CTLine Reference

Carbon > Text & Fonts

2007-05-24

Contents

CTLine Reference

Derived From:	CFType
Framework:	ApplicationServices/CoreText
Declared in	CTLine.h

Overview

The CTLine opaque type represents a line of text.

A CTLine object contains an array of glyph runs. Line objects are created by the typesetter during a framesetting operation and can draw themselves directly into a graphics context.

Functions by Task

Creating Lines

`CTLineCreateWithAttributedString` (page 7)
Creates a single immutable line object directly from an attributed string.

`CTLineCreateTruncatedLine` (page 7)
Creates a truncated line from an existing line.

`CTLineCreateJustifiedLine` (page 6)
Creates a justified line from an existing line.

Drawing the Line

`CTLineDraw` (page 8)
Draws a complete line.

Getting Line Data

`CTLineGetGlyphCount` (page 8)
Returns the total glyph count for the line object.

`CTLineGetGlyphRuns` (page 9)
Returns the array of glyph runs that make up the line object.

`CTLineGetStringRange` (page 11)
: Gets the range of characters that originally spawned the glyphs in the line.

`CTLineGetPenOffsetForFlush` (page 10)
: Gets the pen offset required to draw flush text.

Measuring Lines

`CTLineGetImageBounds` (page 9)
: Calculates the image bounds for a line.

`CTLineGetTypographicBounds` (page 12)
: Calculates the typographic bounds of a line.

`CTLineGetTrailingWhitespaceWidth` (page 12)
: Returns the trailing whitespace width for a line.

Getting Line Positioning

`CTLineGetStringIndexForPosition` (page 11)
: Performs hit testing.

`CTLineGetOffsetForStringIndex` (page 10)
: Determines the graphical offset or offsets for a string index.

Getting the Type Identifier

`CTLineGetTypeID` (page 12)
: Returns the Core Foundation type identifier of the line object.

Functions

CTLineCreateJustifiedLine

Creates a justified line from an existing line.

```
CTLineRef CTLineCreateJustifiedLine( CTLineRef line, CGFloat justificationFactor,
 double justificationWidth );
```

Parameters

line
: The line from which to create a justified line.

justificationFactor
: Full or partial justification. When set to `1.0` or greater, full justification is performed. If this parameter is set to less than `1.0`, varying degrees of partial justification are performed. If it is set to `0` or less, no justification is performed.

justificationWidth

The width to which the resultant line is justified. If *justificationWidth* is less than the actual width of the line, then negative justification is performed (that is, glyphs are squeezed together).

Return Value

A reference to a justified CTLine object if the call was successful; otherwise, `NULL`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CTLine.h`

CTLineCreateTruncatedLine

Creates a truncated line from an existing line.

```
CTLineRef CTLineCreateTruncatedLine( CTLineRef line, double width,
CTLineTruncationType truncationType, CTLineRef truncationToken );
```

Parameters

line

The line from which to create a truncated line.

width

The width at which truncation begins. The line is truncated if its width is greater than the width passed in this parameter.

truncationType

The type of truncation to perform if needed. See "CTLineTruncationType" (page 13) for possible values.

truncationToken

This token is added at the point where truncation took place, to indicate that the line was truncated. Usually, the truncation token is the ellipsis character (`U+2026`). If this parameter is set to `NULL`, then no truncation token is used and the line is simply cut off.

Return Value

A reference to a truncated CTLine object if the call was successful; otherwise, `NULL`.

Discussion

The line specified in *truncationToken* should have a width less than the width specified by the *width* parameter. If the width of the line specified in *truncationToken* is greater than *width* and truncation is needed, the function returns `NULL`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CTLine.h`

CTLineCreateWithAttributedString

Creates a single immutable line object directly from an attributed string.

```
CTLineRef CTLineCreateWithAttributedString( CFAttributedStringRef string );
```

Parameters

`string`
The string from which the line is created.

Return Value
A reference to a CTLine object if the call was successful; otherwise, `NULL`.

Discussion
This function allows clients who need very simple line generation to create a line without creating a typesetter object. The typesetting is done under the hood. Without a typesetter object, the line cannot be properly broken. However, for simple things like text labels, line breaking is not an issue.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

CTLineDraw

Draws a complete line.

```
void CTLineDraw( CTLineRef line, CGContextRef context );
```

Parameters

`line`
The line to draw.

`context`
The context into which the line is drawn.

Discussion
This is a convenience function because the line could be drawn run-by-run by getting the glyph runs, getting the glyphs out of them, and calling a function such as `CGContextShowGlyphsAtPositions`. This call can leave the graphics context in any state and does not flush the context after the draw operation.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

CTLineGetGlyphCount

Returns the total glyph count for the line object.

```
CFIndex CTLineGetGlyphCount( CTLineRef line );
```

Parameters

`line`
The line whose glyph count is returned.

Return Value
The total glyph count for the line passed in.

Discussion
The total glyph count is equal to the sum of all of the glyphs in the glyph runs forming the line.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

CTLineGetGlyphRuns

Returns the array of glyph runs that make up the line object.

```
CFArrayRef CTLineGetGlyphRuns( CTLineRef line );
```

Parameters

`line`
The line whose glyph run array is returned.

Return Value
A `CFArrayRef` containing the CTRun objects that make up the line.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

CTLineGetImageBounds

Calculates the image bounds for a line.

```
CGRect CTLineGetImageBounds( CTLineRef line, CGContextRef context );
```

Parameters

`line`
The line whose image bounds are calculated.

`context`
The context for which the image bounds are calculated. This is required because the context could have settings in it that would cause changes in the image bounds.

Return Value
A rectangle that tightly encloses the paths of the line's glyphs, or, if the line or context is invalid, `CGRectNull`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

CTLineGetOffsetForStringIndex

Determines the graphical offset or offsets for a string index.

```
CGFloat CTLineGetOffsetForStringIndex( CTLineRef line, CFIndex charIndex, CGFloat*
 secondaryOffset );
```

Parameters

`line`

The line from which the offset is requested.

`charIndex`

The string index corresponding to the desired position.

`secondaryOffset`

On output, the secondary offset along the baseline for *`charIndex`*. When a single caret is sufficient for a string index, this value will be the same as the primary offset, which is the return value of this function. May be `NULL`.

Return Value

The primary offset along the baseline for *`charIndex`*, or `0.0` if the line does not support string access.

Discussion

This function returns the graphical offset or offsets corresponding to a string index, suitable for movement between adjacent lines or for drawing a custom caret. For moving between adjacent lines, the primary offset can be adjusted for any relative indentation of the two lines; a `CGPoint` constructed with the adjusted offset for its `x` value and `0.0` for its `y` value is suitable for passing to `CTLineGetStringIndexForPosition` (page 11). For drawing a custom caret, the returned primary offset corresponds to the portion of the caret that represents the visual insertion location for a character whose direction matches the line's writing direction.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CTLine.h`

CTLineGetPenOffsetForFlush

Gets the pen offset required to draw flush text.

```
double CTLineGetPenOffsetForFlush( CTLineRef line, CGFloat flushFactor, double
flushWidth );
```

Parameters

`line`

The line from which to obtain a flush position.

`flushFactor`

Determines the type of flushness. A *`flushFactor`* of `0` or less indicates left flush. A *`flushFactor`* of `1.0` or more indicates right flush. Flush factors between `0` and `1.0` indicate varying degrees of center flush, with a value of `0.5` being totally center flush.

`flushWidth`

Specifies the width to which the flushness operation should apply.

Return Value

The offset from the current pen position for the flush operation.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

CTLineGetStringIndexForPosition

Performs hit testing.

```
CFIndex CTLineGetStringIndexForPosition( CTLineRef line, CGPoint position );
```

Parameters

`line`
The line being examined.

`position`
The location of the mouse click relative to the line's origin.

Return Value
The string index for the position, or if the line does not support string access, `kCFNotFound`. Relative to the line's string range, this value can be no less than the first string index and no greater than the last string index plus 1.

Discussion
This function can be used to determine the string index for a mouse click or other event. This string index corresponds to the character before which the next character should be inserted. This determination is made by analyzing the string from which a typesetter was created and the corresponding glyphs as embodied by a particular line.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

CTLineGetStringRange

Gets the range of characters that originally spawned the glyphs in the line.

```
CFRange CTLineGetStringRange( CTLineRef line );
```

Parameters

`line`
The line from which to obtain the string range.

Return Value
A `CFRange` structure that contains the range over the backing store string that spawned the glyphs, or if the function fails for any reason, an empty range.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

CTLineGetTrailingWhitespaceWidth

Returns the trailing whitespace width for a line.

```
double CTLineGetTrailingWhitespaceWidth( CTLineRef line );
```

Parameters

`line`

The line whose trailing whitespace width is calculated.

Return Value

The width of the line's trailing whitespace. If the line is invalid, this function will always return zero.

Discussion

Creating a line for a width can result in a line that is actually longer than the desired width due to trailing whitespace. Although this is typically not an issue due to whitespace being invisible, this function can be used to determine what amount of a line's width is due to trailing whitespace.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CTLine.h`

CTLineGetTypeID

Returns the Core Foundation type identifier of the line object.

```
CFTypeID CTLineGetTypeID( void );
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CTLine.h`

CTLineGetTypographicBounds

Calculates the typographic bounds of a line.

```
double CTLineGetTypographicBounds( CTLineRef line, CGFloat* ascent, CGFloat* descent,
 CGFloat* leading );
```

Parameters

`line`

The line whose typographic bounds are calculated.

`ascent`

On output, the ascent of the line. This parameter can be set to `NULL` if not needed.

`descent`

On output, the descent of the line. This parameter can be set to `NULL` if not needed.

`leading`

On output, the leading of the line. This parameter can be set to `NULL` if not needed.

Return Value
The typographic width of the line. If the line is invalid, this function returns `0`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

Data Types

CTLineRef

A reference to a line object.

```
typedef const struct __CTLine *CTLineRef;
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CTLine.h`

Constants

CTLineTruncationType

Truncation types required by the `CTLineCreateTruncatedLine` (page 7) function to tell the truncation engine which type of truncation is being requested.

```
enum{
    kCTLineTruncationStart = 0,
    kCTLineTruncationEnd = 1,
    kCTLineTruncationMiddle = 2
};
typedef uint32_t CTLineTruncationType;
```

Constants

`kCTLineTruncationStart`
: Truncate the beginning of the line, leaving the end portion visible.

  Available in Mac OS X v10.5 and later.

  Declared in `CTLine.h`.

`kCTLineTruncationEnd`
: Truncate the end of the line, leaving the start portion visible.

  Available in Mac OS X v10.5 and later.

  Declared in `CTLine.h`.

`kCTLineTruncationMiddle`

Truncate the middle of the line, leaving both the start and the end portions visible.

Available in Mac OS X v10.5 and later.

Declared in `CTLine.h`.

Declared In

`CTLine.h`

Document Revision History

This table describes the changes to *CTLine Reference*.

Date	Notes
2007-05-24	New document that describes the Core Text opaque type used to represent a line of text.

Index