

---

# NSATSTypesetter Class Reference

User Experience: Text Layout



2009-03-04



Apple Inc.  
© 2009 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
408-996-1010

Apple, the Apple logo, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY**

**DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

## NSATTypesetter Class Reference 5

---

Overview	5
Subclassing Notes	5
Tasks	5
Getting a Typesetter	5
Managing the Layout Manager	6
Managing the Text Container	6
Mapping Screen and Printer Fonts	6
Managing Text Tabs	6
Bidirectional Text Processing	7
Accessing Paragraph Typesetting Information	7
Paragraph Layout	7
Line and Paragraph Spacing	7
Glyph Caching	7
Laying out Glyphs	8
Interfacing with Glyph Storage	8
Class Methods	9
sharedTypesetter	9
Instance Methods	9
attributedString	9
bidiProcessingEnabled	10
boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment: glyphPosition:characterIndex:	10
characterRangeForGlyphRange:actualGlyphRange:	10
currentTextContainer	11
deleteGlyphsInRange:	11
getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:	11
getLineFragmentRect:usedRect:forParagraphSeparatorGlyphRange:atProposedOrigin:	12
glyphRangeForCharacterRange:actualCharacterRange:	12
hyphenationFactor	13
hyphenationFactorForGlyphAtIndex:	13
hyphenCharacterForGlyphAtIndex:	14
insertGlyph:atGlyphIndex:characterIndex:	14
layoutManager	14
layoutParagraphAtPoint:	15
lineFragmentPadding	15
lineSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:	15
paragraphGlyphRange	16
paragraphSeparatorGlyphRange	16
paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:	17
paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:	17

[setAttachmentSize:forGlyphRange: 18](#)  
[setAttributedString: 18](#)  
[setBidiLevels:forGlyphRange: 18](#)  
[setBidiProcessingEnabled: 19](#)  
[setDrawsOutsideLineFragment:forGlyphRange: 19](#)  
[setHardInvalidation:forGlyphRange: 19](#)  
[setHyphenationFactor: 20](#)  
[setLineFragmentPadding: 20](#)  
[setLineFragmentRect:forGlyphRange:usedRect:baselineOffset: 20](#)  
[setLocation:withAdvancements:forStartOfGlyphRange: 21](#)  
[setNotShownAttribute:forGlyphRange: 21](#)  
[setParagraphGlyphRange:separatorGlyphRange: 22](#)  
[setTypeSetterBehavior: 22](#)  
[setUsesFontLeading: 23](#)  
[shouldBreakLineByHyphenatingBeforeCharacterAtIndex: 23](#)  
[shouldBreakLineByWordBeforeCharacterAtIndex: 23](#)  
[substituteFontForFont: 24](#)  
[substituteGlyphsInRange:withGlyphs: 24](#)  
[textTabForGlyphLocation:writingDirection:maxLocation: 24](#)  
[typesetterBehavior 25](#)  
[usesFontLeading 25](#)  
[willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset: 25](#)

**Appendix A      [Deprecated NSATSTypesetter Methods 27](#)**

---

[Deprecated in Mac OS X v10.4 27](#)  
[lineFragmentRectForProposedRect:remainingRect: 27](#)

**[Document Revision History 29](#)**

---

**[Index 31](#)**

---

# NSATSTypesetter Class Reference

---

<b>Inherits from</b>	NSTypesetter : NSObject
<b>Conforms to</b>	NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/AppKit.framework
<b>Availability</b>	Available in Mac OS X v10.3 and later.
<b>Declared in</b>	NSATSTypesetter.h
<b>Companion guides</b>	Text System Overview Text Layout Programming Guide for Cocoa

## Overview

`NSATSTypesetter` is a concrete subclass of `NSTypesetter` that places glyphs during the text layout process. The typesetter creates line fragment rectangles, positions glyphs within the line fragments, determines line breaks by word wrapping and hyphenation, and handles tab positioning.

`NSATSTypesetter` encapsulates the advanced typesetting capabilities of Core Text. `NSATSTypesetter` provides enhanced line and character spacing accuracy and supports more languages, including bidirectional languages, than the original, built-in typesetter class `NSSimpleHorizontalTypesetter` (which is deprecated in Mac OS X version 10.4 and later).

## Subclassing Notes

---

`NSATSTypesetter` introduced a set of interfaces in Mac OS X version 10.3 that facilitated subclassing and made it possible to substitute a custom layout engine into the Cocoa text system. In Mac OS X version 10.4, those interfaces moved to `NSTypesetter`, which you can subclass to the same effect. See the `NSTypesetter` reference documentation for relevant subclassing notes.

## Tasks

### Getting a Typesetter

- + [sharedTypesetter](#) (page 9)  
Returns a shared instance of `NSATSTypesetter`.

## Managing the Layout Manager

- `LayoutManager` (page 14)  
Returns the layout manager for the text being typeset.
- `setUsesFontLeading:` (page 23)  
Sets a Boolean value controlling whether the typesetter uses the leading (or line gap) value specified in the font metric information.
- `usesFontLeading` (page 25)  
Returns a Boolean value indicating whether the typesetter uses the leading (or line gap) value specified in the font metric information of the current font.
- `setTypesetterBehavior:` (page 22)  
Sets the default typesetter behavior, which affects glyph spacing and line height.
- `typesetterBehavior` (page 25)  
Returns the current typesetter behavior value.
- `setHyphenationFactor:` (page 20)  
Sets the threshold controlling when hyphenation is attempted
- `hyphenationFactor` (page 13)  
Returns the current hyphenation factor.

## Managing the Text Container

- `currentTextContainer` (page 11)  
Returns the text container for the text being typeset.
- `setLineFragmentPadding:` (page 20)  
Sets the amount (in points) by which text is inset within line fragment rectangles
- `lineFragmentPadding` (page 15)  
Returns the current line fragment padding amount; that is, the portion on each end of the line fragment rectangle left blank.

## Mapping Screen and Printer Fonts

- `substituteFontForFont:` (page 24)  
Returns a screen font suitable for use in place of the specified original font, or simply returns the original font if a screen font can't be used or isn't available.

## Managing Text Tabs

- `textTabForGlyphLocation:writingDirection:maxLocation:` (page 24)  
Returns the text tab next closest to a given glyph location, indexing in the specified direction but not beyond a given glyph location.

## Bidirectional Text Processing

- [setBidiProcessingEnabled:](#) (page 19)  
Sets a Boolean value controlling whether the typesetter performs bidirectional text processing.
- [bidiProcessingEnabled](#) (page 10)  
Returns a Boolean value indicating the bidirectional text processing setting currently in effect.

## Accessing Paragraph Typesetting Information

- [setAttributeutedString:](#) (page 18)  
Sets the text backing store on which this typesetter operates.
- [attributedString](#) (page 9)  
Returns the text backing store, usually an instance of `NSTextStorage`.
- [setParagraphGlyphRange:separatorGlyphRange:](#) (page 22)  
Sets the current glyph range being processed and the paragraph separator glyph range (the range of the paragraph separator character or characters).
- [paragraphGlyphRange](#) (page 16)  
Returns the glyph range currently being processed.
- [paragraphSeparatorGlyphRange](#) (page 16)  
Returns the current paragraph separator range, which is the full range that contains the current glyph range and that extends from one paragraph separator character to the next.

## Paragraph Layout

- [layoutParagraphAtPoint:](#) (page 15)  
Lays out glyphs in the current glyph range until the next paragraph separator is reached.

## Line and Paragraph Spacing

- [lineSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:](#) (page 15)  
Returns the line spacing in effect following the specified glyph.
- [paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:](#) (page 17)  
Returns the paragraph spacing, the number of points of space added following a paragraph, which is in effect after the specified glyph.
- [paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:](#) (page 17)  
Returns the number of points of space added before a paragraph, which is in effect before the specified glyph.

## Glyph Caching

- [setHardInvalidation:forGlyphRange:](#) (page 19)  
Sets a Boolean value controlling whether to force the layout manager to invalidate the portion of the glyph cache in the given glyph range when invalidating layout.

## Laying out Glyphs

- [boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:](#) (page 10)  
Returns the bounding rectangle for the given control glyph, at the given glyph position and character index, in the given text container.
- [getLineFragmentRect:usedRect:forParagraphSeparatorGlyphRange:atProposedOrigin:](#) (page 12)  
Calculates the line fragment rectangle and line fragment used rectangle for blank lines.
- [hyphenCharacterForGlyphAtIndex:](#) (page 14)  
Returns the hyphen character to be inserted after the given glyph when hyphenation is enabled in the layout manager.
- [hyphenationFactorForGlyphAtIndex:](#) (page 13)  
Returns the hyphenation factor in effect at the given glyph index.
- [shouldBreakLineByHyphenatingBeforeCharacterAtIndex:](#) (page 23)  
The typesetter calls this method, if implemented by a subclass, before breaking a line by hyphenating before the character at the given character index, enabling the subclass to control line breaking.
- [shouldBreakLineByWordBeforeCharacterAtIndex:](#) (page 23)  
The typesetter calls this method, if implemented by a subclass, before breaking a line by word wrapping before the character at the given character index, enabling the subclass to control line breaking.
- [willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset:](#) (page 25)  
Called by the typesetter just prior to calling [setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:](#) (page 20) which stores the actual line fragment rectangle location in the layout manager.

## Interfacing with Glyph Storage

- [characterRangeForGlyphRange:actualGlyphRange:](#) (page 10)  
Returns the range for the characters in the receiver's text store that are mapped to the glyphs in the given glyph range.
- [deleteGlyphsInRange:](#) (page 11)  
Deletes the glyphs in the given glyph range from the glyph cache maintained by the layout manager.
- [getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:](#) (page 11)  
Extracts the information needed to lay out the glyphs in the given glyph buffer from the given glyph range.
- [glyphRangeForCharacterRange:actualCharacterRange:](#) (page 12)  
Returns the range for the glyphs mapped to the characters of the text store in the given character range.
- [insertGlyph:atGlyphIndex:characterIndex:](#) (page 14)  
Enables the typesetter to insert a new glyph into the stream.
- [setAttachmentSize:forGlyphRange:](#) (page 18)  
Sets the size the glyphs in the given glyph range (assumed to be attachments) will be asked to draw themselves.
- [setBidiLevels:forGlyphRange:](#) (page 18)  
Sets the direction of the glyphs in the given glyph range for bidirectional text to the given levels.

- [setDrawsOutsideLineFragment:forGlyphRange:](#) (page 19)  
Sets a Boolean value controlling whether the glyphs in the given glyph range exceed the bounds of the line fragment in which they are laid out.
- [setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:](#) (page 20)  
Sets the line fragment rectangle where the glyphs in in the given glyph range are laid out to the given line fragment rectangle.
- [setLocation:withAdvancements:forStartOfGlyphRange:](#) (page 21)  
Sets the location where the glyphs in the given glyph range are laid out to the specified location.
- [setNotShownAttribute:forGlyphRange:](#) (page 21)  
Sets a Boolean value controlling whether the glyphs in the given glyph range are not shown.
- [substituteGlyphsInRange:withGlyphs:](#) (page 24)  
Replaces the glyphs in the given glyph range with the given glyphs.
- [lineFragmentRectForProposedRect:remainingRect:](#) (page 27) **Deprecated in Mac OS X v10.4**  
This method has been deprecated. Use the NSTypesetter method `getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter:` instead.

## Class Methods

### sharedTypesetter

Returns a shared instance of NSATTypesetter.

+ (id)sharedTypesetter

#### Availability

Available in Mac OS X v10.3 and later.

#### Declared In

NSATTypesetter.h

## Instance Methods

### attributedString

Returns the text backing store, usually an instance of NSTextStorage.

- (NSAttributedString \*)attributedString

#### Availability

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

#### See Also

- [setAttributedString:](#) (page 18)

**Declared In**

NSATSTypesetter.h

**bidirectionalProcessingEnabled**

Returns a Boolean value indicating the bidirectional text processing setting currently in effect.

- (BOOL)bidirectionalProcessingEnabled

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [setBidirectionalProcessingEnabled:](#) (page 19)

**Declared In**

NSATSTypesetter.h

**boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:**

Returns the bounding rectangle for the given control glyph, at the given glyph position and character index, in the given text container.

```
- (NSRect)boundingBoxForControlGlyphAtIndex:(NSUInteger)glyphIndex
    forTextContainer:(NSTextContainer *)textContainer
    proposedLineFragment:(NSRect)proposedRect glyphPosition:(NSPoint)glyphPosition
    characterIndex:(NSUInteger)charIndex
```

**Discussion**

Returns the bounding rectangle for the control glyph at *glyphIndex*, at the given *glyphPosition* and character index *charIndex*, in *textContainer*. The proposed line fragment rectangle is specified by *proposedRect*.

The typesetter calls this method when it encounters an NSControlGlyph. The default behavior is to return zero width for control glyphs. A subclass can override this method to do something different, such as implement a way to display control characters.

NSGlyphGenerator can choose whether or not to map control characters to NSControlGlyph. Tab characters, for example, do not use this facility.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**characterRangeForGlyphRange:actualGlyphRange:**

Returns the range for the characters in the receiver's text store that are mapped to the glyphs in the given glyph range.

- (NSRange)characterRangeForGlyphRange:(NSRange)glyphRange  
actualGlyphRange:(NSRangePointer)actualGlyphRange

**Discussion**

If *actualGlyphRange* is non-NULL, expands the requested range as needed so that it identifies all glyphs mapped to those characters and returns the new range by reference in *actualGlyphRange*.

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [glyphRangeForCharacterRange:actualCharacterRange:](#) (page 12)

**Declared In**

NSATSTypesetter.h

**currentTextContainer**

Returns the text container for the text being typeset.

- (NSTextContainer \*)currentTextContainer

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**deleteGlyphsInRange:**

Deletes the glyphs in the given glyph range from the glyph cache maintained by the layout manager.

- (void)deleteGlyphsInRange:(NSRange)glyphRange

**Discussion**

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

[insertGlyph:atGlyphIndex:characterIndex:](#) (page 14)

**Declared In**

NSATSTypesetter.h

**getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:**

Extracts the information needed to lay out the glyphs in the given glyph buffer from the given glyph range.

```
- (NSUInteger)glyphsInRange:(NSRange)glyphsRange glyphs:(NSGlyph *)glyphBuffer
    characterIndexes:(NSUInteger *)charIndexBuffer
    glyphInscriptions:(NSGlyphInscription *)inscribeBuffer elasticBits:(BOOL
*)elasticBuffer
```

**Discussion**

The *charIndexBuffer* contains the original characters for the glyphs. Note that a glyph at index 1 is not necessarily mapped to the character at index 1, since a glyph may be for a ligature or accent.

The *inscribeBuffer* contains the inscription attributes for each glyph, which are used to layout characters that are combined together. The possible values are described in the “Constants” section of the NSLayoutManager reference.

The *elasticBuffer* contains a Boolean value indicating whether a glyph is elastic for each glyph. An elastic glyph can be made longer at the end of a line or when needed for justification.

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**getLineFragmentRect:usedRect:forParagraphSeparatorGlyphRange:atProposedOrigin:**

Calculates the line fragment rectangle and line fragment used rectangle for blank lines.

```
- (void)getLineFragmentRect:(NSRect *)lineFragmentRect usedRect:(NSRect
*)lineFragmentUsedRect
    forParagraphSeparatorGlyphRange:(NSRange)paragraphSeparatorGlyphRange
    atProposedOrigin:(NSPoint)lineOrigin
```

**Discussion**

The method returns the calculated line fragment rectangle in *lineFragmentRect*, and it returns the used rectangle (the portion of the line fragment rectangle that actually contains marks) in *lineFragmentUsedRect*. The *paragraphSeparatorGlyphRange* is the range of glyphs under consideration, and *lineOrigin* is the origin point of the line fragment rectangle. A *paragraphSeparatorGlyphRange* with length 0 indicates an extra line fragment (which occurs if the last character in the paragraph is a line separator.)

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**glyphRangeForCharacterRange:actualCharacterRange:**

Returns the range for the glyphs mapped to the characters of the text store in the given character range.

```
- (NSRange)glyphRangeForCharacterRange:(NSRange)charRange
    actualCharacterRange:(NSRangePointer)actualCharRange
```

**Discussion**

If *actualCharRange* is non-NULL, expands the requested range as needed so that it identifies all characters mapped to those glyphs and returns the new range by reference in *actualCharRange*.

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [characterRangeForGlyphRange:actualGlyphRange:](#) (page 10)

**Declared In**

NSATSTypesetter.h

## hyphenationFactor

Returns the current hyphenation factor.

- (float)hyphenationFactor

**Discussion**

The hyphenation factor is a value ranging from 0.0 to 1.0 that controls when hyphenation is attempted. By default, the value is 0.0, meaning hyphenation is off. A factor of 1.0 causes hyphenation to be attempted always.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [setHyphenationFactor:](#) (page 20)

**Declared In**

NSATSTypesetter.h

## hyphenationFactorForGlyphAtIndex:

Returns the hyphenation factor in effect at the given glyph index.

- (float)hyphenationFactorForGlyphAtIndex:(NSUInteger)glyphIndex

**Discussion**

The hyphenation factor is a value ranging from 0.0 to 1.0 that controls when hyphenation is attempted. By default, the value is 0.0, meaning hyphenation is off. A factor of 1.0 causes hyphenation to be attempted always.

The typesetter calls this method with a proposed hyphenation point for a line break to find the hyphenation factor in effect at that time. A subclass can override this method to customize the text layout process.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [hyphenCharacterForGlyphAtIndex:](#) (page 14)

**Declared In**

NSATSTypesetter.h

**hyphenCharacterForGlyphAtIndex:**

Returns the hyphen character to be inserted after the given glyph when hyphenation is enabled in the layout manager.

```
- (UTF32Char)hyphenCharacterForGlyphAtIndex:(NSUInteger)glyphIndex
```

**Discussion**

The typesetter calls this method before hyphenating. A subclass can override this method to return a different hyphen glyph.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [hyphenationFactorForGlyphAtIndex:](#) (page 13)

**Declared In**

NSATSTypesetter.h

**insertGlyph:atGlyphIndex:characterIndex:**

Enables the typesetter to insert a new glyph into the stream.

```
- (void)insertGlyph:(NSGlyph)glyph atGlyphIndex:(NSUInteger)glyphIndex
    characterIndex:(NSUInteger)charIndex
```

**Discussion**

Inserts *glyph* into the glyph cache at *glyphIndex* and maps it to the character at *charIndex*. If the glyph is mapped to several characters, *charIndex* should indicate the first character to which it's mapped.

The standard typesetter uses this method for inserting hyphenation glyphs. Because this method keeps the glyph caches synchronized, subclasses should always use this method to insert glyphs instead of calling [layoutManager](#) (page 14) directly.

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3 through Mac OS X v10.3.

**Declared In**

NSATSTypesetter.h

**layoutManager**

Returns the layout manager for the text being typeset.

- (NSLayoutManager \*)layoutManager

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**layoutParagraphAtPoint:**

Lays out glyphs in the current glyph range until the next paragraph separator is reached.

- (NSUInteger)layoutParagraphAtPoint:(NSPoint \*)lineFragmentOrigin

**Discussion**

The *lineFragmentOrigin* specifies the upper-left corner of line fragment rectangle. On return, *lineFragmentOrigin* contains the next origin. This method returns the next glyph index. Usually it's the index right after the paragraph separator, but it can be inside the paragraph range if, for example, the end of the text container is reached before the paragraph separator.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**lineFragmentPadding**

Returns the current line fragment padding amount; that is, the portion on each end of the line fragment rectangle left blank.

- (CGFloat)lineFragmentPadding

**Discussion**

Text is inset within the line fragment rectangle by this amount.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [setLineFragmentPadding:](#) (page 20)

**Declared In**

NSATSTypesetter.h

**lineSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:**

Returns the line spacing in effect following the specified glyph.

- (CGFloat)lineSpacingAfterGlyphAtIndex:(NSUInteger)glyphIndex  
withProposedLineFragmentRect:(NSRect)rect

**Discussion**

The NSATTypesetter calls this method to determine the number of points of space to include below the descenders in the used rectangle for the proposed line fragment rectangle *rect*.

Line spacing, also called leading, is an attribute of NSParagraphStyle, which you can set on an NSMutableParagraphStyle object. A font typically includes a default minimum line spacing metric used if none is set in the paragraph style.

If the typesetter behavior specified in the NSLayoutManager is NSTypesetterOriginalBehavior, the text system uses the original, private typesetter NSSimpleHorizontalTypesetter, which adds the line spacing above the ascender. Similarly, NSATTypesetter adds the line spacing above the ascender if the value is negative.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATTypesetter.h

**paragraphGlyphRange**

Returns the glyph range currently being processed.

- (NSRange)paragraphGlyphRange

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [setParagraphGlyphRange:separatorGlyphRange:](#) (page 22)
- [paragraphSeparatorGlyphRange](#) (page 16)

**Declared In**

NSATTypesetter.h

**paragraphSeparatorGlyphRange**

Returns the current paragraph separator range, which is the full range that contains the current glyph range and that extends from one paragraph separator character to the next.

- (NSRange)paragraphSeparatorGlyphRange

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [setParagraphGlyphRange:separatorGlyphRange:](#) (page 22)
- [paragraphGlyphRange](#) (page 16)

**Declared In**

NSATTypesetter.h

**paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:**

Returns the paragraph spacing, the number of points of space added following a paragraph, which is in effect after the specified glyph.

```
- (CGFloat)paragraphSpacingAfterGlyphAtIndex:(NSUInteger)glyphIndex
  withProposedLineFragmentRect:(NSRect)rect
```

**Discussion**

The *rect* argument specifies the line fragment rectangle of the last line in the paragraph.

The typesetter adds the number of points specified in the return value to the bottom of the line fragment rectangle specified by *rect* (but not to the used line fragment rectangle for that line). Paragraph spacing added after a paragraph correlates to the value returned by the `paragraphSpacing` method of `NSParagraphStyle`, which you can set using the `setParagraphSpacing:` method of `NSMutableParagraphStyle`.

**Availability**

Available in Mac OS X v10.3. Moved to `NSTypesetter` in Mac OS X v10.4 and later.

**See Also**

- [paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:](#) (page 17)

**Declared In**

`NSATSTypesetter.h`

**paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:**

Returns the number of points of space added before a paragraph, which is in effect before the specified glyph.

```
- (CGFloat)paragraphSpacingBeforeGlyphAtIndex:(NSUInteger)glyphIndex
  withProposedLineFragmentRect:(NSRect)rect
```

**Discussion**

The *rect* argument specifies the line fragment rectangle of the first line in the paragraph.

The typesetter adds the number of points specified in the return value to the top of the line fragment rectangle specified by *rect* (but not to the used line fragment rectangle for that line). Paragraph spacing added before a paragraph correlates to the value returned by the `paragraphSpacingBefore` method of `NSParagraphStyle`, which you can set using the `setParagraphSpacingBefore:` method of `NSMutableParagraphStyle`.

**Availability**

Available in Mac OS X v10.3. Moved to `NSTypesetter` in Mac OS X v10.4 and later.

**See Also**

- [paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:](#) (page 17)

**Declared In**

`NSATSTypesetter.h`

**setAttachmentSize:forGlyphRange:**

Sets the size the glyphs in the given glyph range (assumed to be attachments) will be asked to draw themselves.

```
- (void)setAttachmentSize:(NSSize)attachmentSize forGlyphRange:(NSRange)glyphRange
```

**Discussion**

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**setAttributedString:**

Sets the text backing store on which this typesetter operates.

```
- (void)setAttributedString:(NSAttributedString *)attrString
```

**Discussion**

The string object is not retained.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [attributedString](#) (page 9)

**Declared In**

NSATSTypesetter.h

**setBidiLevels:forGlyphRange:**

Sets the direction of the glyphs in the given glyph range for bidirectional text to the given levels.

```
- (void)setBidiLevels:(const uint8_t *)levels forGlyphRange:(NSRange)glyphRange
```

**Discussion**

The value of *levels* can range from 0 to 61 as defined by Unicode Standard Annex #9.

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**setBidiProcessingEnabled:**

Sets a Boolean value controlling whether the typesetter performs bidirectional text processing.

- (void)setBidiProcessingEnabled:(BOOL)flag

**Discussion**

You can use this method to disable the bidirectional layout stage if you know the paragraph does not need this stage; that is, if the characters in the backing store are in display order.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [bidiProcessingEnabled](#) (page 10)

**Declared In**

NSATSTypesetter.h

**setDrawsOutsideLineFragment:forGlyphRange:**

Sets a Boolean value controlling whether the glyphs in the given glyph range exceed the bounds of the line fragment in which they are laid out.

- (void)setDrawsOutsideLineFragment:(BOOL)flag forGlyphRange:(NSRange)glyphRange

**Discussion**

This can happen when text is set at a fixed line height. For example, if the user specifies a fixed line height of 12 points and sets the font size to 24 points, the glyphs will exceed their layout rectangles.

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**setHardInvalidation:forGlyphRange:**

Sets a Boolean value controlling whether to force the layout manager to invalidate the portion of the glyph cache in the given glyph range when invalidating layout.

- (void)setHardInvalidation:(BOOL)flag forGlyphRange:(NSRange)glyphRange

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

## setHyphenationFactor:

Sets the threshold controlling when hyphenation is attempted

- (void)setHyphenationFactor:(float)*factor*

### Discussion

The *factor* argument is in the range of 0.0 to 1.0. By default, the value is 0.0, meaning hyphenation is off. A *factor* of 1.0 causes hyphenation to be attempted always.

### Availability

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

### See Also

- [hyphenationFactor](#) (page 13)

### Declared In

NSATSTypesetter.h

## setLineFragmentPadding:

Sets the amount (in points) by which text is inset within line fragment rectangles

- (void)setLineFragmentPadding:(CGFloat)*padding*

### Discussion

Note that line fragment padding isn't a suitable means for expressing margins; you should set the NSTextView object's position and size for document margins or the paragraph margin attributes for text margins.

### Availability

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

### See Also

- [lineFragmentPadding](#) (page 15)

### Declared In

NSATSTypesetter.h

## setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:

Sets the line fragment rectangle where the glyphs in in the given glyph range are laid out to the given line fragment rectangle.

- (void)setLineFragmentRect:(NSRect)*fragmentRect* forGlyphRange:(NSRange)*glyphRange*  
usedRect:(NSRect)*usedRect* baselineOffset:(CGFloat)*baselineOffset*

### Discussion

The exact positions of the glyphs must be set after the line fragment rectangle with `setLocation:forStartOfGlyphRange:.`

The *usedRect* argument indicates the portion of *fragmentRect*, in the `NSTextContainer` object's coordinate system, that actually contains glyphs or other marks that are drawn (including the text container's line fragment padding). The *usedRect* must be equal to or contained within *fragmentRect*. The *baselineOffset* argument is the vertical distance in pixels from the line fragment origin to the baseline on which the glyphs align.

A subclass can override this method to interact with custom glyph storage.

#### Availability

Available in Mac OS X v10.3. Moved to `NSTypesetter` in Mac OS X v10.4 and later.

#### Declared In

`NSATSTypesetter.h`

### setLocation:withAdvancements:forStartOfGlyphRange:

Sets the location where the glyphs in the given glyph range are laid out to the specified location.

```
- (void)setLocation:(NSPoint)location withAdvancements:(const CGFloat *)advancements
    forStartOfGlyphRange:(NSRange)glyphRange
```

#### Discussion

The x-coordinate of *location* is expressed relative to the line fragment rectangle origin, and the y-coordinate is expressed relative to the baseline previously specified by [setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:](#) (page 20). The *glyphRange* defines a series of glyphs that can be displayed with a single PostScript `show` operation (a nominal range). Setting the location for a series of glyphs implies that the glyphs preceding it can't be included in a single `show` operation. The *advancements* argument is the nominal glyph advance width specified in the font metric information.

Before setting the location for a glyph range, you must specify line fragment rectangle with `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:.`

A subclass can override this method to interact with custom glyph storage.

#### Availability

Available in Mac OS X v10.3. Moved to `NSTypesetter` in Mac OS X v10.4 and later.

#### Declared In

`NSATSTypesetter.h`

### setNotShownAttribute:forGlyphRange:

Sets a Boolean value controlling whether the glyphs in the given glyph range are not shown.

```
- (void)setNotShownAttribute:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

#### Discussion

For example, a tab or newline character doesn't leave any marks; it just indicates where following glyphs are laid out.

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

**setParagraphGlyphRange:separatorGlyphRange:**

Sets the current glyph range being processed and the paragraph separator glyph range (the range of the paragraph separator character or characters).

```
(void)setParagraphGlyphRange:(NSRange)paragraphRange
      separatorGlyphRange:(NSRange)paragraphSeparatorRange
```

**Parameters**

*paragraphRange*

The glyph range that becomes current.

*paragraphSeparatorRange*

The paragraph separator glyph range that becomes current.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [paragraphGlyphRange](#) (page 16)
- [paragraphSeparatorGlyphRange](#) (page 16)

**Declared In**

NSATSTypesetter.h

**setTypesetterBehavior:**

Sets the default typesetter behavior, which affects glyph spacing and line height.

```
(void)setTypesetterBehavior:(NSTypesetterBehavior)behavior
```

**Discussion**

The possible values for *behavior* are described in the “Constants” section of the NSLayoutManager reference.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**

- [typesetterBehavior](#) (page 25)

**Declared In**

NSATSTypesetter.h

## setUsesFontLeading:

Sets a Boolean value controlling whether the typesetter uses the leading (or line gap) value specified in the font metric information.

- (void)setUsesFontLeading:(BOOL)flag

### Availability

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

### See Also

- [usesFontLeading](#) (page 25)

### Declared In

NSATSTypesetter.h

## shouldBreakLineByHyphenatingBeforeCharacterAtIndex:

The typesetter calls this method, if implemented by a subclass, before breaking a line by hyphenating before the character at the given character index, enabling the subclass to control line breaking.

- (BOOL)shouldBreakLineByHyphenatingBeforeCharacterAtIndex:(NSUInteger)charIndex

### Discussion

A subclass can override this method to customize the text layout process. If the method returns NO, the typesetter continues looking for a break point.

### Availability

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

### See Also

- [shouldBreakLineByWordBeforeCharacterAtIndex:](#) (page 23)

### Declared In

NSATSTypesetter.h

## shouldBreakLineByWordBeforeCharacterAtIndex:

The typesetter calls this method, if implemented by a subclass, before breaking a line by word wrapping before the character at the given character index, enabling the subclass to control line breaking.

- (BOOL)shouldBreakLineByWordBeforeCharacterAtIndex:(NSUInteger)charIndex

### Discussion

A subclass can override this method to customize the text layout process. If the method returns NO, the typesetter continues looking for a break point.

### Availability

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

### See Also

- [shouldBreakLineByHyphenatingBeforeCharacterAtIndex:](#) (page 23)

**Declared In**

NSATTypesetter.h

**substituteFontForFont:**

Returns a screen font suitable for use in place of the specified original font, or simply returns the original font if a screen font can't be used or isn't available.

```
- (NSFont *)substituteFontForFont:(NSFont *)originalFont
```

**Discussion**

A screen font can be substituted if the receiver is set to use screen fonts and if no NSTextView associated with the receiver is scaled or rotated.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATTypesetter.h

**substituteGlyphsInRange:withGlyphs:**

Replaces the glyphs in the given glyph range with the given glyphs.

```
- (void)substituteGlyphsInRange:(NSRange)glyphRange withGlyphs:(NSGlyph *)glyphs
```

**Discussion**

This method does not alter the glyph-to-character mapping or invalidate layout information.

A subclass can override this method to interact with custom glyph storage.

**Availability**

Available in Mac OS X v10.3 and later.

**Declared In**

NSATTypesetter.h

**textTabForGlyphLocation:writingDirection:maxLocation:**

Returns the text tab next closest to a given glyph location, indexing in the specified direction but not beyond a given glyph location.

```
- (NSTextTab *)textTabForGlyphLocation:(CGFloat)glyphLocation
    writingDirection:(NSWritingDirection)direction maxLocation:(CGFloat)maxLocation
```

**Discussion**

The typesetter calls this method whenever it finds a tab character. To determine the width to advance the next glyph, the typesetter examines the NSParagraphStyle tab array and the default tab interval.

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATTypesetter.h

**typesetterBehavior**

Returns the current typesetter behavior value.

- (NSTypesetterBehavior)typesetterBehavior

**Discussion**The possible return values are described in the “Constants” section of the [NSLayoutManager](#) reference.**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**- [setTypesetterBehavior:](#) (page 22)**Declared In**

NSATTypesetter.h

**usesFontLeading**

Returns a Boolean value indicating whether the typesetter uses the leading (or line gap) value specified in the font metric information of the current font.

- (BOOL)usesFontLeading

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**See Also**- [setUsesFontLeading:](#) (page 23)**Declared In**

NSATTypesetter.h

**willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset:**

Called by the typesetter just prior to calling

[setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:](#) (page 20) which stores the actual line fragment rectangle location in the layout manager.- (void)willSetLineFragmentRect:(NSRect \*)lineRect forGlyphRange:(NSRange)glyphRange  
usedRect:(NSRect \*)usedRect baselineOffset:(CGFloat \*)baselineOffset**Discussion**The *lineRect* argument is the rectangle in which the glyphs in *glyphRange* are laid out. The *usedRect* argument indicates the portion of *lineRect*, in the NSTextContainer object’s coordinate system, that actually contains glyphs or other marks that are drawn (including the text container’s line fragment padding). The *usedRect* must be equal to or contained within *lineRect*. The *baselineOffset* argument is the vertical distance in pixels from the line fragment origin to the baseline on which the glyphs align.

A subclass can override this method to customize the text layout process. For example, it could change the shape of the line fragment rectangle. The subclass is responsible for ensuring that the modified rectangle remains valid (for example, that it lies within the text container).

**Availability**

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

**Declared In**

NSATSTypesetter.h

# Deprecated NSATSTypesetter Methods

---

A method identified as deprecated has been superseded and may become unsupported in the future.

## Deprecated in Mac OS X v10.4

### **lineFragmentRectForProposedRect:remainingRect:**

This method has been deprecated. Use the NSTypesetter method `getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter:` instead. (Deprecated in Mac OS X v10.4.)

```
- (NSRect)lineFragmentRectForProposedRect:(NSRect)proposedRect
    remainingRect:(NSRectPointer)remainingRect
```

#### **Discussion**

Returns the largest rectangle available for the proposed rectangle *proposedRect*. It also returns a rectangle in *remainingRect* containing any remaining space, such as that left on the other side of a hole or gap in the text container.

A subclass can override this method to interact with custom glyph storage.

#### **Availability**

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.4.

#### **Declared In**

NSATSTypesetter.h



# Document Revision History

---

This table describes the changes to *NSATSTypesetter Class Reference*.

Date	Notes
2009-03-04	Removed reference to ATSUI.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

attributedString **instance method** [9](#)

## B

---

bidiProcessingEnabled **instance method** [10](#)  
boundingBoxForControlGlyphAtIndex:  
    forTextContainer:proposedLineFragment:  
    glyphPosition:characterIndex: **instance  
method** [10](#)

## C

---

characterRangeForGlyphRange:actualGlyphRange:  
    **instance method** [10](#)  
currentTextContainer **instance method** [11](#)

## D

---

deleteGlyphsInRange: **instance method** [11](#)

## G

---

getGlyphsInRange:glyphs:characterIndexes:  
    glyphInscriptions:elasticBits: **instance  
method** [11](#)  
getLineFragmentRect:usedRect:  
    forParagraphSeparatorGlyphRange:atProposedOrigin:  
    **instance method** [12](#)  
glyphRangeForCharacterRange:actualCharacterRange:  
    **instance method** [12](#)

## H

---

hyphenationFactor **instance method** [13](#)  
hyphenationFactorForGlyphAtIndex: **instance  
method** [13](#)  
hyphenCharacterForGlyphAtIndex: **instance method**  
[14](#)

## I

---

insertGlyph:atGlyphIndex:characterIndex:  
    **instance method** [14](#)

## L

---

layoutManager **instance method** [14](#)  
layoutParagraphAtPoint: **instance method** [15](#)  
lineFragmentPadding **instance method** [15](#)  
lineFragmentRectForProposedRect:remainingRect:  
    **instance method** [27](#)  
lineSpacingAfterGlyphAtIndex:  
    withProposedLineFragmentRect: **instance  
method** [15](#)

## P

---

paragraphGlyphRange **instance method** [16](#)  
paragraphSeparatorGlyphRange **instance method** [16](#)  
paragraphSpacingAfterGlyphAtIndex:  
    withProposedLineFragmentRect: **instance  
method** [17](#)  
paragraphSpacingBeforeGlyphAtIndex:  
    withProposedLineFragmentRect: **instance  
method** [17](#)

S

---

setAttachmentSize:forGlyphRange: **instance method 18**  
 setAttributedString: **instance method 18**  
 setBidiLevels:forGlyphRange: **instance method 18**  
 setBidiProcessingEnabled: **instance method 19**  
 setDrawsOutsideLineFragment:forGlyphRange: **instance method 19**  
 setHardInvalidation:forGlyphRange: **instance method 19**  
 setHyphenationFactor: **instance method 20**  
 setLineFragmentPadding: **instance method 20**  
 setLineFragmentRect:forGlyphRange:usedRect:baselineOffset: **instance method 20**  
 setLocation:withAdvancements:forStartOfGlyphRange: **instance method 21**  
 setNotShownAttribute:forGlyphRange: **instance method 21**  
 setParagraphGlyphRange:separatorGlyphRange: **instance method 22**  
 setTypesetterBehavior: **instance method 22**  
 setUsesFontLeading: **instance method 23**  
 sharedTypesetter **class method 9**  
 shouldBreakLineByHyphenatingBeforeCharacterAtIndex: **instance method 23**  
 shouldBreakLineByWordBeforeCharacterAtIndex: **instance method 23**  
 substituteFontForFont: **instance method 24**  
 substituteGlyphsInRange:withGlyphs: **instance method 24**

T

---

textTabForGlyphLocation:writingDirection:maxLocation: **instance method 24**  
 typesetterBehavior **instance method 25**

U

---

usesFontLeading **instance method 25**

W

---

willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset: **instance method 25**