in a **convertFont:** message, the receiver converts the font by adding the trait specified by *sender*. This trait is determined by sending a **tag** message to *sender* and interpreting it as a font trait mask for a **convertFont:toHaveTrait:** message.

See also: – **removeFontTrait:**, – **modifyFont:**, – **modifyFontViaPanel:**

availableFontNamesWithTraits:

– (NSArray *)**availableFontNamesWithTraits:**(NSFontTraitMask)*fontTraitMask*

Returns the names of the fonts available in the system whose traits are described exactly by *fontTraitMask* (not the NSFont objects themselves). These fonts are discovered by searching the user's personal font directory, the local font directory, and the OPENSTEP system font directory. You specify the desired traits by combining these font trait mask values using the C bitwise OR operator:

NSItalicFontMask	NSCondensedFontMask
NSBoldFontMask	NSExpandedFontMask
NSNarrowFontMask	NSCompressedFontMask
NSFixedPitchFontMask	NSSmallCapsFontMask
NSPosterFontMask	

NSCondensedFontMask and NSExpandedFontMask are mutually exclusive.

See also: – **availableFonts**

availableFonts

– (NSArray *)**availableFonts**

Returns the names of the fonts available in the system (not the NSFont objects themselves). These fonts are discovered by searching the user's personal font directory, the local font directory, and the OPENSTEP system font directory.

See also: – **availableFontNamesWithTraits:**

convertFont:

– (NSFont *)**convertFont:**(NSFont *)*aFont*

Converts *aFont* according to the object that initiated a font change, typically the Font Panel or Font menu. Returns the converted font, or *aFont* itself if the conversion isn't possible.

This method is invoked in response to a **changeFont:** message, which is itself initiated by an action message such as **addFontTrait:** or **modifyFontViaPanel:**. These initiating methods cause the font manager to query the sender for the action to take and the traits to change. See the class description for more information.

See also: – **convertFont:toFace:**, – **convertFont:toFamily:**, – **convertFont:toHaveTrait:**, – **convertFont:toNotHaveTrait:**, – **convertFont:toSize:**, – **convertWeight:ofFont:**

convertFont:toFace:

– (NSFont *)**convertFont:**(NSFont *)*aFont* **toFace:**(NSString *)*typeface*

Returns an NSFont whose traits are as similar as possible to those of *aFont* except for the typeface, which is changed to *typeface*. Returns *aFont* if it can't be converted. A typeface is a fully specified family-face name, such as Helvetica-BoldOblique or Times-Roman.

This method attempts to match the weight and angle of *aFont* as closely as possible. Italic is mapped to Oblique, for example. Weights are mapped based on an approximate numeric scale of 0–15.

See also: – **convertFont:toFamily:**, – **convertFont:toHaveTrait:**, – **convertFont:toNotHaveTrait:**, – **convertFont:toSize:**, – **convertWeight:ofFont:**, – **convertFont:**

convertFont:toFamily:

– (NSFont *)**convertFont:**(NSFont *)*aFont* **toFamily:**(NSString *)*family*

Returns an NSFont whose traits are as similar as possible to those of *aFont* except for the font family, which is changed to *family*. Returns *aFont* if it can't be converted. A family is a generic font name, such as Helvetica or Times.

This method attempts to match the weight and angle of *aFont* as closely as possible. Italic is mapped to Oblique, for example. Weights are mapped based on an approximate numeric scale of 0–15.

See also: – **convertFont:toFace:**, – **convertFont:toHaveTrait:**, – **convertFont:toNotHaveTrait:**, – **convertFont:toSize:**, – **convertWeight:ofFont:**, – **convertFont:**

convertFont:toHaveTrait:

– (NSFont *)**convertFont:**(NSFont *)*aFont* **toHaveTrait:**(NSFontTraitMask)*fontTrait*

Returns an NSFont whose traits are the same as those of *aFont* except for the traits, which are changed to include the single trait *fontTrait*, which may be any one of:

NSItalicFontMask	NSCondensedFontMask
NSBoldFontMask	NSExpandedFontMask
NSNarrowFontMask	NSCompressedFontMask
NSFixedPitchFontMask	NSSmallCapsFontMask
NSPosterFontMask	

NSCondensedFontMask and NSExpandedFontMask are mutually exclusive.

Returns *aFont* if it can't be converted.

See also: – **convertFont:toNotHaveTrait:**, – **convertFont:toFace:**, – **convertFont:toFamily:**, – **convertFont:toSize:**, – **convertWeight:ofFont:**, – **convertFont:**

convertFont:toNotHaveTrait:

– (NSFont *)**convertFont:**(NSFont *)*aFont* **toNotHaveTrait:**(NSFontTraitMask)*fontTraitMask*

Returns an NSFont whose traits are the same as those of *aFont* except for the traits, which are changed so as not to include the single trait *fontTrait*, which may be any one of:

NSItalicFontMask	NSCondensedFontMask
NSBoldFontMask	NSExpandedFontMask
NSNarrowFontMask	NSCompressedFontMask
NSFixedPitchFontMask	NSSmallCapsFontMask
NSPosterFontMask	

NSCondensedFontMask and NSExpandedFontMask are mutually exclusive.

Returns *aFont* if it can't be converted.

See also: – **convertFont:toHaveTrait:**, – **convertFont:toFace:**, – **convertFont:toFamily:**, – **convertFont:toSize:**, – **convertWeight:ofFont:**, – **convertFont:**

convertFont:toSize:

– (NSFont *)**convertFont:**(NSFont *)*aFont* **toSize:**(float)*size*

Returns an NSFont whose traits are the same as those of *aFont* except for the size, which is changed to *size*. Returns *aFont* if it can't be converted.

See also: – **convertFont:toFace:**, – **convertFont:toFamily:**, – **convertFont:toHaveTrait:**, – **convertFont:toNotHaveTrait:**, – **convertWeight:ofFont:**, – **convertFont:**

convertWeight:ofFont:

– (NSFont *)**convertWeight:**(BOOL)*increaseFlag* **ofFont:**(NSFont *)*aFont*

Returns an NSFont whose weight is greater or lesser than that of *aFont*, if possible. If *increaseFlag* is YES, a heavier font is returned; if it's NO, a lighter font is returned. Returns *aFont* unchanged if it can't be converted. Weights are typically graded along this scale:

NSFontManager's implementation of this method refuses to convert a font's weight if it can't maintain all other traits, such as Italic and Condensed. You might wish to override this method to allow a looser interpretation of weight conversion.

See also: – **convertFont:toFace:**, – **convertFont:toFamily:**, – **convertFont:toHaveTrait:**, – **convertFont:toNotHaveTrait:**, – **convertFont:toSize:**, – **convertFont:**

delegate

– (id)**delegate**

Returns the receiver's delegate.

See also: – **setDelegate:**

fontMenu:

– (NSMenu *)**fontMenu:**(BOOL)*createFlag*

Returns the menu that's hooked up to the font conversion system, creating it if necessary and if *createFlag* is YES.

See also: – **setFontMenu:**

fontNamed:hasTraits:

– (BOOL)**fontNamed:**(NSString *)*typeface* **hasTraits:**(NSFontTraitMask)*fontTraitMask*

Returns YES if the font named *typeface* has all the traits specified in *fontTraitMask*, NO if it doesn't. You specify the desired traits by combining these font trait mask values using the C bitwise OR operator:

NSItalicFontMask	NSCondensedFontMask
NSBoldFontMask	NSExpandedFontMask
NSNarrowFontMask	NSCompressedFontMask
NSFixedPitchFontMask	NSSmallCapsFontMask
NSPosterFontMask	

NSCondensedFontMask and NSExpandedFontMask are mutually exclusive.

fontPanel:

– (NSFontPanel *)**fontPanel:**(BOOL)*createFlag*

Returns the application's shared Font Panel object, creating if necessary and if *createFlag* is YES.

See also: + **sharedFontPanel** (NSFontPanel), + **sharedFontPanelExists** (NSFontPanel), + **setFontPanelFactory:**

fontWithFamily:traits:weight:size:

– (NSFont *)**fontWithFamily:**(NSString *)*family*
traits:(NSFontTraitMask)*fontTraitMask*
weight:(int)*weight*
size:(float)*size*

Attempts to load a font with the specified characteristics, returning the font if successful and **nil** if not. *family* is the generic name of the font desired, such as Times or Helvetica. *weight* is a hint for the weight desired, on a scale of 0–15: a value of 5 indicates a normal or book weight and 9 or more a bold or heavier weight. It's ignored if *fontTraitMask* includes NSBoldFontMask.

You specify *fontTraitMask* by combining these font trait mask values using the C bitwise OR operator:

NSItalicFontMask	NSCondensedFontMask
NSBoldFontMask	NSExpandedFontMask
NSNarrowFontMask	NSCompressedFontMask
NSFixedPitchFontMask	NSSmallCapsFontMask
NSPosterFontMask	

NSCondensedFontMask and NSExpandedFontMask are mutually exclusive.

isEnabled

– (BOOL)**isEnabled**

Returns YES if the font-conversion system's user interface items (the Font Panel and Font menu items) are enabled, NO if they're not.

See also: – **isEnabled** (NSFontPanel), – **isEnabled** (NSMenuItem), – **setEnabled:**

isMultiple

– (BOOL)**isMultiple**

Returns YES if the last font selection recorded has multiple fonts, NO if it's a single font.

See also: – **setSelectedFont:isMultiple:**, – **selectedFont**

modifyFont:

– (void)**modifyFont:**(id)*sender*

This action method causes the receiver to send its action message (**changeFont:** by default) up the responder chain. When a responder replies by providing a font to convert in a **convertFont:** message, the receiver converts the font in the manner specified by *sender*. The conversion is determined by sending a **tag** message to *sender* and invoking a corresponding method:

Sender's Tag	Method Used
NSNoFontChangeAction	None, the font is returned unchanged
NSViaPanelFontAction	The Font Panel's **panelConvertFont:**
NSAddTraitFontAction	**convertFont:toHaveTrait:**
NSRemoveTraitFontAction	**convertFont:toNotHaveTrait:**
NSSizeUpFontAction	**convertFont:toSize:**
NSSizeDownFontAction	**convertFont:toSize:**
NSHeavierFontAction	**convertWeight:ofFont:**
NSLighterFontAction	**convertWeight:ofFont:**

See also: – **addFontTrait:**, – **removeFontTrait:**, – **modifyFontViaPanel:**

modifyFontViaPanel:

– (void)**modifyFontViaPanel:**(id)*sender*

This action method causes the receiver to send its action message (**changeFont:** by default) up the responder chain. When a responder replies by providing a font to convert in a **convertFont:** message, the receiver converts the font by sending a **panelConvertFont:** message to the Font Panel. The panel in turn may send **convertFont:toFamily:**, **convertFont:toHaveTrait:**, and other specific conversion methods to make its change.

See also: – **addFontTrait:**, – **removeFontTrait:**, – **modifyFont:**

orderFrontFontPanel:

– (void)**orderFrontFontPanel:**(id)*sender*

This action method opens the Font Panel by sending it an **orderFront:** message, creating the Font Panel if necessary.

See also: – **fontPanel:**, + **setFontPanelFactory:**

removeFontTrait:

– (void)**removeFontTrait:**(id)*sender*

This action method causes the receiver to send its action message (**changeFont:** by default) up the responder chain. When a responder replies by providing a font to convert in a **convertFont:** message, the receiver converts the font by removing the trait specified by *sender*. This trait is determined by sending a **tag** message to *sender* and interpreting it as a font trait mask for a **convertFont:toNotHaveTrait:** message.

See also: – **addFontTrait:**, – **modifyFont:**, – **modifyFontViaPanel:**

selectedFont

– (NSFont *)**selectedFont**

Returns the last font recorded with a **setSelectedFont:isMultiple:** message. While fonts are being converted in response to a **changeFont:** message, you can determine the font selected in the Font Panel like this:

```
NSFontManager *fontManager = [NSFontManager sharedFontManager];

panelFont = [fontManager convertFont:[fontManager selectedFont]];
```

See also: – **isMultiple**

sendAction

– (BOOL)**sendAction**

Sends the receiver's action message, **changeFont:** by default, up the responder chain, initiating a font change for whatever conversion and trait to change were last requested. Returns YES if some object handled the **changeFont:** message, NO if the message went unheard.

This method is used internally by the font conversion system. You should never need to invoke it directly. Instead, use the action methods such as **addFontTrait:** or **modifyFontViaPanel:**.

See also: – **setAction:**

setAction:

– (void)**setAction:**(SEL)*aSelector*

Sets the action that's sent to the first responder when the user selects a new font from the Font panel or chooses a command from the Font menu to *aSelector*. The default action is **changeFont:**. You should rarely need to change this.

See also: – **action**

setDelegate:

– (void)**setDelegate:**(id)*anObject*

Sets the receiver's delegate to *anObject*.

See also: – **delegate**

setEnabled:

– (void)**setEnabled:**(BOOL)*flag*

Controls whether the font-conversion system's user interface items (the Font Panel and Font menu items) are enabled. If *flag* is YES they're enabled; if *flag* is NO they're disabled.

See also: – **setEnabled:** (NSFontPanel), – **isEnabled**

setFontMenu:

– (void)**setFontMenu:**(NSMenu *)*aMenu*

Records *aMenu* as the application's Font menu.

See also: – **fontMenu:**

setSelectedFont:isMultiple:

– (void)**setSelectedFont:**(NSFont *)*aFont* **isMultiple:**(BOOL)*flag*

Records *aFont* as the currently selected font, and updates the Font Panel to reflect this. If *flag* is YES, the Font Panel indicates that more than one font is contained in the selection.

An object that manipulates fonts should invoke this method whenever it becomes first responder and whenever its selection changes. It shouldn't invoke this method in the process of handling a **changeFont:**

message, as this causes the font manager to lose the information necessary to effect the change. After all fonts have been converted, the font manager itself records the new selected font.

See also: – **selectedFont**, – **isMultiple**

traitsOfFont:

– (NSFontTraitMask)**traitsOfFont:**(NSFont *)*aFont*

Returns the traits of *aFont*, a mask created by combining these options with the C bitwise OR operator:

NSItalicFontMask	NSCondensedFontMask
NSBoldFontMask	NSExpandedFontMask
NSNarrowFontMask	NSCompressedFontMask
NSFixedPitchFontMask	NSSmallCapsFontMask
NSPosterFontMask	

NSCondensedFontMask and NSExpandedFontMask are mutually exclusive.

weightOfFont:

– (int)**weightOfFont:**(NSFont *)*aFont*

Returns a rough numeric measure of *aFont*'s weight, where 0 indicates the lightest possible weight, 5 indicates a normal or book weight, and 9 or more indicates a bold or heavier weight.

Methods Implemented By the Delegate

fontManager:willIncludeFont:

– (BOOL)**fontManager:**(id)*theFontManager* **willIncludeFont:**(const char *)*fontName*

Requests permission from the delegate to display *fontName* in the Font Panel. *fontName* is the full PostScript name of the font, such as "Helvetica-BoldOblique" or "Helvetica-Narrow-Bold". If the delegate returns YES, *fontName* is listed; if the delegate returns NO, it isn't.

This method is invoked repeatedly as necessary whenever the Font Panel needs updating, such as when the Font Panel is first loaded, and when the user selects a family name to see which typefaces in that family are available. Your implementation should execute fairly quickly to guarantee the responsiveness of the Font Panel.