Windows Bridge for iOS

**[CoreText]**

**DEV DESIGN specification**

[This document reuses content from a number of online and internal sources. Implicit thanks to the otherwise uncredited original authors and contributors.]

## Table of Contents

[Table of Contents 1](#_Toc471996943)

[Overview 2](#_Toc471996944)

[Approach 2](#_Toc471996945)

[Dependencies 2](#_Toc471996946)

[Technical Details 2](#_Toc471996947)

[Overview 2](#_Toc471996948)

[APIs 4](#_Toc471996949)

[Current CoreText Implementation 4](#_Toc471996950)

[Text Measure APIs 5](#_Toc471996951)

[Text Render APIs 5](#_Toc471996952)

[Implementing CoreText with DirectWrite 5](#_Toc471996953)

[Text Measure APIs 5](#_Toc471996954)

[Text Render APIs 6](#_Toc471996955)

[CTFontDescriptor Reference 6](#_Toc471996956)

[CTFontCollection Reference 8](#_Toc471996957)

[CTFont Reference 9](#_Toc471996958)

[CTTypeSetter Reference 16](#_Toc471996959)

[CTLine Reference 18](#_Toc471996960)

[CTRun Reference 20](#_Toc471996961)

[CTRunDelegate Reference 22](#_Toc471996962)

[CTFrameSetter Reference 22](#_Toc471996963)

[CTFrame Reference 23](#_Toc471996964)

[CTGlyphInfo Reference 24](#_Toc471996965)

[CTParagraphStyle Reference 25](#_Toc471996966)

[CTTextTab Reference 25](#_Toc471996967)

[CGFont Reference 26](#_Toc471996968)

[Working with PostScript Fonts 26](#_Toc471996969)

[Working with Font Tables 26](#_Toc471996970)

[Getting Font Information 27](#_Toc471996971)

[Performance: EndDraw calls 28](#_Toc471996972)

[Appendix 30](#_Toc471996973)

# Overview

This document outlines the long term strategy in providing a consistent hardware-accelerated text rendering platform between CoreText and UIKit framework (which will leverage XAML) by using [Windows DirectWrite](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368038(v=vs.85).aspx).

# Approach

We will begin to convert all of our existing CoreText implementations over to use Windows DirectWrite. Each CoreText API surface will implement the required translation to convert iOS constructs with Windows DirectWrite.

***WinObjC won’t fill gaps in the Windows platform*** to implement all CoreText functionality; instead, any platform gaps will be driven as feature requests to the owning team(s).

## Dependencies

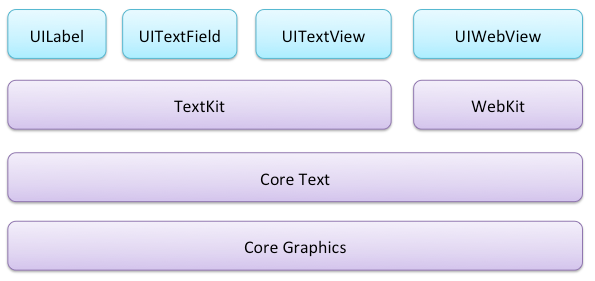
* **DirectWrite:** Direct2D has to be used in order to take advantage of the hardware-accelerated text rendering. Today CoreGraphics framework is implemented over Cairo which only supports software rendering.

# Technical Details

## Overview

Core Text provides a low-level programming interface for laying out text and handling fonts. Though today in iOS, TextKit is the recommended text layout and rendering framework, it has been built on top of Core Text to make use of features to access to font descriptors and collections, font metrics and glyph data, etc.

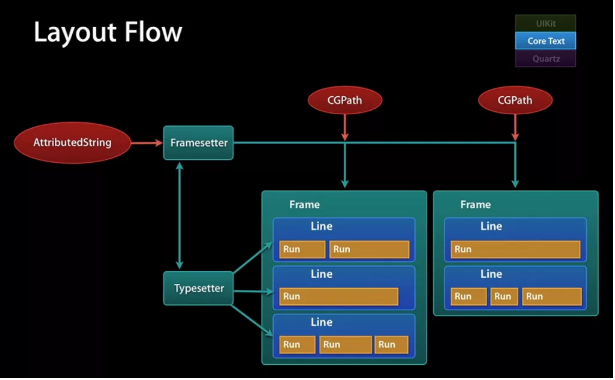
TextKit is a set of classes and protocols in the UIKit framework and it provides high-quality typographical services that enable apps to store, lay out, and display text with all the characteristics of fine typesetting, such as kerning, ligatures, line breaking, and justification. Below is a diagram shows the position of Text Kit among other iOS text and graphics frameworks.



More details on TextKit can be found [here](https://developer.apple.com/library/ios/documentation/StringsTextFonts/Conceptual/TextAndWebiPhoneOS/CustomTextProcessing/CustomTextProcessing.html)

Core Text is a low-level technology for laying out text and handling fonts. Core Text works directly with Core Graphics (CG), also known as Quartz, which is the high-speed graphics rendering engine that handles two-dimensional imaging at the lowest level in iOS. Core Text is designed for development of higher-level text-handling frameworks that need finer control on various text layout and rendering parameters.

The below diagram represents multiple objects which form a hierarchy in figuring out how the text would flow



**AttributedString**

The Core Text layout engine works with attributed strings (CFAttributedStringRef) and graphics paths (CGPathRef). An attributed-string object encapsulates a string backing the displayed text and includes properties (or “attributes”) that define stylistic aspects of the characters in the string—for example, font and color.

**Framesetter**

The Framesetter knows how much text fits into one frame and you can query the frame as to what range from the attributed string fit.

**Typesetter**

The typesetting mechanism in Core Text uses the information in the attributed string to perform character-to-glyph conversion.

**Frame**

For frame to be generated, the framesetter calls a typesetter object. As it lays text out in a frame, the framesetter applies paragraph styles to it, including such attributes as alignment, indentation, etc. The typesetter converts the characters in the attributed string to glyphs and fits those glyphs into the lines that fill a text frame. Frames can draw themselves into a graphics context.

**Line**

Each Frame object contains the paragraph’s line objects. Each line object represents a line of text. A frame object may contain just a single long line object or it might contain a set of lines. Line objects are created by the typesetter during a framesetting operation and, like frames, can draw themselves directly into a graphics context.

**Run**

Each line object contains an array of glyph run objects. A glyph run is a set of consecutive glyphs that share the same attributes and direction. The typesetter creates glyph runs as it produces lines from character strings, attributes, and font objects. This means that a line is constructed of one or more glyphs runs. Glyph runs can also draw themselves into a graphic context.

## APIs

All of CoreText APIs can be broadly classified into two categories –

1. APIs that help with measuring various text rendering layout parameters – from here on called as Text Measure APIs
2. APIs that help with drawing text into a graphics context – from here on called as Text Render APIs

## Current CoreText Implementation

Today CoreText is implemented using FreeType. In the current implementation of CoreText with FreeType, we chose to provide support for low level Glyph access to the rendered text at the cost of implementing the text layout engine our self.

### Text Measure APIs

In our current implementation, all of text measurement is manually handled in WinObjC code i.e. we have implemented all of text layout, line termination, indentation etc in code. This approach has a number of drawbacks –

1. Barely works for English language
2. No support for languages written from right to left or top to bottom
3. No support for most of Windows OpenType fonts
4. Performance issues – say when the font size needs to be updated in real time, the whole text layout logic has to re-run to re-layout text for the measure APIs to pick up the updated values.

### Text Render APIs

Cairo exposes text rendering APIs and because it also uses FreeType to measure and layout text, CoreText today relies on Cairo to render text. Because Cairo is also our 2D rendering engine, CoreText works well with all CoreGraphics APIs even though it is software rendered.

## Implementing CoreText with DirectWrite

Windows DirectWrite API supports measuring, drawing, and hit-testing of multi-format text. DirectWrite also provides a low-level glyph rendering API for developers who want to perform their own layout and Unicode-to-glyph processing. DirectWrite handles text in all supported languages for global and localized applications.

Most of DirectWrite constructs to format and layout text map pretty well with CoreText. This lets us back most of the CoreText objects with DirectWrite object, letting us interact with DirectWrite in the backend without affecting the user facing CoreText APIs. Details of these mapping is discussed in details in each of the reference sections below.

In the below sections each of the CoreText feature areas are mapped to its counterpart in DirectWrite that will be used to implement them in WinObjC.

A table captures the recommended interface mapping between the two platform and calls out specific gaps if any. Below coloring scheme is used to represent the mapping –

|  |  |
| --- | --- |
| **CoreText** | **DirectWrite** |
| Interface1 | No functionally compatible interface is available |
| Interface2 | Interface is not fully compatible functionally |
| Interface3 | A functionally compatible interface is available |

***Refer to Appendix section to get an idea of the most used CoreText APIs that can be interop’d using DirectWrite and their current status.***

### Text Measure APIs

DirectWrite supports custom render which DirectWrite will call with text layout details and this information is sufficient for us to construct the various CoreText objects that developers would use to perform text measurement. DirectWrite also supports all Windows OpenType fonts and languages efficiently and has a very optimized text layout engine for optimal performance.

### Text Render APIs

DirectWrite supports two types of render targets –

1. Direct2D – Hardware Accelerated
2. IWICBitmap – Software based

For CoreText to be to make use of hardware accelerated text rendering, we need to move our CoreGraphics stack to Direct2D. Until we finalize our investigation on this CoreText will make use of IWICBitmap to render text.

On WinObjC IWICBitmap is a basically backed by a buffer to which CoreText blitz its content, so in WinObjC we can back IWICBitmap with the same buffer Cairo is being backed with (i.e. XAML Writable Bitmap) and expose it through CoreGraphics context



### CTFontDescriptor Reference

The CTFontDescriptor opaque type represents a font descriptor, that is, a dictionary of attributes such as name, point size, and variation that can completely specify a font. A font descriptor can be an incomplete specification; in which case the system chooses the most appropriate font to match the given attributes.

Below are the font properties that are supported in CoreText and their mapping with DirectWrite –

|  |  |
| --- | --- |
| **CoreText** | **DirectWrite** |
| kCTFontURLAttribute |  |
| kCTFontNameAttribute |  |
| kCTFontDisplayNameAttribute |  |
| kCTFontFamilyNameAttribute | A wide char array used to look up for a IDWriteFontFamily from IDWriteFontCollection |
| kCTFontStyleNameAttribute |  |
| kCTFontTraitsAttribute  (kCTFontWeightTrait) | DWRITE\_FONT\_WEIGHT    From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368082(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368082(v=vs.85).aspx)>    **Needs manual mapping and no other trait is supported** |
| kCTFontVariationAttribute |  |
| kCTFontSizeAttribute | Set by the application or a default value should be queried or set  **Cannot be used as a key to query for a matching font.** |
| kCTFontMatrixAttribute |  |
| kCTFontCascadeListAttribute |  |
| kCTFontCharacterSetAttribute |  |
| kCTFontLanguagesAttribute |  |
| kCTFontBaselineAdjustAttribute |  |
| kCTFontMacintoshEncodingsAttribute |  |
| kCTFontFeaturesAttribute |  |
| kCTFontFeatureSettingsAttribute |  |
| kCTFontFixedAdvanceAttribute |  |
| kCTFontOrientationAttribute |  |
| kCTFontFormatAttribute |  |
| kCTFontRegistrationScopeAttribute |  |
| kCTFontPriorityAttribute |  |
| kCTFontEnabledAttribute |  |

#### Data Structure

*struct \_\_CTFontDescriptor {*

*CFMutableDictionayRef \_attributes;*

*};*

#### \_attributes = Dictionary of font attributes

#### Creating font descriptors

These APIs provide ability to create font descriptors

Note: Though we can create font descriptors with any attribute, font matching on DirectWrite will only support look-up based on Family Name and font Weight Trait

|  |  |
| --- | --- |
| **CoreText** |  |
| [CTFontDescriptorCreateWithNameAndSize](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontDescriptorRef/index.html#//apple_ref/c/func/CTFontDescriptorCreateWithNameAndSize) | We can just return a newly created font descriptor dictionary (data structure) with name & size attribute values set |
| [CTFontDescriptorCreateWithAttributes](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontDescriptorRef/index.html#//apple_ref/c/func/CTFontDescriptorCreateWithAttributes) | -ditto- |
| [CTFontDescriptorCreateCopyWithAttributes](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontDescriptorRef/index.html#//apple_ref/c/func/CTFontDescriptorCreateCopyWithAttributes) | -ditto- |
| [CTFontDescriptorCreateMatchingFontDescriptors](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontDescriptorRef/index.html#//apple_ref/c/func/CTFontDescriptorCreateMatchingFontDescriptors) | -ditto- |
| [CTFontDescriptorCreateCopyWithVariation](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontDescriptorRef/index.html#//apple_ref/c/func/CTFontDescriptorCreateCopyWithVariation) | -ditto- |
| [CTFontDescriptorCreateCopyWithFeature](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontDescriptorRef/index.html#//apple_ref/c/func/CTFontDescriptorCreateCopyWithFeature) | -ditto- |

#### Getting font descriptors

These APIs provide ability to retrieve font attributes given descriptor

|  |  |
| --- | --- |
| **CoreText** |  |
| [CTFontDescriptorCopyAttributes](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontDescriptorRef/index.html#//apple_ref/c/func/CTFontDescriptorCopyAttributes) | We can perform operations on the font descriptor dictionary |
| [CTFontDescriptorCopyAttribute](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontDescriptorRef/index.html#//apple_ref/c/func/CTFontDescriptorCopyAttribute) | -ditto- |
| [CTFontDescriptorCopyLocalizedAttribute](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontDescriptorRef/index.html#//apple_ref/c/func/CTFontDescriptorCopyLocalizedAttribute) | -ditto- |

#### Takeaway

* DirectWrite today only supports using Family Name and font Weight Trait to find a matching font.
* Font descriptors (attributes) can be maintained in a dictionary data structure and has no real dependency with DirectWrite.

### CTFontCollection Reference

The CTFontCollection opaque type represents a collection of fonts.

#### Data Structure

*struct \_\_CTFontCollection {*

*CFFontDescriptorRef \_attributes;*

*ComPtr<IDWriteFontFamily> \_dwriteFontFamily;*

*};*

#### \_attributes = Dictionary of font attributes that match this font collection

#### \_dwriteFontFamily = DirectWrite font family object corresponding to this font collection

#### Creating font collections

These APIs provide ability to create font collections given descriptors

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCollectionCreateFromAvailableFonts](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontCollectionRef/index.html#//apple_ref/c/func/CTFontCollectionCreateFromAvailableFonts) | IDWriteFactory::GetSystemFontCollection  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368208(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368208(v=vs.85).aspx)>    IDWriteFontCollection interface  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368214(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368214(v=vs.85).aspx)>    IDWriteFontFamily interface  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371042(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371042(v=vs.85).aspx)>  **No support for options** |
| [CTFontCollectionCreateWithFontDescriptors](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontCollectionRef/index.html#//apple_ref/c/func/CTFontCollectionCreateWithFontDescriptors) | IDWriteFactory::GetSystemFontCollection  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368208(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368208(v=vs.85).aspx)>    IDWriteFontCollection::FindFamilyName  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368217(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368217(v=vs.85).aspx)>    IDWriteFontFamily::GetFirstMatchingFont  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371051(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371051(v=vs.85).aspx)>  **Only kCTFontFamilyNameAttribute and kCTFontTraitsAttribute (kCTFontWeightTrait) is supported** |
| [CTFontCollectionCreateCopyWithFontDescriptors](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontCollectionRef/index.html#//apple_ref/c/func/CTFontCollectionCreateCopyWithFontDescriptors) | -ditto- |

#### Getting font descriptors

These APIs given font collection returns their descriptors

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCollectionCreateMatchingFontDescriptors](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontCollectionRef/index.html#//apple_ref/c/func/CTFontCollectionCreateMatchingFontDescriptors) | Return CTFontDescriptor associated with the font collection |
| [CTFontCollectionCreateMatchingFontDescriptorsSortedWithCallback](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontCollectionRef/index.html#//apple_ref/c/func/CTFontCollectionCreateMatchingFontDescriptorsSortedWithCallback) | Return CTFontDescriptor associated with the font collection, sorted by the user defined callback |
| [CTFontCollectionSortDescriptorsCallback](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontCollectionRef/index.html#//apple_ref/c/func/CTFontCollectionSortDescriptorsCallback) | Callback caller can implement to sort CTFontDescriptor |

#### Takeaway

* CTFontCollection maps to [IDWriteFontFamily interface](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371042(v=vs.85).aspx) on Windows

### CTFont Reference

The CTFont opaque type represents a Core Text font object. Font objects represent fonts to an application, providing access to characteristics of the font, such as point size, and other attributes.

#### Data Structure

*struct \_\_CTFont {*

*CTFontDescriptorRef \_attributes;*

*ComPtr<IDWriteFont> \_dwriteFont;*

*ComPtr<IDWriteFontFace> \_dwriteFontFace;*

*};*

#### \_attributes = Dictionary of font attributes that match this font

#### \_dwriteFont = DirectWrite font object corresponding to this font

#### \_dwriteFontFace = DirectWrite font face object corresponding to this font

#### Creating fonts

These APIs provide ability to create fonts objects with font/family names and font descriptors.

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCreateWithName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateWithName) |  |
| [CTFontCreateWithNameAndOptions](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateWithNameAndOptions) |  |
| [CTFontCreateWithFontDescriptor](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateWithFontDescriptor) | IDWriteFactory::GetSystemFontCollection  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368208(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368208(v=vs.85).aspx)>    IDWriteFontCollection::FindFamilyName  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368217(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368217(v=vs.85).aspx)>  IDWriteFontFamily::GetFirstMatchingFont  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371051(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371051(v=vs.85).aspx)>    **Only kCTFontFamilyNameAttribute and kCTFontTraitsAttribute (kCTFontWeightTrait) is supported**  **Not all arguments are supported - affine transformation matrix** |
| [CTFontCreateWithFontDescriptorAndOptions](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateWithFontDescriptorAndOptions) | -ditto-    **Only** kCTFontOptionsPreferSystemFont **option is supported**  **Not all arguments are supported - affine transformation matrix** |
| [CTFontCreateUIFontForLanguage](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateUIFontForLanguage) |  |
| [CTFontCreateCopyWithAttributes](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateCopyWithAttributes) | IDWriteFactory::GetSystemFontCollection  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368208(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368208(v=vs.85).aspx)>    IDWriteFontCollection::FindFamilyName  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368217(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368217(v=vs.85).aspx)>  IDWriteFontFamily::GetFirstMatchingFont  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371051(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371051(v=vs.85).aspx)>  **Only kCTFontFamilyNameAttribute and kCTFontTraitsAttribute (kCTFontWeightTrait) is supported**  **Not all arguments are supported - affine transformation matrix** |
| [CTFontCreateCopyWithSymbolicTraits](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateCopyWithSymbolicTraits) | -ditto-    **Not all symbolic traits are supported**  **Not all arguments are supported - affine transformation matrix** |
| [CTFontCreateCopyWithFamily](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateCopyWithFamily) | -ditto-  **Not all arguments are supported - affine transformation matrix** |
| [CTFontCreateForString](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateForString) |  |

#### Getting font data

These APIs provide ability to font descriptors and attributes given the font object

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCopyFontDescriptor](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyFontDescriptor) | Return CTFontDescriptor associated with the font collection |
| [CTFontCopyAttribute](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyAttribute) | Return a copy of the attributed from the associated CTFontDescriptor |
| [CTFontGetSize](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetSize) | Return CTFontDescriptor associated with the font collection |
| [CTFontGetMatrix](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetMatrix) |  |
| [CTFontGetSymbolicTraits](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetSymbolicTraits) | Return kCTFontTraitsAttribute from the CTFontDescriptor associated with the font collection |
| [CTFontCopyTraits](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyTraits) | Return a copy of the kCTFontTraitsAttribute from the CTFontDescriptor associated with the font collection |

#### Getting font names

Provides APIs to retrieve various font names like post script, family names with localization

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCopyPostScriptName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyPostScriptName) | IDWriteFont::GetInformationalStrings  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371147(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371147(v=vs.85).aspx)>    DWRITE\_INFORMATIONAL\_STRING\_ID enumeration  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368094(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368094(v=vs.85).aspx)> |
| [CTFontCopyFamilyName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyFamilyName) | IDWriteFont::GetFontFamily  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371143(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371143(v=vs.85).aspx)> |
| [CTFontCopyFullName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyFullName) | IDWriteFont::GetInformationalStrings  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371147(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371147(v=vs.85).aspx)>    DWRITE\_INFORMATIONAL\_STRING\_ID enumeration  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368094(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368094(v=vs.85).aspx)> |
| [CTFontCopyDisplayName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyDisplayName) | **We should be able to use the full name as display name** |
| [CTFontCopyName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyName) | IDWriteFont::GetInformationalStrings  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371147(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371147(v=vs.85).aspx)>    DWRITE\_INFORMATIONAL\_STRING\_ID enumeration  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368094(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368094(v=vs.85).aspx)> |
| [CTFontCopyLocalizedName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyLocalizedName) | -ditto- |

#### Working with encoding

Provides APIs to retrieve character set, string encoding and language support

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCopyCharacterSet](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyCharacterSet) | IDWriteFont1::GetUnicodeRanges method  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/hh780407(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/hh780407(v=vs.85).aspx)> |
| [CTFontGetStringEncoding](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetStringEncoding) | Always return **kCFStringEncodingUnicode** |
| [CTFontCopySupportedLanguages](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopySupportedLanguages) | IDWriteFont3::GetFontFaceReference  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dn890771(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dn890771(v=vs.85).aspx)>    IDWriteFontSetBuilder::AddFontFaceReference  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dn933237(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dn933237(v=vs.85).aspx)>    IDWriteFontSetBuilder::CreateFontSet  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dn933241(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dn933241(v=vs.85).aspx)>    IDWriteFontSet::GetPropertyValues  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dn933252(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dn933252(v=vs.85).aspx)>    DWRITE\_FONT\_PROPERTY\_ID enumeration  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dn933213(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dn933213(v=vs.85).aspx)>    DWRITE\_FONT\_PROPERTY\_ID\_SUPPORTED\_SCRIPT\_LANGUAGE\_TAG |

#### Getting font metrics

Provides APIs to retrieve various font metrics

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontGetAscent](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetAscent) | IDWriteFont::GetMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371149(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371149(v=vs.85).aspx)>    DWRITE\_FONT\_METRICS structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368074(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368074(v=vs.85).aspx)> |
| [CTFontGetDescent](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetDescent) | -ditto- |
| [CTFontGetLeading](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetLeading) | DWRITE\_LINE\_METRICS1 structure  From <*https://msdn.microsoft.com/en-us/library/windows/desktop/dn933215(v=vs.85).aspx*> |
| [CTFontGetUnitsPerEm](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetUnitsPerEm) | IDWriteFont::GetMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371149(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371149(v=vs.85).aspx)>    DWRITE\_FONT\_METRICS structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368074(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368074(v=vs.85).aspx)> |
| [CTFontGetGlyphCount](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetGlyphCount) | IDWriteFontFace::GetGlyphCount  *From* < <https://msdn.microsoft.com/en-us/library/windows/desktop/dd370993(v=vs.85).aspx>> |
| [CTFontGetBoundingBox](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetBoundingBox) | IDWriteFont1::GetMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/hh780405(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/hh780405(v=vs.85).aspx)>    DWRITE\_FONT\_METRICS1 structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/jj126259(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/jj126259(v=vs.85).aspx)>  **Can be manually calculated** |
| [CTFontGetUnderlinePosition](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetUnderlinePosition) | IDWriteFont::GetMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371149(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371149(v=vs.85).aspx)>    DWRITE\_FONT\_METRICS structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368074(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368074(v=vs.85).aspx)> |
| [CTFontGetUnderlineThickness](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetUnderlineThickness) | -ditto- |
| [CTFontGetSlantAngle](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetSlantAngle) | IDWriteFontFace1::GetCaretMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/hh780410(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/hh780410(v=vs.85).aspx)>    DWRITE\_CARET\_METRICS structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/jj126258(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/jj126258(v=vs.85).aspx)> |
| [CTFontGetCapHeight](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetCapHeight) | IDWriteFont::GetMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371149(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371149(v=vs.85).aspx)>    DWRITE\_FONT\_METRICS structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368074(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368074(v=vs.85).aspx)> |
| [CTFontGetXHeight](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetXHeight) | -ditto- |

#### Getting glyph data

Provides APIs to retrieve various glyph data

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCreatePathForGlyph](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreatePathForGlyph) | Once our CoreGraphics stack moves to D2D, use  IDWriteFontFace::GetGlyphRunOutline  From < <https://msdn.microsoft.com/en-us/library/windows/desktop/dd371003(v=vs.85).aspx>>  **Without D2D, we need code to manually convert** IDWriteGeometrySink to CGPath |
| [CTFontGetGlyphWithName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetGlyphWithName) |  |
| [CTFontGetBoundingRectsForGlyphs](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetBoundingRectsForGlyphs) | IDWriteFont1::GetMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/hh780405(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/hh780405(v=vs.85).aspx)>    DWRITE\_FONT\_METRICS1 structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/jj126259(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/jj126259(v=vs.85).aspx)>  **Can be manually calculated but will not be very accurate** |
| [CTFontGetAdvancesForGlyphs](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetAdvancesForGlyphs) | IDWriteFont::CreateFontFace  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371137(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371137(v=vs.85).aspx)>    IDWriteFontFace1::GetDesignGlyphAdvances  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/hh780411(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/hh780411(v=vs.85).aspx)> |
| [CTFontGetVerticalTranslationsForGlyphs](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetVerticalTranslationsForGlyphs) |  |

#### Working with font variations

Provides APIs to retrieve font variations

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCopyVariationAxes](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyVariationAxes) |  |
| [CTFontCopyVariation](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyVariation) |  |

#### Getting font features

Provides APIs to retrieve font features

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCopyFeatures](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyFeatures) |  |
| [CTFontCopyFeatureSettings](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyFeatureSettings) |  |

#### Working with glyphs

Provides APIs to work with glyphs

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontGetGlyphsForCharacters](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetGlyphsForCharacters) | IDWriteFont::CreateFontFace  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371137(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371137(v=vs.85).aspx)>    IDWriteFontFace::GetGlyphIndices  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd370998(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370998(v=vs.85).aspx)>    IDWriteFontFace::GetDesignGlyphMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd370986(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370986(v=vs.85).aspx)> |
| [CTFontDrawGlyphs](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontDrawGlyphs) | Supports rendering to custom renderer, bitmap or D2D    Rendering DirectWrite  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/ff485855(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/ff485855(v=vs.85).aspx)>  **For hardware accelerated rendering need to move CoreGraphics to D2D**  **Bitmap rendering will be used for now and it makes use of WARP and is software accelerated** |
| [CTFontGetLigatureCaretPositions](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetLigatureCaretPositions) |  |

#### Converting fonts

Provides APIs to convert fonts to graphic fonts with attributes

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCopyGraphicsFont](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyGraphicsFont) |  |
| [CTFontCreateWithGraphicsFont](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCreateWithGraphicsFont) |  |

#### Getting font table data

Provides APIs to retrieve font table data

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFontCopyAvailableTables](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyAvailableTables) | DWriteFontFace::TryGetFontTable  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371039(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371039(v=vs.85).aspx)>  **Need to manually iterate over all tables?** |
| [CTFontCopyTable](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyTable) | -ditto- |

#### Takeaway

* CTFont maps to [IDWriteFont](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368213(v=vs.85).aspx) and [IDWriteFontFace interface](https://msdn.microsoft.com/en-us/library/windows/desktop/dd370983(v=vs.85).aspx) on Windows
* Below are the rendering options available –
  + To take advantage of hardware accelerated rendering, CoreGraphics needs to use Direct 2D
  + Text can be rendered to a bitmap and the bitmap buffer is backed by the XAML Bitmap buffer that is currently used by Cairo for rendering.

### CTTypeSetter Reference

The CTTypesetter opaque type represents a typesetter, which performs line layout.

A typesetter object takes as input an attributed string and produces a line of typeset glyphs (composed into glyph runs) in a CTLine object.

#### Data Structure

*struct \_\_CTTypeSetter {*

*CFAttributedStringRef \_string;*

*CTFontRef \_font;*

*ComPtr<IDWriteTextFormat> \_dwriteTextFormat;*

*ComPtr<IDWriteTextLayout> \_dwriteTextLayout;*

*CTLineRef \_line;*

*};*

#### \_string = Attributed string corresponding to this type setter

#### \_font = CTFont object corresponding to this type setter

#### \_dwriteTextFormat = DirectWrite text formatter object corresponding to this font setter

#### \_dwriteTextLayout = DirectWrite text layout object corresponding to this font setter

#### \_line = CTLine object that can be created from this font setter

#### Creating a Typesetter

Provides APIs to create a typesetter with attributed strings

**Note:** Font details (CTFont object) can be obtained from the string’s (CFAttributedStringRef) attribute - kCTFontAttributeName

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTTypesetterCreateWithAttributedString](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterCreateWithAttributedString) | IDWriteFactory::CreateTextFormat  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368203(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368203(v=vs.85).aspx)>    IDWriteFactory::CreateTextLayout  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368205(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368205(v=vs.85).aspx)>  **Not all String attributes can be supported E.g. Paragraph styling attributes.** |
| [CTTypesetterCreateWithAttributedStringAndOptions](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterCreateWithAttributedStringAndOptions) | -ditto-    [**Options not supported**](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/doc/constant_group/Typesetter_Options) |

#### Creating Lines

Provides APIs to create a CTLine objects from a CTTypesetter

In order to get the glyph run details that constitutes a CTLine, a [custom text renderer](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371523(v=vs.85).aspx) needs to be implemented and [DrawGlyphRun method](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371526(v=vs.85).aspx) would provide the glyph run details required to construct the CTLine object.

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTTypesetterCreateLine](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterCreateLine) | IDWriteFactory::CreateTextFormat  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368203(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368203(v=vs.85).aspx)>    IDWriteFactory::CreateTextLayout  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368205(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368205(v=vs.85).aspx)>    IDWriteTextLayout::GetLineMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd316763(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd316763(v=vs.85).aspx)>    IDWriteTextLayout::Draw  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd316726(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd316726(v=vs.85).aspx)>    IDWriteTextRenderer::DrawGlyphRun  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371526(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371526(v=vs.85).aspx)> |
| [CTTypesetterCreateLineWithOffset](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterCreateLineWithOffset) | -Ditto-    **Offset not supported** |

Below is the API call sequence to construct a CTLine object



#### Breaking Lines

Provides APIs to return text index where line breaks have to occur for a given width

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTTypesetterSuggestLineBreak](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterSuggestLineBreak) | IDWriteFactory::CreateTextFormat  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368203(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368203(v=vs.85).aspx)>    IDWriteFactory::CreateTextLayout  From <<https://msdn.microsoft.com/en-us/library/windows/desktop/dd368205(v=vs.85).aspx>>  IDWriteTextLayout::GetLineMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd316763(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd316763(v=vs.85).aspx)> |
| [CTTypesetterSuggestLineBreakWithOffset](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterSuggestLineBreakWithOffset) | -ditto- |
| [CTTypesetterSuggestClusterBreak](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterSuggestClusterBreak) | IDWriteTextLayout::GetClusterMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd316729(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd316729(v=vs.85).aspx)> |
| [CTTypesetterSuggestClusterBreakWithOffset](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterSuggestClusterBreakWithOffset) | -ditto- |

#### Takeaway

* CTTypeSetter maps to [IDWriteTextFormat](https://msdn.microsoft.com/en-us/library/windows/desktop/dd316628(v=vs.85).aspx) and [IDWriteTextLayout](https://msdn.microsoft.com/en-us/library/windows/desktop/dd316718(v=vs.85).aspx) interface on Windows
* Need to implement a custom renderer interface to capture all glyph run details that constitutes a line. Note: The render interface actually does not render anything.

### CTLine Reference

The CTLine opaque type represents a line of text. A CTLine object contains an array of glyph runs.

#### Data Structure

*struct \_\_CTLine {*

*CTTypeSetterRef \_typeSetter;*

*CTMutableArrayRef \_runs;*

*};*

#### \_typeSetter = Typesetter object corresponding to this line

#### \_runs = Array of glyph runs that make this line

#### Creating lines

Provides APIs to create a line with attributed strings without creating typesetters.

**Note:** We will use a typesetter object to create a line object under the cover

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTLineCreateWithAttributedString](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineCreateWithAttributedString) | [CTTypesetterCreateWithAttributedString](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterCreateWithAttributedString)  [CTTypesetterCreateLine](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterCreateLine)  **Not all String attributes can be supported E.g. Paragraph styling attributes.** |
| [CTLineCreateTruncatedLine](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineCreateTruncatedLine) |  |
| [CTLineCreateJustifiedLine](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineCreateJustifiedLine) |  |

#### Drawing the line

Provides API to draw a line into the provided CoreGraphics context

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTLineDraw](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineDraw) | IDWriteFactory::CreateTextLayout  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368205(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368205(v=vs.85).aspx)>    IDWriteTextLayout::Draw  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd316726(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd316726(v=vs.85).aspx)>  IDWriteTextRenderer::DrawGlyphRun  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371526(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371526(v=vs.85).aspx)>  **Need implementing a custom renderer to get individual glyph runs**    Rendering DirectWrite  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/ff485855(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/ff485855(v=vs.85).aspx)>  **For hardware accelerated rendering need to move CoreGraphics to D2D**  **Bitmap rendering will be used for now and it makes use of WARP and is software accelerated** |

#### Getting line data

Provides API to get various line related information like glyph details

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTLineGetGlyphCount](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineGetGlyphCount) | Access the CTRun details that constitutes the line and count the number of glyphs from it |
| [CTLineGetGlyphRuns](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineGetGlyphRuns) | -ditto- |
| [CTLineGetStringRange](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineGetStringRange) | -ditto- |
| [CTLineGetPenOffsetForFlush](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineGetPenOffsetForFlush) |  |

#### Measuring lines

Provides APIs to get various line bounds

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTLineGetImageBounds](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineGetImageBounds) |  |
| [CTLineGetTypographicBounds](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineGetTypographicBounds) | **Can be manually calculated based on the glyph runs that constitutes the line.** |
| [CTLineGetTrailingWhitespaceWidth](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineGetTrailingWhitespaceWidth) | IDWriteTextLayout::GetLineMetrics  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd316763(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd316763(v=vs.85).aspx)> |

#### Getting line positions

Provides APIs to help perform hit testing

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTLineGetStringIndexForPosition](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineGetStringIndexForPosition) | IDWriteTextLayout::HitTestPoint    From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371464(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371464(v=vs.85).aspx)> |
| [CTLineGetOffsetForStringIndex](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTLineRef/index.html#//apple_ref/c/func/CTLineGetOffsetForStringIndex) | IDWriteTextLayout::HitTestTextPosition    From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371469(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371469(v=vs.85).aspx)> |

#### Takeaway

* CTLine object gets created from a CTTypeSetter interface
* Line bounds calculation needs to be manually performed with ?!?

### CTRun Reference

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

CTRun object is created when CTTypeSetter is used to produce lines from character strings, attributes and font objects.

#### Data Structure

*struct \_\_CTRun {*

*CTTypeSetterRef \_typeSetter;*

*DWRITE\_GLYPH\_RUN \_dwriteGlyphRun;*

*DWRITE\_GLYPH\_RUN\_DESCRIPTION \_dwriteGlyphDescription;*

*};*

#### \_typeSetter = Typesetter object corresponding to this glyph run

#### \_dwriteGlyphRun = DirectWrite glyph structure corresponding to this glyph run

#### \_dwriteGlyphDescription = DirectWrite glyph description structure corresponding to this glyph run

#### Getting glyph run data

Provides APIs to get glyph related data/metrics.

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTRunGetGlyphCount](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetGlyphCount) | DWRITE\_GLYPH\_RUN structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368089(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368089(v=vs.85).aspx)> |
| [CTRunGetAttributes](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetAttributes) | Return the CTFontDescriptor object members  CTTypeSetterRef -> CTFontRef -> CTFontDescriptorRef    **Not all members are supported** |
| [CTRunGetStatus](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetStatus) | DWRITE\_GLYPH\_RUN structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368089(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368089(v=vs.85).aspx)>    **Not all status can be provided** |
| [CTRunGetGlyphsPtr](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetGlyphsPtr) |  |
| [CTRunGetGlyphs](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetGlyphs) |  |
| [CTRunGetPositionsPtr](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetPositionsPtr) |  |
| [CTRunGetPositions](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetPositions) |  |
| [CTRunGetAdvancesPtr](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetAdvancesPtr) |  |
| [CTRunGetAdvances](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetAdvances) | DWRITE\_GLYPH\_RUN\_DESCRIPTION structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368091(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368091(v=vs.85).aspx)> |
| [CTRunGetStringIndicesPtr](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetStringIndicesPtr) | -ditto- |
| [CTRunGetStringIndices](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetStringIndices) | -ditto- |
| [CTRunGetStringRange](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetStringRange) | -ditto- |

#### Measuring glyph runs

Provides APIs to get various glyph run bounds

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTRunGetTypographicBounds](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetTypographicBounds) | DWRITE\_GLYPH\_RUN\_DESCRIPTION structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368091(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368091(v=vs.85).aspx)>  **Calculating bounds can be done manually** |
| [CTRunGetImageBounds](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetImageBounds) | DWRITE\_GLYPH\_RUN\_DESCRIPTION structure  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368091(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368091(v=vs.85).aspx)>  **Calculating bounds can be done manually** |

#### Drawing the glyph run

Provides API to draw a glyph run into the provided CoreGraphics context

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTRunDraw](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunDraw) | IDWriteTextLayout::Draw  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd316726(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd316726(v=vs.85).aspx)>  **Need implementing a custom renderer to get individual glyph runs**    Rendering DirectWrite  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/ff485855(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/ff485855(v=vs.85).aspx)>    **Bitmap rendering will be used for now and it makes use of WARP and is software accelerated** |
| [CTRunGetTextMatrix](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTRunRef/index.html#//apple_ref/c/func/CTRunGetTextMatrix) |  |

#### Takeaway

* DWrite does not provide access to individual glyphs inside a glyph run.

### CTRunDelegate Reference

The CTRunDelegate opaque type represents a run delegate, which is assigned to a run (attribute range) to control typographic traits such glyph ascent, glyph descent, and glyph width. The callbacks defined for CTRunDelegate objects are provided by the owner of a run delegate and are used to modify glyph metrics during layout.

Windows does not support modifying glyph metrics during layout

### CTFrameSetter Reference

The CTFramesetter opaque type is used to generate text frames. The framesetter takes an attributed string object and a shape descriptor object and calls into the typesetter to create line objects that fill that shape. The output is a frame object containing an array of lines.

#### Data Structure

*struct \_\_CTFrameSetter {*

*CTTypeSetterRef \_typeSetter;*

*};*

#### \_typeSetter = Typesetter object corresponding to this frame setter

#### Creating a FrameSetter

Provides API to create a CTFrameSetter to use

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| CTFramesetterCreateWithAttributedString | [CTTypesetterCreateWithAttributedString](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTypesetterRef/index.html#//apple_ref/c/func/CTTypesetterCreateWithAttributedString)  **Not all String attributes can be supported** |

#### Creating frames

Provides APIs to create a CTFrame objects

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFramesetterCreateFrame](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFramesetterRef/index.html#//apple_ref/c/func/CTFramesetterCreateFrame) | Create a collection of CTFrame object comprising of a collection of CTLine objects based on the calculated width for each line    **Frame attributes can be supported – kCTFramePathClippingPathAttributeName, kCTFramePathFillRuleAttributeName** |
| [CTFramesetterGetTypesetter](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFramesetterRef/index.html#//apple_ref/c/func/CTFramesetterGetTypesetter) | Return the CTTypesetter object |

#### Get framing size

Provides API to determines the frame size needed for a string range

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFramesetterSuggestFrameSizeWithConstraints](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFramesetterRef/index.html#//apple_ref/c/func/CTFramesetterSuggestFrameSizeWithConstraints) | Manually generate a Frame layout based on the provided constraints?? |

### CTFrame Reference

The CTFrame opaque type represents a frame containing multiple lines of text. The frame object contains an array of line objects that can be retrieved for individual rendering or to get glyph information.

#### Data Structure

*struct \_\_CTFrame {*

*CTFrameSetterRef \_frameSetter;*

*CFMutableArrayRef \_lines;*

*};*

#### \_frameSetter = Framesetter object used to create this frame

#### \_lines = Array of CTLine objects that form this frame

#### Getting frame data

Provides APIs to get frame information

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFrameGetStringRange](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFrameRef/index.html#//apple_ref/c/func/CTFrameGetStringRange) | Can be determined from all CTLine objects that form the frame |
| [CTFrameGetVisibleStringRange](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFrameRef/index.html#//apple_ref/c/func/CTFrameGetVisibleStringRange) | Can be determined from all CTLine objects that form the frame |
| [CTFrameGetPath](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFrameRef/index.html#//apple_ref/c/func/CTFrameGetPath) |  |
| [CTFrameGetFrameAttributes](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFrameRef/index.html#//apple_ref/c/func/CTFrameGetFrameAttributes) | Return the CTTypesetter object contents  CTFrameSetter -> CTTypeSetter    **Frame attributes can be supported – kCTFramePathClippingPathAttributeName, kCTFramePathFillRuleAttributeName** |

#### Getting lines

Provides APIs to get line information

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFrameGetLines](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFrameRef/index.html#//apple_ref/c/func/CTFrameGetLines) | Return the CTLine objects that form the frame |
| [CTFrameGetLineOrigins](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFrameRef/index.html#//apple_ref/c/func/CTFrameGetLineOrigins) | Can be determined from all CTLine objects that form the frame    **Calculating origins can be done manually** |

#### Draw frames

Provides API to draw frame

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTFrameDraw](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFrameRef/index.html#//apple_ref/c/func/CTFrameDraw) | Can use the collection of CTLine objects to draw individual lines  CTLineDraw |

### CTGlyphInfo Reference

#### The CTGlyphInfo opaque type enables you to override a font's specified mapping from Unicode to the glyph ID

Windows DirectWrite does not support this feature

#### Creating GlyphInfo objects

Provides APIs to create a GlyphInfo

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTGlyphInfoCreateWithGlyphName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTGlyphInfoRef/index.html#//apple_ref/c/func/CTGlyphInfoCreateWithGlyphName) |  |
| [CTGlyphInfoCreateWithGlyph](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTGlyphInfoRef/index.html#//apple_ref/c/func/CTGlyphInfoCreateWithGlyph) |  |
| [CTGlyphInfoCreateWithCharacterIdentifier](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTGlyphInfoRef/index.html#//apple_ref/c/func/CTGlyphInfoCreateWithCharacterIdentifier) |  |

*Getting GlyphInfo data*

Provides APIs to get a GlyphInfo data

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTGlyphInfoGetGlyphName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTGlyphInfoRef/index.html#//apple_ref/c/func/CTGlyphInfoGetGlyphName) |  |
| [CTGlyphInfoGetCharacterIdentifier](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTGlyphInfoRef/index.html#//apple_ref/c/func/CTGlyphInfoGetCharacterIdentifier) |  |
| [CTGlyphInfoGetCharacterCollection](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTGlyphInfoRef/index.html#//apple_ref/c/func/CTGlyphInfoGetCharacterCollection) |  |

### CTParagraphStyle Reference

The CTParagraphStyle opaque type represents paragraph or ruler attributes in an attributed string.

A paragraph style object represents a complex attribute value in an attributed string, storing a number of subattributes that affect paragraph layout for the characters of the string. Among these subattributes are alignment, tab stops, writing direction, line-breaking mode, and indentation settings.

Windows DirectWrite does not support this feature.

*Creating Paragraph styles*

Provides APIs to create a paragraph styles

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTParagraphStyleCreate](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTParagraphStyleRef/index.html#//apple_ref/c/func/CTParagraphStyleCreate) |  |
| [CTParagraphStyleCreateCopy](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTParagraphStyleRef/index.html#//apple_ref/c/func/CTParagraphStyleCreateCopy) |  |

*Getting paragraph styles values*

Provides API to get paragraph style

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTParagraphStyleGetValueForSpecifier](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTParagraphStyleRef/index.html#//apple_ref/c/func/CTParagraphStyleGetValueForSpecifier) |  |

### CTTextTab Reference

The CTTextTab opaque type represents a tab in a paragraph style, storing an alignment type and location.

#### Data Structure

*struct \_\_CTTextTab {*

[*CTTextAlignment*](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTParagraphStyleRef/index.html#//apple_ref/doc/c_ref/CTTextAlignment)*\_alignment;*

*};*

#### \_alignment = Text alignment to support

*Creating text tabs*

Provides APIs to create tab text

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTTextTabCreate](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTextTabRef/index.html#//apple_ref/c/func/CTTextTabCreate) | IDWriteFactory::CreateTextFormat   From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd368203(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd368203(v=vs.85).aspx)>    **Only alignment property can be used** |

*Getting text tab data*

Provides API to get text tab data

|  |  |
| --- | --- |
| **CoreText** | **DWrite** |
| [CTTextTabGetAlignment](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTextTabRef/index.html#//apple_ref/c/func/CTTextTabGetAlignment) | Return alignment information |
| [CTTextTabGetLocation](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTextTabRef/index.html#//apple_ref/c/func/CTTextTabGetLocation) |  |
| [CTTextTabGetOptions](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTTextTabRef/index.html#//apple_ref/c/func/CTTextTabGetOptions) |  |

### CGFont Reference

CoreGraphics provides a set of character glyphs and layout information for drawing text. These is no real support to draw text using CoreGraphics frameworks and most of the APIs have been deprecated. CGFont seems to be the lone class that is left out now

CGFont APIs share a striking similarity to CTFont APIs, so we should be able to share the underlying DWrite interface between the two classes. Note: We cannot back CGFont with CTFont as this would put a cisrcular dependency between CoreGraphics and CoreText.

*Creating CGFont object*

|  |  |
| --- | --- |
| **CoreGraphics** | **DWrite** |
| [CGFontCreateWithDataProvider](https://developer.apple.com/reference/coregraphics/1396367-cgfontcreatewithdataprovider?language=objc) |  |
| [CGFontCreateWithFontName](https://developer.apple.com/reference/coregraphics/1396330-cgfontcreatewithfontname?language=objc) | Similar to calling CTFontCreate\* with attribute kCTFontNameAttribute |
| [CGFontCreateCopyWithVariations](https://developer.apple.com/reference/coregraphics/1396373-cgfontcreatecopywithvariations?language=objc) | Regular CreateCopy |

### *Working with PostScript Fonts*

|  |  |
| --- | --- |
| **CoreGraphics** | **CoreText** |
| [CGFontCopyPostScriptName](https://developer.apple.com/reference/coregraphics/1396346-cgfontcopypostscriptname?language=objc) | [CTFontCopyPostScriptName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyPostScriptName) |
| [CGFontCanCreatePostScriptSubset](https://developer.apple.com/reference/coregraphics/1396365-cgfontcancreatepostscriptsubset?language=objc) |  |
| [CGFontCreatePostScriptSubset](https://developer.apple.com/reference/coregraphics/1396324-cgfontcreatepostscriptsubset?language=objc) |  |
| [CGFontCreatePostScriptEncoding](https://developer.apple.com/reference/coregraphics/1396348-cgfontcreatepostscriptencoding?language=objc) |  |

### *Working with Font Tables*

|  |  |
| --- | --- |
| **CoreGraphics** | **CoreText** |
| [CGFontCopyTableTags](https://developer.apple.com/reference/coregraphics/1396392-cgfontcopytabletags?language=objc) | DWriteFontFace::TryGetFontTable  From <[*https://msdn.microsoft.com/en-us/library/windows/desktop/dd371039(v=vs.85).aspx*](https://msdn.microsoft.com/en-us/library/windows/desktop/dd371039(v=vs.85).aspx)>  **Need to manually iterate over all tables?** |
| [CGFontCopyTableForTag](https://developer.apple.com/reference/coregraphics/1396402-cgfontcopytablefortag?language=objc) | -ditto- |

### *Getting Font Information*

|  |  |
| --- | --- |
| **CoreGraphics** | **CoreText** |
| [CGFontCopyVariationAxes](https://developer.apple.com/reference/coregraphics/1396376-cgfontcopyvariationaxes?language=objc) |  |
| [CGFontCopyVariations](https://developer.apple.com/reference/coregraphics/1396355-cgfontcopyvariations?language=objc) |  |
| [CGFontCopyFullName](https://developer.apple.com/reference/coregraphics/1396357-cgfontcopyfullname?language=objc) | [CTFontCopyFullName](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontCopyFullName) |
| [CGFontGetAscent](https://developer.apple.com/reference/coregraphics/1396359-cgfontgetascent?language=objc) | [CTFontGetAscent](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetAscent) |
| [CGFontGetDescent](https://developer.apple.com/reference/coregraphics/1396351-cgfontgetdescent?language=objc) | [CTFontGetDescent](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetDescent) |
| [CGFontGetLeading](https://developer.apple.com/reference/coregraphics/1396390-cgfontgetleading?language=objc) | [CTFontGetLeading](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetLeading) |
| [CGFontGetCapHeight](https://developer.apple.com/reference/coregraphics/1396338-cgfontgetcapheight?language=objc) | [CTFontGetCapHeight](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetCapHeight) |
| [CGFontGetXHeight](https://developer.apple.com/reference/coregraphics/1396410-cgfontgetxheight?language=objc) | [CTFontGetXHeight](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetXHeight) |
| [CGFontGetFontBBox](https://developer.apple.com/reference/coregraphics/1396353-cgfontgetfontbbox?language=objc) |  |
| [CGFontGetItalicAngle](https://developer.apple.com/reference/coregraphics/1396404-cgfontgetitalicangle?language=objc) | [CTFontGetSlantAngle](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetSlantAngle) |
| [CGFontGetStemV](https://developer.apple.com/reference/coregraphics/1396380-cgfontgetstemv?language=objc) |  |
| [CGFontGetGlyphBBoxes](https://developer.apple.com/reference/coregraphics/1396342-cgfontgetglyphbboxes?language=objc) | [CTFontGetBoundingBox](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetBoundingBox) |
| [CGFontGetGlyphWithGlyphName](https://developer.apple.com/reference/coregraphics/1396340-cgfontgetglyphwithglyphname?language=objc) |  |
| [CGFontCopyGlyphNameForGlyph](https://developer.apple.com/reference/coregraphics/1396349-cgfontcopyglyphnameforglyph?language=objc) |  |
| [CGFontGetNumberOfGlyphs](https://developer.apple.com/reference/coregraphics/1396371-cgfontgetnumberofglyphs?language=objc) | [CTFontGetGlyphCount](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetGlyphCount) |
| [CGFontGetGlyphAdvances](https://developer.apple.com/reference/coregraphics/1396332-cgfontgetglyphadvances?language=objc) | [CTFontGetAdvancesForGlyphs](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetAdvancesForGlyphs) |
| [CGFontGetUnitsPerEm](https://developer.apple.com/reference/coregraphics/1396344-cgfontgetunitsperem?language=objc) | [CTFontGetUnitsPerEm](https://developer.apple.com/library/ios/documentation/Carbon/Reference/CTFontRef/index.html#//apple_ref/c/func/CTFontGetUnitsPerEm) |

## Performance: EndDraw calls

Consider a call pattern like the following (accurate to develop 1/12/2017):

*void CTFrameDraw(…) {*

*for (…) {*

*CTLineDraw(…);*

*}*

*}*

*void CTLineDraw(…) {*

*for (…) {*

*CTRunDraw(…);*

*}*

*}*

*void CTRunDraw(…) {*

*renderTarget->BeginDraw();*

*…*

*renderTarget->EndDraw();*

*}*

1. Reuse of code is generally good, and the paradigm of ‘implement larger components in terms of smaller ones’ makes sense here. We would like to not change this if possible.
2. The API contract of CTFrameDraw() et al. expects that the object in question is fully drawn after the function returns, so they must behave as if they are in “immediate mode.”
3. Begin/EndDraw() are extremely inefficient if called very many times. Developers are strongly recommended to batch a high number of drawing commands before calling Flush() or EndDraw(), not actually drawing until the last moment possible. In other words, from a performance perspective, the API ought to behave as if it was in “non-immediate mode,”.

A solution that reconciles 2) and 3) would be one where CTRunDraw(), CTLineDraw(), act in “immediate mode” if called by themselves, and “non-immediate mode” if called from CTLineDraw(), CTFrameDraw(), or other tight loop. In other words, they would like to call EndDraw() only after completing the outermost loop in the current call.

A good solution, then, would be to set up a sort of ‘stack’, mapped to the depth of loops we’re currently in. We can push/pop onto the stack before and after each tight loop, and only call BeginDraw() if we’re pushing onto an empty stack, and only call EndDraw() if the stack is empty after popping.

Since our ID2D1RenderTargets are generally owned by a CGContext, CGContext seems like a good place to set up this stack. (On CGD2D, there is one instance of a function with a short-lived render target not owned by a CGContext, but its Begin/EndDraw()s are already at the beginning/end of that function/its lifetime, so no optimization is necessary. If we end up adding other ‘unowned’ render targets that require this optimization, we can move the logic to a different wrapper around a ID2D1RenderTarget, but the prospect currently seems unlikely). An implementation would look something like the following:

*struct \_\_CGContext {*

*uint32\_t \_beginEndDrawDepth;*

*…*

*};*

*void \_CGContextPushBeginDraw(…) {*

*if (\_beginEndDrawDepth++ == 0) {*

*\_renderTarget->BeginDraw();*

*}*

*}*

*void \_CGContextPopEndDraw(…) {*

*if (--\_beginEndDrawDepth == 0) {*

*\_renderTarget->EndDraw();*

*}*

*}*

*void OuterFunctionWithATightLoop(…) {*

*\_CGContextPushBeginDraw(context);*

*for (…) {*

*InnerFunction(…);*

*}*

*\_CGContextPopEndDraw(context);*

*}*

*void InnerFunction(…) {*

*\_CGContextPushBeginDraw(context);*

*…*

*\_CGContextPopEndDraw(context);*

*}*

*(In an actual implementation, we would probably use wil::ScopeExit to call PopEndDraw() to be safe, but for sample code purposes, it’s more illustrative to call directly)*

Under this implementation, OuterFunctionWithATightLoop() only calls EndDraw() once instead of *n* times, but InnerFunction() still acts under “immediate mode” if called on its own. Moreover, this solution is relatively generalizable to other instances of the pattern (anywhere a drawing function is called in a tight loop), and does not require too much churn.

This has an obvious drawback - we put a burden on devs to call these functions around tight loops. However, this type of burden seems fairly standard for the type of performance optimization we’re trying to do (graphics optimization), and probably can’t be lowered much further without causing drawbacks in other areas.

Aside from directly around tight loops, it also makes sense to add these PushBegin/PopEndDraw() calls to UIKit draw functions that have a high probability of venturing into draw commands, such as [UIView drawLayer:] (does not surround a tight loop itself, but does travel into app code that does so). We may wish to tackle these opportunistically to avoid regressions.

*Alternatives Considered*

**Loop unrolling:** We could sacrifice 1) and unroll the bodies of the lower functions into the higher functions, and then call Begin/EndDraw() around the boundaries of those functions. This seems pretty inferior to the recommended solution for the obvious reason that we sacrifice 1).

**Private versions of the Draw functions:** We could have have a CTRunDraw() and CTLineDraw() call a \_CTRunDraw() that does not call Begin/EndDraw(), and only call those in the bodies of the public functions. This seems less generalizable than the recommended solution, and requires some clunkiness with private functions.

*Merging with CGD2D*

**CGContext implementation:** Since the implementation of CGContext is in a fairly different place on CGD2D, we may have to move around the declaration of \_beginEndDrawDepth.

**More EndDraw() calls:** There are EndDraw() calls in a few additional places in CGD2D compared to develop. We should replace them with PushBegin/PopEndDraw() calls, and travel up their stacks and place PushBegin/PopEndDraw() calls around tight loops, just as on develop.

**Needing to update state in the middle:** On CGD2D, there are instances of drawing functions that require an updated state in the middle of the function, as subsequent drawing is based on it (think shadow). Because two BeginDraw()s or two EndDraw()s in a row are invalid and will leave the render target in an error state, we’ll need an extra two functions that force through a Begin/EndDraw() call, checking the current stack depth and calling End/BeginDraw() first if necessary.

# Appendix

Below is the list of the popular Coretext APIs used by applications –

|  |  |  |  |
| --- | --- | --- | --- |
| **Count** | **Api Name** | **Today’s status** | **With DirectWrite** |
| 50 | CTFontCreateWithName |  |  |
| 49 | CTFramesetterCreateWithAttributedString |  |  |
| 48 | kCTFontAttributeName |  |  |
| 48 | CTFramesetterCreateFrame |  |  |
| 46 | CTFramesetterSuggestFrameSizeWithConstraints |  |  |
| 45 | CTFrameGetLines |  |  |
| 43 | CTLineGetTypographicBounds |  |  |
| 43 | CTFrameGetLineOrigins |  |  |
| 42 | CTLineGetStringRange |  |  |
| 42 | CTLineGetGlyphRuns |  |  |
| 41 | CTRunGetTypographicBounds |  |  |
| 41 | CTParagraphStyleCreate |  |  |
| 40 | CTLineGetOffsetForStringIndex |  |  |
| 40 | CTLineCreateWithAttributedString |  |  |
| 39 | CTRunGetStringRange |  |  |
| 39 | CTLineDraw |  |  |
| 38 | CTRunGetAttributes |  |  |
| 38 | CTLineGetStringIndexForPosition |  |  |
| 38 | CTLineCreateTruncatedLine |  |  |
| 35 | CTFontGetSize |  |  |
| 25 | CTFontGetUnderlineThickness |  |  |
| 23 | CTFontGetAscent |  |  |
| 22 | CTFrameDraw |  |  |
| 22 | CTFontGetDescent |  |  |
| 20 | CTLineGetPenOffsetForFlush |  |  |
| 19 | CTFontDescriptorCopyAttribute |  |  |
| 19 | CTFontDescriptorCreateWithAttributes |  |  |
| 19 | CTFontCreateCopyWithAttributes |  |  |
| 18 | CTRunGetGlyphCount |  |  |
| 17 | CTRunGetPositions |  |  |
| 16 | CTRunDelegateCreate |  |  |
| 15 | CTFontManagerRegisterGraphicsFont |  |  |
| 15 | CTRunGetStatus |  |  |
| 15 | CTFontManagerRegisterFontsForURL |  |  |
| 14 | CTFontCopyPostScriptName |  |  |
| 14 | CTFontGetSymbolicTraits |  |  |
| 14 | CTLineGetImageBounds |  |  |
| 14 | CTFontGetBoundingRectsForGlyphs |  |  |
| 13 | CTFontCreateWithFontDescriptor |  |  |
| 13 | CTFontGetAdvancesForGlyphs |  |  |
| 12 | CTRunGetStringIndices |  |  |
| 12 | CTRunGetGlyphs |  |  |
| 12 | CTFontGetLeading |  |  |
| 12 | CTFontGetGlyphsForCharacters |  |  |
| 11 | CTFontCreateCopyWithSymbolicTraits |  |  |
| 11 | CTFontDrawGlyphs |  |  |
| 11 | CTTypesetterCreateWithAttributedString |  |  |
| 11 | CTFontCopyFontDescriptor |  |  |
| 10 | CTRunDelegateGetRefCon |  |  |
| 10 | CTParagraphStyleGetValueForSpecifier |  |  |
| 10 | CTFontCopyName |  |  |
| 9 | CTLineGetGlyphCount |  |  |
| 9 | CTFontDescriptorCreateCopyWithAttributes |  |  |
| 9 | CTFontCreateWithGraphicsFont |  |  |
| 9 | CTFontCopyFamilyName |  |  |
| 8 | CTFontCreateUIFontForLanguage |  |  |
| 8 | CTFontCopyFullName |  |  |
| 8 | CTTypesetterSuggestLineBreak |  |  |
| 7 | CTFontDescriptorCreateMatchingFontDescriptor |  |  |
| 7 | CTRunGetPositionsPtr |  |  |
| 7 | CTTypesetterCreateLine |  |  |
| 7 | CTRunGetAdvances |  |  |
| 7 | CTFontGetCapHeight |  |  |
| 7 | CTFontCreateForString |  |  |
| 7 | CTFontCopyGraphicsFont |  |  |
| 6 | CTLineGetTrailingWhitespaceWidth |  |  |
| 6 | CTFrameGetVisibleStringRange |  |  |
| 6 | CTFontGetXHeight |  |  |
| 5 | CTRunGetTextMatrix |  |  |
| 5 | CTRunGetImageBounds |  |  |
| 5 | CTRunDraw |  |  |
| 5 | CTFontGetUnderlinePosition |  |  |
| 5 | CTLineGetBoundsWithOptions |  |  |
| 5 | CTFontDescriptorCreateWithNameAndSize |  |  |
| 5 | CTFontGetBoundingBox |  |  |
| 5 | CTFontGetTypeID |  |  |
| 5 | CTFontCopyTable |  |  |
| 4 | CTTextTabGetLocation |  |  |
| 4 | CTTextTabCreate |  |  |
| 4 | CTFontGetVerticalTranslationsForGlyphs |  |  |
| 4 | CTFontDescriptorCreateMatchingFontDescriptors |  |  |
| 4 | CTFrameGetPath |  |  |
| 4 | CTFontDescriptorCopyAttributes |  |  |
| 3 | CTTypesetterSuggestLineBreakWithOffset |  |  |
| 3 | CTTypesetterSuggestClusterBreakWithOffset |  |  |
| 3 | CTTypesetterCreateLineWithOffset |  |  |
| 3 | CTGetCoreTextVersion |  |  |
| 3 | CTFontDescriptorMatchFontDescriptorsWithProgressHandler |  |  |
| 3 | CTFontCopyTraits |  |  |
| 3 | CTFontGetStringEncoding |  |  |
| 3 | CTFontDescriptorCopyLocalizedAttribute |  |  |
| 3 | CTRunGetGlyphsPtr |  |  |
| 3 | CTFontCollectionCreateMatchingFontDescriptors |  |  |
| 3 | CTFontCollectionCreateFromAvailableFonts |  |  |
| 3 | CTFontManagerUnregisterFontsForURL |  |  |
| 3 | CTLineCreateJustifiedLine |  |  |
| 3 | CTRunGetStringIndicesPtr |  |  |
| 3 | CTFontCreatePathForGlyph |  |  |
| 2 | CTParagraphStyleGetTypeID |  |  |
| 2 | CTFrameGetStringRange |  |  |
| 2 | CTFontGetUnitsPerEm |  |  |
| 2 | CTFontManagerUnregisterGraphicsFont |  |  |
| 1 | CTTextTabGetOptions |  |  |
| 1 | CTRunDelegateGetTypeID |  |  |
| 1 | CTRubyAnnotationGetTypeID |  |  |
| 1 | CTRubyAnnotationGetTextForPosition |  |  |
| 1 | CTRubyAnnotationGetSizeFactor |  |  |
| 1 | CTRubyAnnotationGetOverhang |  |  |
| 1 | CTRubyAnnotationGetAlignment |  |  |
| 1 | CTRubyAnnotationCreate |  |  |
| 1 | CTTextTabGetAlignment |  |  |
| 1 | CTFontGetMatrix |  |  |
| 1 | CTFontGetGlyphCount |  |  |
| 1 | CTFontCopyLocalizedName |  |  |
| 1 | CTFontCopyCharacterSet |  |  |
| 1 | CTFontCopyAttribute |  |  |
| 1 | CTFontGetSlantAngle |  |  |
| 1 | CTFontCopyAvailableTables |  |  |