

---

# CTRun Reference

[Carbon](#) > [Text & Fonts](#)



2007-05-24



Apple Inc.  
© 2007 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, Carbon, 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

---

## **CTRun Reference 5**

Overview	5
Functions by Task	5
Getting Glyph Run Data	5
Measuring the Glyph Run	6
Drawing the Glyph Run	6
Getting the Type Identifier	6
Functions	6
CTRunDraw	6
CTRunGetAttributes	7
CTRunGetGlyphCount	7
CTRunGetGlyphs	7
CTRunGetGlyphsPtr	8
CTRunGetImageBounds	8
CTRunGetPositions	9
CTRunGetPositionsPtr	9
CTRunGetStatus	10
CTRunGetStringIndices	10
CTRunGetStringIndicesPtr	11
CTRunGetStringRange	11
CTRunGetTextMatrix	12
CTRunGetTypeID	12
CTRunGetTypographicBounds	12
Data Types	13
CTRunRef	13
Constants	13
CTRunStatus	13

---

## **Document Revision History 15**

---

## **Index 17**

---



# CTRun Reference

---

<b>Derived From:</b>	CType
<b>Framework:</b>	ApplicationServices/CoreText
<b>Declared in</b>	CTRun.h

## Overview

The CTRun opaque type represents a glyph run, which is a set of consecutive glyphs sharing the same attributes and direction.

The typesetter creates glyph runs as it produces lines from character strings, attributes, and font objects. That is, a line is constructed of one or more glyph runs. Glyph runs can draw themselves into a graphic context, if desired, although most users have no need to interact directly with glyph runs.

## Functions by Task

### Getting Glyph Run Data

[CTRunGetGlyphCount](#) (page 7)

Gets the glyph count for the run.

[CTRunGetAttributes](#) (page 7)

Returns the attribute dictionary that was used to create the glyph run.

[CTRunGetStatus](#) (page 10)

Returns the run's status.

[CTRunGetGlyphsPtr](#) (page 8)

Returns a direct pointer for the glyph array stored in the run.

[CTRunGetGlyphs](#) (page 7)

Copies a range of glyphs into a user-provided buffer.

[CTRunGetPositionsPtr](#) (page 9)

Returns a direct pointer for the glyph position array stored in the run.

[CTRunGetPositions](#) (page 9)

Copies a range of glyph positions into a user-provided buffer.

[CTRunGetStringIndicesPtr](#) (page 11)

Returns a direct pointer for the string indices stored in the run.

[CTRunGetStringIndices](#) (page 10)

Copies a range of string indices into a user-provided buffer.

[CTRunGetStringRange](#) (page 11)

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

## Measuring the Glyph Run

[CTRunGetTypographicBounds](#) (page 12)

Gets the typographic bounds of the run.

[CTRunGetImageBounds](#) (page 8)

Calculates the image bounds for a glyph range.

## Drawing the Glyph Run

[CTRunDraw](#) (page 6)

Draws a complete run or part of one.

[CTRunGetTextMatrix](#) (page 12)

Returns the text matrix needed to draw this run.

## Getting the Type Identifier

[CTRunGetTypeID](#) (page 12)

Returns the Core Foundation type identifier of the run object.

# Functions

### CTRunDraw

Draws a complete run or part of one.

```
void CTRunDraw( CTRunRef run, CGContextRef context, CFRange range );
```

#### Parameters

*run*

The run to draw.

*context*

The context into which to draw the run.

*range*

The portion of the run to draw. If the length of the range is set to 0, then the draw operation continues from the start index of the range to the end of the run.

#### Discussion

This is a convenience call, because the run could be drawn by accessing the glyphs. 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**

CTRun.h

**CTRunGetAttributes**

Returns the attribute dictionary that was used to create the glyph run.

```
CFDictionaryRef CTRunGetAttributes( CTRunRef run );
```

**Parameters**

*run*

The run for which to return attributes.

**Return Value**

A valid `CFDictionaryRef` or `NULL` on error or if the run has no attributes.

**Discussion**

The dictionary returned is either the same one that was set as an attribute dictionary on the original attributed string or a dictionary that has been manufactured by the layout engine. Attribute dictionaries can be manufactured in the case of font substitution or if the run is missing critical attributes.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CTRun.h

**CTRunGetGlyphCount**

Gets the glyph count for the run.

```
CFIndex CTRunGetGlyphCount( CTRunRef run );
```

**Parameters**

*run*

The run for which to return the glyph count.

**Return Value**

The number of glyphs that the run contains, or if there are no glyphs in this run, a value of 0.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CTRun.h

**CTRunGetGlyphs**

Copies a range of glyphs into a user-provided buffer.

```
void CTRunGetGlyphs( CTRunRef run, CFRange range, CGGlyph* buffer );
```

**Parameters***run*

The run from which to copy glyphs.

*range*

The range of glyphs to copy. If the length of the range is set to 0, then the copy operation continues from the range's start index to the end of the run.

*buffer*

The buffer the glyphs are copied to. The buffer must be allocated to at least the value specified by the range's length.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CTRun.h

**CTRunGetGlyphsPtr**

Returns a direct pointer for the glyph array stored in the run.

```
const CGGlyph* CTRunGetGlyphsPtr( CTRunRef run );
```

**Parameters***run*

The run from which to return glyphs.

**Return Value**

A valid pointer to an array of CGGlyph structures, or NULL.

**Discussion**

The glyph array will have a length equal to the value returned by [CTRunGetGlyphCount](#) (page 7). The caller should be prepared for this function to return NULL even if there are glyphs in the stream. If this function returns NULL, the caller must allocate its own buffer and call [CTRunGetGlyphs](#) to fetch the glyphs.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CTRun.h

**CTRunGetImageBounds**

Calculates the image bounds for a glyph range.

```
CGRect CTRunGetImageBounds( CTRunRef run, CGContextRef context, CFRange range );
```

**Parameters***run*

The run for which to calculate the image bounds.



*context*

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

*range*

The portion of the run to measure. If the length of the range is set to 0, then the measure operation continues from the start index of the range to the end of the run.

#### **Return Value**

A rectangle that tightly encloses the paths of the run's glyphs, or, if *run*, *context*, or *range* is invalid, `CGRectNull`.

#### **Availability**

Available in Mac OS X v10.5 and later.

#### **Declared In**

`CTRun.h`

## **CTRunGetPositions**

Copies a range of glyph positions into a user-provided buffer.

```
void CTRunGetPositions( CTRunRef run, CFRange range, CGPoint* buffer );
```

#### **Parameters**

*run*

The run from which to copy glyph positions.

*range*

The range of glyph positions to copy. If the length of the range is set to 0, then the copy operation will continue from the start index of the range to the end of the run.

*buffer*

The buffer to which the glyph positions are copied. The buffer must be allocated to at least the value specified by the range's length.

#### **Availability**

Available in Mac OS X v10.5 and later.

#### **Declared In**

`CTRun.h`

## **CTRunGetPositionsPtr**

Returns a direct pointer for the glyph position array stored in the run.

```
const CGPoint* CTRunGetPositionsPtr( CTRunRef run );
```

#### **Parameters**

*run*

The run from which to access glyph positions.

#### **Return Value**

A valid pointer to an array of `CGPoint` structures, or `NULL`.

**Discussion**

The glyph positions in a run are relative to the origin of the line containing the run. The position array will have a length equal to the value returned by [CTRunGetGlyphCount](#) (page 7). The caller should be prepared for this function to return `NULL` even if there are glyphs in the stream. If this function returns `NULL`, the caller must allocate its own buffer and call [CTRunGetPositions](#) (page 9) to fetch the glyph positions.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

`CTRun.h`

**CTRunGetStatus**

Returns the run's status.

```
CTRunStatus CTRunGetStatus( CTRunRef run );
```

**Parameters**

*run*

The run for which to return the status.

**Return Value**

The run's status.

**Discussion**

Runs have status that can be used to expedite certain operations. Knowing the direction and ordering of a run's glyphs can aid in string index analysis, whereas knowing whether the positions reference the identity text matrix can avoid expensive comparisons. This status is provided as a convenience, because this information is not strictly necessary but can be helpful in some circumstances.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

`CTRun.h`

**CTRunGetStringIndices**

Copies a range of string indices into a user-provided buffer.

```
void CTRunGetStringIndices( CTRunRef run, CFRange range, CFIndex* buffer );
```

**Parameters**

*run*

The run from which to copy the string indices.

*range*

The range of string indices to copy. If the length of the range is set to 0, then the copy operation continues from the range's start index to the end of the run.

*buffer*

The buffer to which the string indices are copied. The buffer must be allocated to at least the value specified by the range's length.

**Discussion**

The indices are the character indices that originally spawned the glyphs that make up the run. They can be used to map the glyphs in the run back to the characters in the backing store.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CTRun.h

**CTRunGetStringIndicesPtr**

Returns a direct pointer for the string indices stored in the run.

```
const CFIndex* CTRunGetStringIndicesPtr( CTRunRef run );
```

**Parameters**

*run*

The run for which to return string indices.

**Return Value**

A valid pointer to an array of `CFIndex` structures, or `NULL`.

**Discussion**

The indices are the character indices that originally spawned the glyphs that make up the run. They can be used to map the glyphs in the run back to the characters in the backing store. The string indices array will have a length equal to the value returned by [CTRunGetGlyphCount](#) (page 7). The caller should be prepared for this function to return `NULL` even if there are glyphs in the stream. If this function returns `NULL`, the caller must allocate its own buffer and call [CTRunGetStringIndices](#) (page 10) to fetch the indices.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CTRun.h

**CTRunGetStringRange**

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

```
CFRange CTRunGetStringRange( CTRunRef run );
```

**Parameters**

*run*

The run for which to access the string range.

**Return Value**

The range of characters that originally spawned the glyphs, or if *run* is invalid, an empty range.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CTRun.h

## CTRunGetTextMatrix

Returns the text matrix needed to draw this run.

```
CGAffineTransform CTRunGetTextMatrix( CTRunRef run );
```

### Parameters

*run*

The run object from which to get the text matrix.

### Return Value

A CGAffineTransform structure.

### Discussion

To properly draw the glyphs in a run, the fields `tx` and `ty` of the CGAffineTransform returned by this function should be set to the current text position.

### Availability

Available in Mac OS X v10.5 and later.

### Declared In

CTRun.h

## CTRunGetTypeID

Returns the Core Foundation type identifier of the run object.

```
CFTypeID CTRunGetTypeID( void );
```

### Availability

Available in Mac OS X v10.5 and later.

### Declared In

CTRun.h

## CTRunGetTypographicBounds

Gets the typographic bounds of the run.

```
double CTRunGetTypographicBounds( CTRunRef run, CFRange range, CGFloat* ascent,
CGFloat* descent, CGFloat* leading );
```

### Parameters

*run*

The run for which to calculate the typographic bounds.

*range*

The portion of the run to measure. If the length of the range is set to 0, then the measure operation continues from the range's start index to the end of the run.

*ascent*

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

*descent*

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

*leading*

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

**Return Value**

The typographic width of the run, or if *run* or *range* is invalid, 0.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CTRun.h

## Data Types

**CTRunRef**

A reference to a run object.

```
typedef const struct __CTRun *CTRunRef;
```

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

CTRun.h

## Constants

**CTRunStatus**

A bitfield passed back by the [CTRunGetStatus](#) (page 10) function that is used to indicate the disposition of the run.

```
enum{
kCTRunStatusNoStatus = 0,
kCTRunStatusRightToLeft = (1 << 0),
kCTRunStatusNonMonotonic = (1 << 1),
kCTRunStatusHasNonIdentityMatrix = (1 << 2)
};
typedef uint32_t CTRunStatus;
```

**Constants**

`kCTRunStatusNoStatus`

The run has no special attributes.

Available in Mac OS X v10.5 and later.

Declared in CTRun.h.

`kCTRunStatusRightToLeft`

The run proceeds from right to left.

Available in Mac OS X v10.5 and later.

Declared in `CTRun.h`.

`kCTRunStatusNonMonotonic`

The run has been reordered in some way such that the string indices associated with the glyphs are no longer strictly increasing (for left-to-right runs) or decreasing (for right-to-left runs).

Available in Mac OS X v10.5 and later.

Declared in `CTRun.h`.

`kCTRunStatusHasNonIdentityMatrix`

The run requires a specific text matrix to be set in the current Core Graphics context for proper drawing.

Available in Mac OS X v10.5 and later.

Declared in `CTRun.h`.

**Declared In**

`CTRun.h`

# Document Revision History

---

This table describes the changes to *CTRun Reference*.

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

## REVISION HISTORY

### Document Revision History



# Index

---

## C

---

CTRunDraw **function** [6](#)  
CTRunGetAttributes **function** [7](#)  
CTRunGetGlyphCount **function** [7](#)  
CTRunGetGlyphs **function** [7](#)  
CTRunGetGlyphsPtr **function** [8](#)  
CTRunGetImageBounds **function** [8](#)  
CTRunGetPositions **function** [9](#)  
CTRunGetPositionsPtr **function** [9](#)  
CTRunGetStatus **function** [10](#)  
CTRunGetStringIndices **function** [10](#)  
CTRunGetStringIndicesPtr **function** [11](#)  
CTRunGetStringRange **function** [11](#)  
CTRunGetTextMatrix **function** [12](#)  
CTRunGetTypeID **function** [12](#)  
CTRunGetTypographicBounds **function** [12](#)  
CTRunRef **structure** [13](#)  
CTRunStatus [13](#)

## K

---

kCTRunStatusHasNonIdentityMatrix **constant** [14](#)  
kCTRunStatusNonMonotonic **constant** [14](#)  
kCTRunStatusNoStatus **constant** [13](#)  
kCTRunStatusRightToLeft **constant** [14](#)