NSCStringText

Inherits From:	NSText : NSView : NSResponder : NSObject
Conforms To:	NSChangeSpelling, NSIgnoreMisspelledWords (NSText) NSCoding (NSResponder) NSObject (NSObject)
Declared In:	AppKit/NSCStringText.h

Class Description

The NSCStringText class declares the programmatic interface to objects that manage text using eight-bit character encodings. The encoding is the same as the default C string encoding provided by **defaultCStringEncoding** in the NSString class. NSCStringText can be used in situations where backwards compatibility with the detailed interfaces of the NEXTSTEP Text object is important. Applications that can use the interface of NSText should do so.

The NSCStringText class is unlike most other classes in the Application Kit in its complexity and range of features. One of its design goals is to provide a comprehensive set of text-handling features so that you'll rarely need to create a subclass. An NSCStringText object can (among other things):

- Control the color of its text and background.
- Control the font and layout characteristics of its text.
- Control whether text is editable.
- Wrap text on a word or character basis.
- Write text to, or read it from, a file, as either RTF or plain ASCII data.
- Display graphic images within its text.
- Communicate with other applications through the Services menu.

- Let another object, the delegate, dynamically control its properties.
- Let the user copy and paste text within and between applications.
- Let the user copy and paste font and format information between NSCStringText objects.
- Let the user check the spelling of words in its text.
- Let the user control the format of paragraphs by manipulating a ruler.

NSCStringText can deal only with eight-bit characters. Therefore, it is not able to deal with Unicode character sets, and NSCStringText can't be fully internationalized.

Plain and Rich NSCStringText Objects

When you create an NSCStringText object directly, by default it allows only one font, line height, text color, and paragraph format for the entire text. You can set the default font used by new NSCStringText instances by sending the NSCStringText class object a **setDefaultFont:** message. Once an NSCStringText object is created, you can alter its global settings using methods such as **setFont:**, **setLineHeight:**, **setTextGray:**, and **setAlignment:**. For convenience, such an NSCStringText object will be called a *plain* NSCStringText object.

To allow multiple values for these attributes, you must send the NSCStringText object a **setRichText:YES** message. An NSCStringText object that allows multiple fonts also allows multiple paragraph formats, line heights, and so on. For convenience, such an NSCStringText object will be called a *rich* NSCStringText object.

A rich NSCStringText object can use RTF (Rich Text Format) as an interchange format. Not all RTF control words are supported: On input, an NSCStringText object ignores any control word it doesn't recognize; some of those it can read and interpret it doesn't write out. Refer to the class description of NSText for a list of the RTF control words that an NSCStringText object recognizes.

Note: An NSCStringText object writes eight-bit characters in the default C string encoding, which differs somewhat from the ANSI character set.

In an NSCStringText object, each sequence of characters having the same attributes is called a *run*. A plain NSCStringText object has only one run for the entire text. A rich NSCStringText object can have multiple runs. Methods such as **setSelFont**: and **setSelColor:** let you programmatically modify the attributes of the selected sequence of characters in a rich NSCStringText object. As discussed below, the user can set these attributes using the Font panel and the ruler.

NSCStringText objects are designed to work closely with various objects and services. Some of theseÐsuch as the delegate or an embedded graphic objectÐrequire a degree of programming on your part. OthersÐsuch as the Font panel, spelling checker, ruler, and Services menuÐtake no effort other than deciding whether the service should be enabled or disabled. The following sections discuss these interrelationships.

Notifying the NSCStringText Object's Delegate

Many of an NSCStringText object's actions can be controlled through an associated object, the NSCStringText object's delegate. If it implements any of the following methods, the delegate receives the corresponding message at the appropriate time:

textWillResize:
textDidResize:oldBounds:
textWillSetSel:toFont:
textWillConvert:fromFont:toFont:
textWillStartReadingRichText:
textWillFinishReadingRichText:
textWillWrite:
textDidRead:paperSize:

So, for example, if the delegate implements the **textWillConvert:fromFont:toFont:** method, it will receive notification upon the user's first attempt to change the font of the text. Moreover, depending on the method's return value, the delegate can either allow or prohibit changes to the text. See ªMethods Implemented by the Delegateº. The delegate can be any object you choose, and one delegate can control multiple NSCStringText objects.

Adding Graphics to the Text

A rich NSCStringText object allows graphics to be embedded in the text. Each graphic is treated as a single (possibly large) ªcharacterº: The text's line height and character placement are adjusted to accommodate the graphic ªcharacter.º Graphics are embedded in the text in either of two ways: programmatically or directly through user actions. The programmatic approach is discussed first.

In the programmatic approach, you add an objectÐgenerally a subclass of NSCellÐto the text. This object manages the graphic image by drawing it when appropriate. Although NSCell subclasses are commonly used, the only requirement is that the embedded object responds to these messagesÐsee ªMethods Implemented by an Embedded Graphic Objectº for more information:

highlight:withFrame:inView:
drawWithFrame:inView:
trackMouse:inRect:ofView:untilMouseUp:
cellSize:
readRichText:forView:
richTextforView:

You place the graphic object in the text by sending the NSCStringText object a **replaceSelWithCell:** message.

An NSCStringText object displays a graphic in its text by sending the managing object a **drawWithFrame:inView:** message. To record the graphic to a file or to the pasteboard, the NSCStringText object sends the managing object a **richTextforView:** message. The object must then write an RTF control word along with any data (such as the path of a TIFF file containing its image data) it might need to recreate its image. To reestablish the text containing the graphic image from RTF data, an NSCStringText object must know which class to associate with particular RTF control words. You associate a control word with a class object by sending the NSCStringText class object a **registerDirective:forClass:** message. Thereafter, whenever an NSCStringText object finds the registered control word in the RTF data being read from a file or the pasteboard, it will create a new instance of the class and send the object a **readRichText:forView:** message.

An alternate means of adding an image to the text is for the user to drag an EPS or TIFF file icon directly into an NSCStringText object. The NSCStringText object automatically creates a graphic object to manage the display of the image. This feature requires a rich NSCStringText object that has been configured to receive dragged imagesÐsee the **setImportsGraphics:** method of the NSText class.

Images that have been imported in this way can be written as RTFD documents. Programmatic creation of RTFD documents is not supported in this version of OpenStep. RTFD documents use a file package, or directory, to store the components of the document (the ªDº stands for ªdirectoryº). The file package has the name of the document plus a ª.rtfdº extension. The file package always contains a file called TXT.rtf for the text of the document, and one or more TIFF or EPS files for the images. An NSCStringText object can transfer information in an RTFD document to a file and read it from a fileÐsee the **writeRTFDToFile:atomically:** and **readRTFDFromFile:** methods in the NSText methods.

Cooperating with Other Objects and Services

NSCStringText objects are designed to work with the Application Kit's font conversion system. By default, an NSCStringText object keeps the Font panel updated with the font of the current selection. It also changes the font of the selection (for a rich NSCStringText object) or of the entire text (for a default NSCStringText object) to reflect the user's choices in the Font panel or menu. To disconnect an NSCStringText object from this service, send it a **setUsesFontPanel:NO** message (this method is actually implemented by NSTextÐthe superclass).

If an NSCStringText object is a subview of an NSScrollView, it can cooperate with the NSScrollView to display and update a ruler that displays formatting information. The NSScrollView retiles its subviews to make room for the ruler, and the NSCStringText object updates the ruler with the format information of the paragraph containing the selection. The **toggleRuler:** method controls the display of this ruler. Users can modify paragraph formats by manipulating the components of the ruler.

By means of the Services menu, an NSCStringText object can make use of facilities outside the scope of its own application. By default, an NSCStringText object registers with the services system that it can send and receive RTF and plain ASCII data. If the application containing the NSCStringText object has a Services menu, a menu item is added for each service provider that can accept or return these formats. To prevent NSCStringText objects from registering for services, send the NSCStringText class object an **excludeFromServicesMenu:YES** message before any NSCStringText objects are created.

Coordinates and sizes mentioned in the method descriptions below are in PostScript unitsÐ1/72 of an inch.

Initializing a New NSCStringText Object

- (id)**initWithFrame:**(NSRect)*frameRect* **text:**(NSString *)*theText* **alignment:**(NSTextAlignment)*mode*	Returns a new NSCStringText object at *frameRect* initialized with the contents of *theText* and with *mode* alignment.

Modifying the Frame Rectangle

- (void)**resizeTextWithOldBounds:**(NSRect)*oldBounds* **maxRect:**(NSRect)*maxRect*	Used by the NSCStringText object to resize and redisplay itself.

Laying Out the Text

- (int)**calcLine**	Calculates line breaks.
- (BOOL)**changeTabStopAt:**(float)*oldX* **to:**(float)*newX*	Resets the position of the specified tab stop.
- (BOOL)**charWrap**	Returns whether extra long words are wrapped.
- (void *)**defaultParagraphStyle**	Returns the default paragraph style.
- (float)**descentLine**	Returns distance from base line to bottom of line.
- (void)**getMarginLeft:**(float *)*leftMargin* **right:**(float *)*rightMargin* **top:**(float *)*topMargin* **bottom:**(float *)*bottomMargin*	Gets by reference the dimensions of margins around the text.
- (void)**getMinWidth:**(float *)*width* **minHeight:**(float *)*height* **maxWidth:**(float)*widthMax* **maxHeight:**(float)*heightMax*	Given the *widthMax* and *heightMax*, calculates the minimum area needed to display the text and returns *width* and *height* by reference.
- (float)**lineHeight**	Returns height of a line of text.
- (void *)**paragraphStyleForFont:**(NSFont *)*fontId* **alignment:**(int)*alignment*	Recalculates the paragraph style based on new font *fontId* and *alignment*.
- (void)**setCharWrap:**(BOOL)*flag*	Sets whether extra long words are wrapped.

- (void)**setDescentLine:**(float)*value*	Sets the distance from the base line to the bottom of line to *value*.
- (void)**setLineHeight:**(float)*value*	Sets the height of a line of text to *value*.
- (void)**setMarginLeft:**(float)*leftMargin* **right:**(float)*rightMargin* **top:**(float)*topMargin* **bottom:**(float)*bottomMargin*	Adjusts the margins around the text.
- (void)**setNoWrap**	Disables word wrap.
- (void)**setParagraphStyle:**(void *)*paraStyle*	Sets the default paragraph style for the entire text.
- (BOOL)**setSelProp:**(NSParagraphProperty)*property* **to:**(float)*value*	Sets a paragraph *property* for one or more selected paragraphs to *value*.

Reporting Line and Position

- (int)**lineFromPosition:**(int)*position*	Converts character *position* to line number.
- (int)**positionFromLine:**(int)*line*	Converts *line* number to character position.

Reading and Writing Text

- (void)**finishReadingRichText**	Sent after the NSCStringText object reads RTF data.
- (NSTextBlock *)**firstTextBlock**	Returns a pointer to the first text block in the NSCStringText object.
- (NSRect)**paragraphRect:**(int)*paraNumber* **start:**(int *)*startPos* **end:**(int *)*endPos*	Returns the location and size of a paragraph identified by *paraNumber*; also returns the starting and ending character positions by reference.
- (void)**startReadingRichText**	Sent before the NSCStringText object begins reading RTF data.

Editing Text

- (void)**clear:**(id)*sender*	Deletes the selected text.
- (void)**hideCaret**	Removes the caret from the text display.

- (void)**showCaret**	Displays the previously hidden caret in the text display.

Managing the Selection

- (void)**getSelectionStart:**(NSSelPt *)*start* **end:**(NSSelPt *)*end*	Gets information (by reference) relating to the starting and ending character positions of the selection.
- (void)**replaceSel:**(NSString *)*aString*	Replaces the selection with *aString*.
- (void)**replaceSel:**(NSString *)*aString* **length:**(int)*length*	Replaces the selection with *length* bytes of *aString*.
- (void)**replaceSel:**(NSString *)*aString* **length:**(int)*length* **runs:**(NSRunArray *)*insertRuns*	Replaces the selection with *length* bytes of *aString*. *insertRuns* is a pointer to the current run in the run array.
- (void)**scrollSelToVisible**	Brings the selection within the frame rectangle.
- (void)**selectError**	Selects all the text.
- (void)**selectNull**	Deselects the current selection.
- (void)**setSelectionStart:**(int)*start* **end:**(int)*end*	Selects text from characters *start* through *end*.
- (void)**selectText:**(id)*sender*	Makes the receiver the first responder and selects all text.

Setting the Font

+ (NSFont *)**defaultFont**	Returns the default NSFont object for NSCStringText objects.
+ (void)**setDefaultFont:**(NSFont *)*anObject*	Makes *anObject* the default NSFont object for NSCStringText objects.
- (void)**setFont:**(NSFont *)*fontObj* **paragraphStyle:**(void *)*paragraphStyle*	Sets the NSFont object and paragraph style for all text.
- (void)**setSelFont:**(NSFont *)*fontObj*	Sets the NSFont object for the selection.
- (void)**setSelFont:**(NSFont *)*fontObj* **paragraphStyle:**(void *)*paragraphStyle*	Sets the NSFont object and paragraph style for the selection.
- (void)**setSelFontFamily:**(NSString *)*fontName*	Sets the font family for the selection.

- (void)**setSelFontSize:**(float)*size*	Sets the font size for the selection.
- (void)**setSelFontStyle:**(NSFontTraitMask)*traits*	Sets the font style for the selection.

Finding Text

- (BOOL)**findText:**(NSString *)*textPattern* **ignoreCase:**(BOOL)*ignoreCase* **backwards:**(BOOL)*backwards* **wrap:**(BOOL)*wrap*	Searches for *textPattern* in the text, starting at the insertion point. *ignoreCase* instructs the search to disregard case; *backwards* means search backwards; *wrap* means that when the search reaches the beginning or end of the text (depending on the direction), it should continue by wrapping to the end or beginning of the text.

Modifying Graphic Attributes

- (NSColor *)**runColor:**(NSRun *)*run*	Returns the color of the specified text run.
- (NSColor *)**selColor**	Returns the color of the selected text.
- (void)**setSelColor:**(NSColor *)*color*	Sets the color of the selected text.

Reusing an NSCStringText Object

- (void)**renewFont:**(NSFont *)*newFontObj* **text:**(NSString *)*newText* **frame:**(NSRect)*newFrame* **tag:**(int)*newTag*	Resets the NSCStringText object to draw different text *newText* in font *newFontId* within frame *newFrame*.
- (void)**renewFont:**(NSString *)*newFontName* **size:**(float)*newFontSize* **style:**(int)*newFontStyle* **text:**(NSString *)*newText* **frame:**(NSRect)*newFrame* **tag:**(int)*newTag*	Resets the NSCStringText object to draw different text *newText* in the font identified by *newFontName*, *newFontSize*, and *newFontStyle.* Drawing occurs within frame *newFrame*.
- (void)**renewRuns:**(NSRunArray *)*newRuns* **text:**(NSString *)*newText* **frame:**(NSRect)*newFrame* **tag:**(int)*newTag*	Resets the NSCStringText object to draw different text *newText* in *newFrame*.

Setting Window Attributes

- (BOOL)**isRetainedWhileDrawing**	Returns whether a retained window is used for drawing.
- (void)**setRetainedWhileDrawing:**(BOOL)*flag*	Allows use of a retained window when drawing.

Assigning a Tag

- (void)**setTag:**(int)*anInt*	Makes *anInt* the NSCStringText object's tag.
- (int)**tag**	Returns the NSCStringText object's tag.

Handling Event Messages

- (void)**becomeKeyWindow**	Activates the caret if selection has width of 0.
- (void)**moveCaret:**(unsigned short)*theKey*	Moves the caret in response to arrow keys.
- (void)**resignKeyWindow**	Deactivates the caret.

Displaying Graphics within the Text

+ **registerDirective:**(NSString *)*directive* **forClass:***class*	Associates an RTF control word (*directive*) with *class* (usually NSCell and subclasses); objects of this class are encoded through RTF control words in NSCStringText objects.
- (NSPoint)**locationOfCell:**(NSCell *)*cell*	Returns the location of *cell*.
- (void)**replaceSelWithCell:**(NSCell *)*cell*	Replaces the selection with cell object *cell*.
- (void)**setLocation:**(NSPoint)*origin* **ofCell:**(NSCell *)*cell*	Sets the *origin* point of *cell*.

Using the Services Menu and the Pasteboard

+ **excludeFromServicesMenu:**(BOOL)*flag*	Controls whether NSCStringText objects can register for services.
- (BOOL)**readSelectionFromPasteboard:**(NSPasteboard *)*pboard*	

Replaces the selection with data from pasteboard *pboard*.

- (id)**validRequestorForSendType:**(NSString *)*sendType*
  returnType:(NSString *)*returnType* — Determines which Service menu items are enabled.

- (BOOL)**writeSelectionToPasteboard:**(NSPasteboard *)*pboard*
  types:(NSArray *)*types* — Copies the selection to pasteboard *pboard*.

Setting Tables and Functions

- (const NSFSM *)**breakTable**	Returns the table defining word boundaries.
- (const unsigned char *)**charCategoryTable**	Returns the table defining character categories.
- (NSCharFilterFunc)**charFilter**	Returns the current character filter function.
- (const NSFSM *)**clickTable**	Returns the table defining double-click selection.
- (NSTextFunc)**drawFunc**	Returns the current draw function.
- (const unsigned char *)**postSelSmartTable**	Returns cut and paste table for right word boundary.
- (const unsigned char *)**preSelSmartTable**	Returns cut and paste table for left word boundary.
- (NSTextFunc)**scanFunc**	Returns the current scan function.
- (void)**setBreakTable:**(const NSFSM *)*aTable*	Sets the table defining word boundaries.
- (void)**setCharCategoryTable:**(const unsigned char *)*aTable*	Sets the table defining character categories used in the word wrap or click tables.
- (void)**setCharFilter:**(NSCharFilterFunc)*aFunction*	Makes *aFunction* the character filter function.
- (void)**setClickTable:**(const NSFSM *)*aTable*	Sets the table defining double-click selection.
- (void)**setDrawFunc:**(NSTextFunc)*aFunction*	Makes *aFunction* the function that draws the text.
- (void)**setPostSelSmartTable:**(const unsigned char *)*aTable*	Sets the cut and paste table for right word boundary.
- (void)**setPreSelSmartTable:**(const unsigned char *)*aTable*	Sets the cut and paste table for left word boundary.

- (void)**setScanFunc:**(NSTextFunc)*aFunction*	Makes *aFunction* the scan function.
- (void)**setTextFilter:**(NSTextFilterFunc)*aFunction*	Makes *aFunction* the text filter function.
- (NSTextFilterFunc)**textFilter**	Returns the current text filter function.

Printing

- (void)**adjustPageHeightNew:**(float *)*newBottom* **top:**(float)*oldTop* **bottom:**(float)*oldBottom* **limit:**(float)*bottomLimit*	Assists with automatic pagination of text.

Implemented by an Embedded Graphic Object

- (NSSize)**cellSize**	Embedded cell returns its size.
- (void)**drawWithFrame:**(NSRect)*cellFrame* **inView:**(NSView *)*controlView*	Embedded object draws itself, including frame, within *cellFrame* in *controlView*.
- (void)**highlight:**(BOOL)*flag* **withFrame:**(NSRect)*cellFrame* **inView:**(NSView *)*controlView*	Embedded object highlights or unhighlights itself with *cellFrame* of *controlView*, depending on the value of *flag*.
- (void)**readRichText:**(NSString *)*stringObject* **forView:**(NSView *)*view*	Embedded object reads its RTF representation from *stringObject* and initializes itself.
- (NSString *)**richTextForView:**(NSView *)*view*	Embedded object stores its RTF representation within view as a string object and returns it.
- (BOOL)**trackMouse:**(NSEvent *)*theEvent* **inRect:**(NSRect)*cellFrame* **ofView:**(NSView *)*controlView* **untilMouseUp:**(BOOL)*untilMouseUp*	Embedded object implements this method to track mouse movement within tracking rectangle (*cellFrame*) and to detect mouse-up event (*untilMouseUp*).

Implemented by the Delegate

- (void)**textDidRead:**(NSCStringText *)*textObject* **paperSize:**(NSSize)*paperSize*	Lets the delegate review paper size.

- (NSRect)**textDidResize:**(NSCStringText *)*textObject*
  oldBounds:(NSRect)*oldBounds* — Reports size change to delegate.

- (NSFont *)**textWillConvert:**(NSCStringText *)*textObject*
  fromFont:(NSFont *)*font* — Lets delegate intercede in selection's font change.
  toFont:(NSFont *)*font*

- (void)**textWillFinishReadingRichText:**(NSCStringText *)*textObject*
  Informs delegate that the NSCStringText object finished reading RTF data.

- (void)**textWillResize:**(NSCStringText *)*textObject*
  Informs delegate of impending size change.

- (void)**textWillSetSel:**(NSCStringText *)*textObject*
  toFont:(NSFont *)*font* — Lets delegate intercede in the updating of the Font panel.

- (void)**textWillStartReadingRichText:**(NSCStringText *)*textObject*
  Informs delegate that NSCStringText object will read RTF data.

- (NSSize)**textWillWrite:**(NSCStringText *)*textObject*
  Lets the delegate specify paper size.

Compatibility Methods

- (NSCStringTextInternalState *)**cStringTextInternalState**
  Returns a structure that represents the instance variables of the NSCStringText object. The structure is defined in **appkit/NSCStringText.h**, and in the ªTypes and Constantsº section of the Application Kit documentation. Note that this method is provided for applications that really must depend on changing the values of an NSCStringText object's instance variables.