Core Graphics Reference Collection

Graphics & Imaging > Quartz

2006-12-11

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

Simultaneously published in the United States and Canada.

Contents

Tables

Introduction

Framework	/System/Library/Frameworks/CoreGraphics
Header file directories	/System/Library/Frameworks/CoreGraphics
Companion guide	Quartz 2D Programming Guide
Declared in	CABase.h CGAffineTransform.h CGBitmapContext.h CGColor.h CGColorSpace.h CGContext.h CGDataConsumer.h CGDataProvider.h CGDirectDisplay.h CGDirectPalette.h CGDisplayConfiguration.h CGDisplayFade.h CGError.h CGEvent.h CGEventSource.h CGEventTypes.h CGFont.h CGFunction.h CGGLContext.h CGGeometry.h CGGradient.h CGImage.h CGImageDestination.h CGImageProperties.h CGImageSource.h CGLayer.h CGPDFArray.h CGPDFContentStream.h CGPDFContext.h CGPDFDictionary.h CGPDFDocument.h CGPDFObject.h CGPDFOperatorTable.h CGPDFPage.h CGPDFScanner.h CGPDFStream.h CGPDFString.h CGPSConverter.h CGPath.h CGPattern.h CGRemoteOperation.h

CGSession.h
CGShading.h
CGWindowLevel.h

This collection of documents provides the API reference for the Core Graphics technologies, which are fundamental to the Mac OS X graphics and windowing environment. Core Graphics technologies include Quartz 2D, the drawing API that implements a superset of the PDF 1.4 specification; Quartz Display Services, the API for configuring and controlling display hardware; and Quartz Events Services, the API for observing and alterating low-level user input events.

Opaque Types

PART I

Opaque Types

CGBitmapContext Reference

Derived From:	`CGContextRef` (page 131)
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGBitmapContext.h
Companion guide	Quartz 2D Programming Guide

Overview

The CGBitmapContext header file defines functions that create and operate on a Quartz bitmap graphics context. A bitmap graphics context is a type of `CGContextRef` (page 131) that you can use for drawing bits to memory. The functions in this reference operate only on Quartz bitmap graphics contexts created using the functionCGBitmapContextCreate (page 18).

The number of components for each pixel in a bitmap graphics context is specified by a color space (defined by a `CGColorSpaceRef` (page 50), which includes RGB, grayscale, and CMYK, and which also may specify a destination color profile). The bitmap graphics context specifies whether the bitmap should contain an alpha channel, and how the bitmap is generated.

Functions by Task

Creating Bitmap Contexts

`CGBitmapContextCreate` (page 18)
Creates a bitmap graphics context.

`CGBitmapContextCreateImage` (page 19)
Creates and returns a Quartz image from the pixel data in a bitmap graphics context.

Getting Information About Bitmap Contexts

These functions return the values of attributes specified when a bitmap context is created.

`CGBitmapContextGetBitmapInfo` (page 20)
Obtains the bitmap information associated with a bitmap graphics context.

`CGBitmapContextGetAlphaInfo` (page 20)
Returns the alpha information associated with the context, which indicates how a bitmap context handles the alpha component.

`CGBitmapContextGetBitsPerComponent` (page 21)
Returns the bits per component of a bitmap context.

`CGBitmapContextGetBitsPerPixel` (page 21)
Returns the bits per pixel of a bitmap context.

`CGBitmapContextGetBytesPerRow` (page 22)
Returns the bytes per row of a bitmap context.

`CGBitmapContextGetColorSpace` (page 22)
Returns the color space of a bitmap context.

`CGBitmapContextGetData` (page 22)
Returns a pointer to the image data associated with a bitmap context.

`CGBitmapContextGetHeight` (page 23)
Returns the height in pixels of a bitmap context.

`CGBitmapContextGetWidth` (page 23)
Returns the width in pixels of a bitmap context.

Functions

CGBitmapContextCreate

Creates a bitmap graphics context.

```
CGContextRef CGBitmapContextCreate (
   void *data,
   size_t width,
   size_t height,
   size_t bitsPerComponent,
   size_t bytesPerRow,
   CGColorSpaceRef colorspace,
   CGBitmapInfo bitmapInfo
);
```

Parameters

data

A pointer to the destination in memory where the drawing is to be rendered. The size of this memory block should be at least `(bytesPerRow*height)` bytes.

Starting in Mac OS X v10.3, you can pass `NULL` if you don't care where the data is stored. This frees you from managing your own memory, which reduces memory leak issues. Quartz has more flexibility when it manages data storage for you. For example, it's possible for Quartz to use OpenGL for rendering if it takes care of the memory.

width

The width, in pixels, of the required bitmap.

height

The height, in pixels, of the required bitmap.

bitsPerComponent

The number of bits to use for each component of a pixel in memory. For example, for a 32-bit pixel format and an RGB color space, you would specify a value of 8 bits per component. For more information about supported pixel formats, see *Quartz 2D Programming Guide*.

bytesPerRow
The number of bytes of memory to use per row of the bitmap.

colorspace
The color space to use for the bitmap context. Note that indexed color spaces are not supported for bitmap graphics contexts.

bitmapInfo
A `CGBitmapInfo` constant that specifies whether the bitmap should contain an alpha channel and its relative location in a pixel, along with whether the components are floating-point or integer values. (See *CGImage Reference* for a description `CGBitmapInfo` constants.) In *Quartz 2D Programming Guide*, see "Creating a Bitmap Graphics Context" (in the *Graphics Contexts* chapter) for the color space, bits per pixel, bits per pixel component, and bitmap information constant combinations that you can use when creating a bitmap context with `CGBitmapContextCreate`.

Return Value
A new bitmap context, or `NULL` if a context could not be created. You are responsible for releasing this object using `CGContextRelease` (page 96).

Discussion
When you call this function, Quartz creates a bitmap drawing environment—that is, a bitmap context—to your specifications. When you draw into this context, Quartz renders your drawing as bitmapped data in the specified block of memory.

The pixel format for a new bitmap context is determined by three parameters—the number of bits per component, the color space, and an alpha option (expressed as a `CGBitmapInfo` (page 221) constant). The alpha value determines the opacity of a pixel when it is drawn.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGBitmapContext.h`

CGBitmapContextCreateImage

Creates and returns a Quartz image from the pixel data in a bitmap graphics context.

```
CGImageRef CGBitmapContextCreateImage (
   CGContextRef c
);
```

Parameters

c
A bitmap graphics context.

Return Value
A CGImage object that contains a snapshot of the bitmap graphics context or `NULL` if the image is not created.

Discussion
The CGImage object returned by this function is created by a copy operation. Subsequent changes to the bitmap graphics context do not affect the contents of the returned image. In some cases the copy operation actually follows copy-on-write semantics, so that the actual physical copy of the bits occur only if the underlying

data in the bitmap graphics context is modified. As a consequence, you may want to use the resulting image and release it before you perform additional drawing into the bitmap graphics context. In this way, you can avoid the actual physical copy of the data.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGBitmapContext.h`

CGBitmapContextGetAlphaInfo

Returns the alpha information associated with the context, which indicates how a bitmap context handles the alpha component.

```
CGImageAlphaInfo CGBitmapContextGetAlphaInfo (
   CGContextRef c
);
```

Parameters

context

A bitmap context.

Return Value
A bitmap information constant. If the specified context is not a bitmap context, `kCGImageAlphaNone` (page 220) is returned. See `CGImageAlphaInfo` (renamed to `CGBitmapInfo` in Mac OS X v10.4) for more information about values.

Discussion
Every bitmap context contains an attribute that specifies whether the bitmap contains an alpha component, and how it is generated. The alpha component determines the opacity of a pixel when it is drawn.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGBitmapContext.h`

CGBitmapContextGetBitmapInfo

Obtains the bitmap information associated with a bitmap graphics context.

```
CGBitmapInfo CGBitmapContextGetBitmapInfo (
   CGContextRef c
);
```

Parameters

c

A bitmap graphics context.

Return Value
The bitmap info of the bitmap graphics context or `0` if `c` is not a bitmap graphics context. See *CGImage Reference* for a description of the `CGBitmapInfo` (page 221) constants that can be returned.

Discussion
The `CGBitmapInfo` data returned by the function specifies whether the bitmap contains an alpha channel and how the alpha channel is generated, along with whether the components are floating-point or integer.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGBitmapContext.h`

CGBitmapContextGetBitsPerComponent

Returns the bits per component of a bitmap context.

```
size_t CGBitmapContextGetBitsPerComponent (
   CGContextRef c
);
```

Parameters

context
The bitmap context to examine.

Return Value
The number of bits per component in the specified context, or `0` if the context is not a bitmap context.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGBitmapContext.h`

CGBitmapContextGetBitsPerPixel

Returns the bits per pixel of a bitmap context.

```
size_t CGBitmapContextGetBitsPerPixel (
   CGContextRef c
);
```

Parameters

context
The bitmap context to examine.

Return Value
The number of bits per pixel in the specified context, or `0` if the context is not a bitmap context.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGBitmapContext.h`

CGBitmapContextGetBytesPerRow

Returns the bytes per row of a bitmap context.

```
size_t CGBitmapContextGetBytesPerRow (
   CGContextRef c
);
```

Parameters

context

The bitmap context to examine.

Return Value

The number of bytes per row of the specified context, or `0` if the context is not a bitmap context.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetColorSpace

Returns the color space of a bitmap context.

```
CGColorSpaceRef CGBitmapContextGetColorSpace (
   CGContextRef c
);
```

Parameters

context

The bitmap context to examine.

Return Value

The color space of the specified context, or `NULL` if the context is not a bitmap context. You are responsible for retaining and releasing this object as necessary.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGBitmapContext.h`

CGBitmapContextGetData

Returns a pointer to the image data associated with a bitmap context.

```
void * CGBitmapContextGetData (
   CGContextRef c
);
```

Parameters

context

The bitmap context to examine.

Return Value
A pointer to the specified bitmap context's image data, or `NULL` if the context is not a bitmap context.

Availability
Available in Mac OS X version 10.2 and later.

Related Sample Code
CarbonSketch

Declared In
`CGBitmapContext.h`

CGBitmapContextGetHeight

Returns the height in pixels of a bitmap context.

```
size_t CGBitmapContextGetHeight (
   CGContextRef c
);
```

Parameters

context
The bitmap context to examine.

Return Value
The height in pixels of the specified context, or `0` if the context is not a bitmap context.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGBitmapContext.h`

CGBitmapContextGetWidth

Returns the width in pixels of a bitmap context.

```
size_t CGBitmapContextGetWidth (
   CGContextRef c
);
```

Parameters

context
The bitmap context to examine.

Return Value
The width in pixels of the specified context, or `0` if the context is not a bitmap context.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGBitmapContext.h`

CGColor Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGColor.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGColorRef` opaque type contains a set of components (such as red, green, and blue) that uniquely define a color, and a color space that specifies how those components should be interpreted. Quartz color objects provide a fast and convenient way to manage and set colors, especially colors that are used repeatedly. Quartz drawing operations use color objects for setting fill and stroke colors, managing alpha, and setting color with a pattern.

See also these related references: *CGContext Reference*, *CGColorSpace Reference*, and *CGPattern Reference*.

Functions by Task

Getting a Constant Color

`CGColorGetConstantColor` (page 32)
Returns a color object that represents a constant color.

Retaining and Releasing Color Objects

`CGColorRelease` (page 34)
Decrements the retain count of a Quartz color.

`CGColorRetain` (page 34)
Increments the retain count of a Quartz color.

Creating Quartz Colors

`CGColorCreate` (page 26)
Creates a Quartz color using a list of intensity values (including alpha) and an associated color space.

`CGColorCreateCopy` (page 27)
Creates a copy of an existing Quartz color.

`CGColorCreateGenericGray` (page 29)
Creates a color in the Generic gray color space.

`CGColorCreateGenericRGB` (page 29)
Creates a color in the Generic RGB color space.

`CGColorCreateGenericCMYK` (page 28)
Creates a color in the Generic CMYK color space.

`CGColorCreateCopyWithAlpha` (page 27)
Creates a copy of an existing Quartz color, substituting a new alpha value.

`CGColorCreateWithPattern` (page 30)
Creates a Quartz color using a list of intensity values (including alpha), a pattern color space, and a pattern.

Getting Information about Quartz Colors

`CGColorEqualToColor` (page 30)
Indicates whether two colors are equal.

`CGColorGetAlpha` (page 31)
Returns the value of the alpha component associated with a Quartz color.

`CGColorGetColorSpace` (page 31)
Returns the color space associated with a Quartz color.

`CGColorGetComponents` (page 32)
Returns the values of the color components (including alpha) associated with a Quartz color.

`CGColorGetNumberOfComponents` (page 32)
Returns the number of color components (including alpha) associated with a Quartz color.

`CGColorGetPattern` (page 33)
Returns the pattern associated with a Quartz color in a pattern color space.

`CGColorGetTypeID` (page 33)
Returns the Core Foundation type identifier for a Quartz color data type.

Functions

CGColorCreate

Creates a Quartz color using a list of intensity values (including alpha) and an associated color space.

```
CGColorRef CGColorCreate (
   CGColorSpaceRef colorspace,
   const CGFloat components[]
);
```

Parameters

colorspace

A color space for the new color. Quartz retains this object; upon return, you may safely release it.

components

An array of intensity values describing the color. The array should contain *n*+1 values that correspond to the *n* color components in the specified color space, followed by the alpha component. Each component value should be in the range appropriate for the color space. Values outside this range will be clamped to the nearest correct value.

Return Value

A new Quartz color. You are responsible for releasing this object using `CGColorRelease` (page 34).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGColor.h`

CGColorCreateCopy

Creates a copy of an existing Quartz color.

```
CGColorRef CGColorCreateCopy (
   CGColorRef color
);
```

Parameters

color

A Quartz color.

Return Value

A copy of the specified color. You are responsible for releasing this object using `CGColorRelease` (page 34).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGColor.h`

CGColorCreateCopyWithAlpha

Creates a copy of an existing Quartz color, substituting a new alpha value.

```
CGColorRef CGColorCreateCopyWithAlpha (
   CGColorRef color,
   CGFloat alpha
);
```

Parameters

color

The Quartz color to copy.

alpha

A value that specifies the desired opacity of the copy. Values outside the range `[0,1]` are clamped to `0` or `1`.

Return Value
A copy of the specified color, using the specified alpha value. You are responsible for releasing this object using `CGColorRelease` (page 34).

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGColor.h`

CGColorCreateGenericCMYK

Creates a color in the Generic CMYK color space.

```
CGColorRef CGColorCreateGenericCMYK(
    CGFloat cyan,
    CGFloat magenta,
    CGFloat yellow,
    CGFloat black,
    CGFloat alpha
);
```

Parameters

cyan

A cyan value (`0.0` - `1.0`).

magenta

A magenta value (`0.0` - `1.0`).

yellow

A yellow value (`0.0` - `1.0`).

black

A black value (`0.0` - `1.0`).

alpha

An alpha value (0.0 - 1.0).

Return Value
A color object.

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

Declared In
CGColor.h

CGColorCreateGenericGray

Creates a color in the Generic gray color space.

```
CGColorRef CGColorCreateGenericGray(
    CGFloat gray,
    CGFloat alpha
);
```

Parameters

gray
A grayscale value (0.0 - 1.0).

alpha
An alpha value (0.0 - 1.0).

Return Value
A color object.

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

Declared In
CGColor.h

CGColorCreateGenericRGB

Creates a color in the Generic RGB color space.

```
CGColorRef CGColorCreateGenericRGB(
    CGFloat red,
    CGFloat green,
    CGFloat blue,
    CGFloat alpha
);
```

Parameters

red
A red component value (0.0 - 1.0).

green
A green component value (0.0 - 1.0).

blue
A blue component value (0.0 - 1.0).

alpha
An alpha value (0.0 - 1.0).

Return Value
A color object.

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

Related Sample Code
CALayerEssentials

Declared In
CGColor.h

CGColorCreateWithPattern

Creates a Quartz color using a list of intensity values (including alpha), a pattern color space, and a pattern.

```
CGColorRef CGColorCreateWithPattern (
   CGColorSpaceRef colorspace,
   CGPatternRef pattern,
   const CGFloat components[]
);
```

Parameters

colorspace
A pattern color space for the new color. Quartz retains the color space you pass in. On return, you may safely release it.

pattern
A pattern for the new color object. Quartz retains the pattern you pass in. On return, you may safely release it.

components
An array of intensity values describing the color. The array should contain `n + 1` values that correspond to the `n` color components in the specified color space, followed by the alpha component. Each component value should be in the range appropriate for the color space. Values outside this range will be clamped to the nearest correct value.

Return Value
A new Quartz color. You are responsible for releasing this object using `CGColorRelease` (page 34).

Availability
Available in Mac OS X version 10.3 and later.

Declared In
CGColor.h

CGColorEqualToColor

Indicates whether two colors are equal.

```
bool CGColorEqualToColor (
   CGColorRef color1,
   CGColorRef color2
);
```

Parameters

color1
The first Quartz color to compare.

color2
The second Quartz color to compare.

Return Value
A Boolean value that, if `true`, indicates that the specified colors are equal. If the colors are not equal, the value is `false`.

Discussion
Two colors are equal if they have equal color spaces and numerically equal color components.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGColor.h`

CGColorGetAlpha

Returns the value of the alpha component associated with a Quartz color.

```
CGFloat CGColorGetAlpha (
   CGColorRef color
);
```

Parameters

color
A Quartz color.

Return Value
An alpha intensity value in the range `[0,1]`. The value represents the opacity of the color.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGColor.h`

CGColorGetColorSpace

Returns the color space associated with a Quartz color.

```
CGColorSpaceRef CGColorGetColorSpace (
   CGColorRef color
);
```

Parameters

color
A Quartz color.

Return Value
The Quartz color space for the specified color. You are responsible for retaining and releasing it as needed.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGColor.h`

CGColorGetComponents

Returns the values of the color components (including alpha) associated with a Quartz color.

```
const CGFloat * CGColorGetComponents (
   CGColorRef color
);
```

Parameters

color

A Quartz color.

Return Value

An array of intensity values for the color components (including alpha) associated with the specified color. The size of the array is one more than the number of components of the color space for the color.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

CGColor.h

CGColorGetConstantColor

Returns a color object that represents a constant color.

```
CGColorRef CGColorGetConstantColor(
    CFStringRef colorName
);
```

Parameters

colorName

A color name. You can pass any of the "Constant Colors" (page 35) constant.

Return Value

A color object.

Discussion

As `CGColorGetConstantColor` is not a "Copy" or "Create" function, it does not necessarily return a new reference each time it's called. As a consequence, you should not release the returned value. However, colors returned from `CGColorGetConstantColor` can be retained and released in a properly nested fashion, just as any other Core Foundation type can.

Availability

Available in Mac OS X v10.5 and later.

Declared In

CGColor.h

CGColorGetNumberOfComponents

Returns the number of color components (including alpha) associated with a Quartz color.

```
size_t CGColorGetNumberOfComponents (
   CGColorRef color
);
```

Parameters

color

A Quartz color.

Return Value
The number of color components (including alpha) associated with the specified color. This number is one more than the number of components of the color space for the color.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGColor.h`

CGColorGetPattern

Returns the pattern associated with a Quartz color in a pattern color space.

```
CGPatternRef CGColorGetPattern (
   CGColorRef color
);
```

Parameters

color

A Quartz color.

Return Value
The pattern for the specified color. You are responsible for retaining and releasing the pattern as needed.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGColor.h`

CGColorGetTypeID

Returns the Core Foundation type identifier for a Quartz color data type.

```
CFTypeID CGColorGetTypeID (
   void
);
```

Return Value
The Core Foundation type identifier for `CGColorRef`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGColor.h`

CGColorRelease

Decrements the retain count of a Quartz color.

```
void CGColorRelease (
    CGColorRef color
);
```

Parameters

color

The Quartz color to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `color` parameter is `NULL`.

Availability

Available in Mac OS X version 10.3 and later.

Related Sample Code

CALayerEssentials

Declared In

`CGColor.h`

CGColorRetain

Increments the retain count of a Quartz color.

```
CGColorRef CGColorRetain (
    CGColorRef color
);
```

Parameters

color

The Quartz color to retain.

Return Value

The same color you passed in as the *color* parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `color` parameter is `NULL`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGColor.h`

Data Types

CGColorRef

An opaque type that represents a color used in Quartz 2D drawing.

```
typedef struct CGColor *CGColorRef;
```

Discussion
`CGColorRef` is the fundamental data type used internally by Quartz to represent colors. CGColor objects. and the functions that operate on them, provide a fast and convenient way of managing and setting colors directly, especially colors that are reused (such as black for text).

In Mac OS X version 10.3 and later, `CGColorRef` is derived from `CFTypeRef` and inherits the properties that all Core Foundation types have in common. For more information, see CFType Reference.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGColor.h`

Constants

Constant Colors

Commonly used colors.

```
const CFStringRef kCGColorWhite;
const CFStringRef kCGColorBlack;
const CFStringRef kCGColorClear;
```

Constants
`kCGColorWhite`
The white color in the Generic gray color space.
Available in Mac OS X v10.5 and later.
Declared in `CGColor.h`.

`kCGColorBlack`
The black color in the Generic gray color space.
Available in Mac OS X v10.5 and later.
Declared in `CGColor.h`.

`kCGColorClear`
The clear color in the Generic gray color space.
Available in Mac OS X v10.5 and later.
Declared in `CGColor.h`.

Declared In
`CGColor.h`

CGColorSpace Reference

Derived From:	*CFType Reference*
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGColorSpace.h
Companion guides	Quartz 2D Programming Guide CGColor Reference CGContext Reference

Overview

The `CGColorSpaceRef` opaque type encapsulates color space information that is used to specify how Quartz interprets color information. A color space specifies how color values are interpreted. A color space is multi-dimensional, and each dimension represents a specific color component. For example, the colors in an RGB color space have three dimensions or components—red, green, and blue. The intensity of each component is represented by floating point values—their range and meaning depends on the color space in question.

Different types of devices (scanners, monitors, printers) operate within different color spaces (RGB, CMYK, grayscale). Additionally, two devices of the same type (for example, color displays from different manufacturers) may operate within the same kind of color space, yet still produce a different range of colors, or gamut. Color spaces that are correctly specified ensure that an image has a consistent appearance regardless of the output device.

Quartz supports several kinds of color spaces:

- Calibrated color spaces ensure that colors appear the same when displayed on different devices. The visual appearance of the color is preserved, as far as the capabilities of the device allow.
- Device-dependent color spaces are tied to the system of color representation of a particular device. Device color spaces are not recommended when high-fidelity color preservation is important.
- Special color spaces—indexed and pattern. An indexed color space contains a color table with up to 256 entries and a base color space to which the color table entries are mapped. Each entry in the color table specifies one color in the base color space. A pattern color space is used when stroking or filling with a pattern. Pattern color spaces are supported in Mac OS X version 10.1 and later.

Functions by Task

Creating Device-Independent Color Spaces

`CGColorSpaceCreateCalibratedGray` (page 39)
Creates a calibrated grayscale color space.

`CGColorSpaceCreateCalibratedRGB` (page 40)
Creates a calibrated RGB color space.

`CGColorSpaceCreateICCBased` (page 43)
Creates a device-independent color space that is defined according to the ICC color profile specification.

`CGColorSpaceCreateLab` (page 44)
Creates a device-independent color space that is relative to human color perception, according to the CIE L*a*b* standard.

Creating Generic or Device-Dependent Color Spaces

In Mac OS X v10.4 and later, the color space returned by each of these functions is no longer device-dependent and is replaced by a generic counterpart.

`CGColorSpaceCreateDeviceCMYK` (page 41)
Creates a device-dependent CMYK color space.

`CGColorSpaceCreateDeviceGray` (page 42)
Creates a device-dependent grayscale color space.

`CGColorSpaceCreateDeviceRGB` (page 42)
Creates a device-dependent RGB color space.

`CGColorSpaceCreateWithPlatformColorSpace` (page 46)
Creates a platform-specific color space.

Creating Special Color Spaces

`CGColorSpaceCreateIndexed` (page 44)
Creates an indexed color space, consisting of colors specified by a color lookup table.

`CGColorSpaceCreatePattern` (page 45)
Creates a pattern color space.

`CGColorSpaceCreateWithName` (page 46)
Creates a specified type of Quartz color space.

Getting Information About Color Spaces

`CGColorSpaceCopyICCProfile` (page 39)
Returns a copy of the ICC profile of the provided color space.

`CGColorSpaceGetNumberOfComponents` (page 48)
Returns the number of color components in a color space.

`CGColorSpaceGetTypeID` (page 49)
Returns the Core Foundation type identifier for Quartz color spaces.

`CGColorSpaceGetModel` (page 48)
Returns the color space model of the provided color space.

`CGColorSpaceGetBaseColorSpace` (page 47)
Returns the base color space of a pattern or indexed color space.

`CGColorSpaceGetColorTableCount` (page 48)
Returns the number of entries in the color table of an indexed color space.

`CGColorSpaceGetColorTable` (page 47)
Copies the entries in the color table of an indexed color space.

Retaining and Releasing Color Spaces

`CGColorSpaceRelease` (page 49)
Decrements the retain count of a color space.

`CGColorSpaceRetain` (page 50)
Increments the retain count of a color space.

Functions

CGColorSpaceCopyICCProfile

Returns a copy of the ICC profile of the provided color space.

```
CFDataRef CGColorSpaceCopyICCProfile(
    CGColorSpaceRef space
);
```

Parameters

space
The color space whose ICC profile you want to obtain.

Return Value
The ICC profile or `NULL` if the color space does not have an ICC profile.

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

Declared In
`CGColorSpace.h`

CGColorSpaceCreateCalibratedGray

Creates a calibrated grayscale color space.

```
CGColorSpaceRef CGColorSpaceCreateCalibratedGray (
   const CGFloat whitePoint[3],
   const CGFloat blackPoint[3],
   CGFloat gamma
);
```

Parameters

whitePoint

An array of 3 numbers specifying the tristimulus value, in the CIE 1931 XYZ-space, of the diffuse white point.

blackPoint

An array of 3 numbers specifying the tristimulus value, in CIE 1931 XYZ-space, of the diffuse black point.

gamma

The gamma value appropriate to the imaging device.

Return Value

A new calibrated gray color space. You are responsible for releasing this object by calling `CGColorSpaceRelease` (page 49). If unsuccessful, returns `NULL`.

Discussion

Creates a device-independent grayscale color space that represents colors relative to a reference white point. This white point is based on the whitest light that can be generated by the output device. Colors in a device-independent color space should appear the same when displayed on different devices, to the extent that the capabilities of the device allow.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceCreateCalibratedRGB

Creates a calibrated RGB color space.

```
CGColorSpaceRef CGColorSpaceCreateCalibratedRGB (
   const CGFloat whitePoint[3],
   const CGFloat blackPoint[3],
   const CGFloat gamma[3],
   const CGFloat matrix[9]
);
```

Parameters

whitePoint

An array of 3 numbers specifying the tristimulus value, in the CIE 1931 XYZ-space, of the diffuse white point.

blackPoint

An array of 3 numbers specifying the tristimulus value, in CIE 1931 XYZ-space, of the diffuse black point.

gamma

An array of 3 numbers specifying the gamma for the red, green, and blue components of the color space.

matrix

An array of 9 numbers specifying the linear interpretation of the gamma-modified RGB values of the color space with respect to the final XYZ representation.

Return Value
A new calibrated RGB color space. You are responsible for releasing this object by calling `CGColorSpaceRelease` (page 49). If unsuccessful, returns `NULL`.

Discussion
Creates a device-independent RGB color space that represents colors relative to a reference white point. This white point is based on the whitest light that can be generated by the output device. Colors in a device-independent color space should appear the same when displayed on different devices, to the extent that the capabilities of the device allow.

For color spaces that require a detailed gamma, such as the piecewise transfer function used in sRGB or ITU-R BT.709, you may want to use the function `CGColorSpaceCreateICCBased` (page 43) instead, because it can accurately represent these gamma curves.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGColorSpace.h`

CGColorSpaceCreateDeviceCMYK

Creates a device-dependent CMYK color space.

```
CGColorSpaceRef CGColorSpaceCreateDeviceCMYK (
   void
);
```

Return Value
A device-dependent CMYK color space. You are responsible for releasing this object by calling `CGColorSpaceRelease` (page 49). If unsuccessful, returns `NULL`.

Discussion
In Mac OS X v10.4 and later, this color space is no longer device-dependent and is replaced by the generic counterpart—`kCGColorSpaceGenericCMYK`—described in "`Color Space Names`" (page 51). If you use this function in Mac OS X v10.4 and later, colors are mapped to the generic color spaces. If you want to bypass color matching, use the color space of the destination context.

Colors in a device-dependent color space are not transformed or otherwise modified when displayed on an output device—that is, there is no attempt to maintain the visual appearance of a color. As a consequence, colors in a device color space often appear different when displayed on different output devices. For this reason, device color spaces are not recommended when color preservation is important.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGColorSpace.h`

CGColorSpaceCreateDeviceGray

Creates a device-dependent grayscale color space.

```
CGColorSpaceRef CGColorSpaceCreateDeviceGray (
   void
);
```

Return Value
A device-dependent gray color space. You are responsible for releasing this object by calling `CGColorSpaceRelease` (page 49). If unsuccessful, returns `NULL`.

Discussion
In Mac OS X v10.4 and later, this color space is no longer device-dependent and is replaced by the generic counterpart—`kCGColorSpaceGenericGray`—described in "Color Space Names" (page 51). If you use this function in Mac OS X v10.4 and later, colors are mapped to the generic color spaces. If you want to bypass color matching, use the color space of the destination context.

Colors in a device-dependent color space are not transformed or otherwise modified when displayed on an output device—that is, there is no attempt to maintain the visual appearance of a color. As a consequence, colors in a device color space often appear different when displayed on different output devices. For this reason, device color spaces are not recommended when color preservation is important.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGColorSpace.h`

CGColorSpaceCreateDeviceRGB

Creates a device-dependent RGB color space.

```
CGColorSpaceRef CGColorSpaceCreateDeviceRGB (
   void
);
```

Return Value
A device-dependent RGB color space. You are responsible for releasing this object by calling `CGColorSpaceRelease` (page 49). If unsuccessful, returns `NULL`.

Discussion
In Mac OS X v10.4 and later, this color space is no longer device-dependent and is replaced by the generic counterpart—`kCGColorSpaceGenericRGB`—described in "Color Space Names" (page 51). If you use this function in Mac OS X v10.4 and later, colors are mapped to the generic color spaces. If you want to bypass color matching, use the color space of the destination context.

Colors in a device-dependent color space are not transformed or otherwise modified when displayed on an output device—that is, there is no attempt to maintain the visual appearance of a color. As a consequence, colors in a device color space often appear different when displayed on different output devices. For this reason, device color spaces are not recommended when color preservation is important.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGColorSpace.h

CGColorSpaceCreateICCBased

Creates a device-independent color space that is defined according to the ICC color profile specification.

```
CGColorSpaceRef CGColorSpaceCreateICCBased (
   size_t nComponents,
   const CGFloat *range,
   CGDataProviderRef profile,
   CGColorSpaceRef alternate
);
```

Parameters

nComponents
The number of color components in the color space defined by the ICC profile data. This must match the number of components actually in the ICC profile and must equal 1, 3, or 4.

range
An array of numbers that specify the minimum and maximum valid values of the corresponding color components. The size of the array is two times the number of components. If `c[k]` is the `k`th color component, the valid range is range`[2*k]` ≤ `c[k]` ≤ range`[2*k+1]`.

profile
A data provider that supplies the ICC profile.

alternateSpace
An alternate color space to use in case the ICC profile is not supported. The alternate color space must have `nComponents` color components. You must supply an alternate color space. If this parameter is `NULL`, then the function returns `NULL`.

Return Value
A new ICC-based color space object. You are responsible for releasing this object by calling `CGColorSpaceRelease` (page 49). If unsuccessful, returns `NULL`.

Discussion
This function creates an ICC-based color space from an ICC color profile, as defined by the International Color Consortium. ICC profiles define the reproducible color gamut (the range of colors supported by a device) and other characteristics of a particular output device, providing a way to accurately transform the color space of one device to the color space of another. The ICC profile is usually provided by the manufacturer of the device. Additionally, some color monitors and printers contain electronically embedded ICC profile information, as do some bitmap formats such as TIFF. Colors in a device-independent color space should appear the same when displayed on different devices, to the extent that the capabilities of the device allow.

You may want to use this function for a color space that requires a detailed gamma, such as the piecewise transfer function used in sRGB or ITU-R BT.709, because this function can accurately represent these gamma curves.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGColorSpace.h

CGColorSpaceCreateIndexed

Creates an indexed color space, consisting of colors specified by a color lookup table.

```
CGColorSpaceRef CGColorSpaceCreateIndexed (
   CGColorSpaceRef baseSpace,
   size_t lastIndex,
   const unsigned char *colorTable
);
```

Parameters

baseSpace

The color space on which the color table is based.

lastIndex

The maximum valid index value for the color table. The value must be less than or equal to 255.

colorTable

An array of `m*(lastIndex+1)` bytes, where `m` is the number of color components in the base color space. Each byte is an unsigned integer in the range `0` to `255` that is scaled to the range of the corresponding color component in the base color space.

Return Value

A new indexed color space object. You are responsible for releasing this object by calling `CGColorSpaceRelease` (page 49). If unsuccessful, returns `NULL`.

Discussion

An indexed color space contains a color table with up to 255 entries, and a base color space to which the color table entries are mapped. Each entry in the color table specifies one color in the base color space. A value in an indexed color space is treated as an index into the color table of the color space. The data in the table is in meshed format. (For example, for an RGB color space RGB, RGB, RGB, and so on.)

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceCreateLab

Creates a device-independent color space that is relative to human color perception, according to the CIE L*a*b* standard.

```
CGColorSpaceRef CGColorSpaceCreateLab (
   const CGFloat whitePoint[3],
   const CGFloat blackPoint[3],
   const CGFloat range[4]
);
```

Parameters

whitePoint

An array of 3 numbers that specify the tristimulus value, in the CIE 1931 XYZ-space, of the diffuse white point.

blackPoint

An array of 3 numbers that specify the tristimulus value, in CIE 1931 XYZ-space, of the diffuse black point.

range

An array of 4 numbers that specify the range of valid values for the a* and b* components of the color space. The a* component represents values running from green to red, and the b* component represents values running from blue to yellow.

Return Value
A new L*a*b* color space. You are responsible for releasing this object by calling `CGColorSpaceRelease` (page 49). If unsuccessful, returns `NULL`.

Discussion
The CIE L*a*b* space is a nonlinear transformation of the Munsell color notation system (a system which specifies colors by hue, value, and saturation—or "chroma"—values), designed to match perceived color difference with quantitative distance in color space. The L* component represents the lightness value, the a* component represents values running from green to red, and the b* component represents values running from blue to yellow. This roughly corresponds to the way the human brain is thought to decode colors. Colors in a device-independent color space should appear the same when displayed on different devices, to the extent that the capabilities of the device allow.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGColorSpace.h`

CGColorSpaceCreatePattern

Creates a pattern color space.

```
CGColorSpaceRef CGColorSpaceCreatePattern (
   CGColorSpaceRef baseSpace
);
```

Parameters

baseSpace

For masking patterns, the underlying color space that specifies the colors to be painted through the mask. For colored patterns, you should pass `NULL`.

Return Value
A new pattern color space. You are responsible for releasing this object by calling `CGColorSpaceRelease` (page 49). If unsuccessful, returns `NULL`.

Discussion
For information on creating and using patterns, see *Quartz 2D Programming Guide* and *CGPattern Reference*. Quartz retains the color space you pass in. Upon return, you may safely release it by calling `CGColorSpaceRelease` (page 49).

Availability
Available in Mac OS X v10.1 and later.

Declared In
`CGColorSpace.h`

CGColorSpaceCreateWithName

Creates a specified type of Quartz color space.

```
CGColorSpaceRef CGColorSpaceCreateWithName (
   CFStringRef name
);
```

Parameters

name

A color space name. See "Color Space Names" (page 51) for a list of the valid Quartz-defined names.

Return Value

A new generic color space. You are responsible for releasing this object by calling CGColorSpaceRelease (page 49). If unsuccessful, returns NULL.

Discussion

You can use this function to create a generic color space. For more information, see "Color Space Names" (page 51).

Prior to Mac OS X v10.4, you could pass this function one of the constants defined in "Named Color Spaces (Deprecated)" (page 54). As of Mac OS X v10.4, this function returns a generic color space even if you pass is one of the deprecated named color spaces.

Availability

Available in Mac OS X v10.2 and later.

Declared In

CGColorSpace.h

CGColorSpaceCreateWithPlatformColorSpace

Creates a platform-specific color space.

```
CGColorSpaceRef CGColorSpaceCreateWithPlatformColorSpace (
   void *platformColorSpaceReference
);
```

Parameters

platformColorSpace

A generic pointer to a platform-specific color space. In Mac OS X, pass a CMProfileRef—a ColorSync profile. Quartz uses this pointer (and the underlying information) only during the function call.

Return Value

A new color space. You are responsible for releasing this object by calling CGColorSpaceRelease (page 49). If unsuccessful, returns NULL.

Discussion

Colors in a device-dependent color space are not transformed or otherwise modified when displayed on an output device—that is, there is no attempt to maintain the visual appearance of a color. As a consequence, colors in a device color space often appear different when displayed on different output devices. For this reason, device color spaces are not recommended when color preservation is important.

Availability

Available in Mac OS X v10.1 and later.

Related Sample Code
CarbonSketch

Declared In
`CGColorSpace.h`

CGColorSpaceGetBaseColorSpace

Returns the base color space of a pattern or indexed color space.

```
CGColorSpace CGColorSpaceGetBaseColorSpace(
    CGColorSpaceRef space
);
```

Parameters

space

A color space object for a pattern or indexed color space.

Return Value
The base color space if the `space` parameter is a pattern or indexed color space; otherwise, `NULL`.

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

Declared In
`CGColorSpace.h`

CGColorSpaceGetColorTable

Copies the entries in the color table of an indexed color space.

```
void CGColorSpaceGetColorTable(
        CGColorSpaceRef space,
        unsigned char *table);
);
```

Parameters

space

A color space object for an indexed color space.

table

The array pointed to by table should be at least as large as the number of entries in the color table. On output, the array contains the table data in the same format as that passed to `CGColorSpaceCreateIndexed` (page 44).

Discussion
This function does nothing if the color space is not an indexed color space. To determine whether a color space is an indexed color space, call the function `CGColorSpaceGetModel` (page 48).

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

See Also
`CGColorSpaceGetColorTableCount` (page 48)

Declared In
CGColorSpace.h

CGColorSpaceGetColorTableCount

Returns the number of entries in the color table of an indexed color space.

```
size_t CGColorSpaceGetColorTableCount(
    CGColorSpaceRef space
);
```

Parameters

space
A color space object for an indexed color space.

Return Value
The number of entries in the color table of the `space` parameter if the color space is an indexed color space; otherwise, returns `0`.

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

See Also
`CGColorSpaceGetColorTable` (page 47)

Declared In
CGColorSpace.h

CGColorSpaceGetModel

Returns the color space model of the provided color space.

```
CGColorSpaceModel CGColorSpaceGetModel(
    CGColorSpaceRef space
);
```

Parameters

space
A color space object.

Return Value
One of the constants described in "Color Space Models" (page 51).

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

Declared In
CGColorSpace.h

CGColorSpaceGetNumberOfComponents

Returns the number of color components in a color space.

```
size_t CGColorSpaceGetNumberOfComponents (
   CGColorSpaceRef cs
);
```

Parameters

cs

The Quartz color space to examine.

Return Value

The number of color components in the specified color space, not including the alpha value. For example, for an RGB color space, `CGColorSpaceGetNumberOfComponents` returns a value of `3`.

Discussion

A color space defines an n-dimensional space whose dimensions (or components) represent intensity values. For example, you specify colors in RGB space as three intensity values: red, green, and blue. You can use the `CGColorSpaceGetNumberOfComponents` function to obtain the number of components in a given color space.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceGetTypeID

Returns the Core Foundation type identifier for Quartz color spaces.

```
CFTypeID CGColorSpaceGetTypeID (
   void
);
```

Return Value

The identifier for the opaque type `CGColorSpaceRef` (page 50).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGColorSpace.h`

CGColorSpaceRelease

Decrements the retain count of a color space.

```
void CGColorSpaceRelease (
   CGColorSpaceRef cs
);
```

Parameters

cs

The Quartz color space to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `cs` parameter is `NULL`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGColorSpace.h`

CGColorSpaceRetain

Increments the retain count of a color space.

```
CGColorSpaceRef CGColorSpaceRetain (
   CGColorSpaceRef cs
);
```

Parameters

cs

The Quartz color space to retain.

Return Value
The same color space you passed in as the *cs* parameter.

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `cs` parameter is `NULL`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGColorSpace.h`

Data Types

CGColorSpaceRef

An opaque type that encapsulates color space information.

```
typedef struct CGColorSpace *CGColorSpaceRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGColorSpace.h`

Constants

Color Space Names

Convenience constants for commonly used color spaces.

```
CFStringRef kCGColorSpaceGenericGray
CFStringRef kCGColorSpaceGenericRGB
CFStringRef kCGColorSpaceGenericCMYK
CFStringRef kCGColorSpaceGenericRGBLinear
CFStringRef kCGColorSpaceAdobeRGB1998
CFStringRef kCGColorSpaceSRGB
```

Constants

`kCGColorSpaceGenericGray`
The name of the generic gray color space.

`kCGColorSpaceGenericRGB`
The name of the generic RGB color space.

`kCGColorSpaceGenericCMYK`
The name of the generic CMYK color space.

`kCGColorSpaceGenericRGBLinear`
The name of the generic linear RGB color space. This is the same as `kCGColorSpaceGenericRGB` (page 51), but with a gamma equal to `1.0`.

`kCGColorSpaceAdobeRGB1998`
The name of the Adobe RGB (1998) color space. For more information, see "Adobe RGB (1998) Color Image Encoding", Version 2005-05, Adobe Systems Inc. (http://www.adobe.com).

`kCGColorSpaceSRGB`
The name of the SRGB color space.

Discussion

A color space name constant can be passed as a parameter to the function `CGColorSpaceCreateWithName` (page 46). These color spaces replace `"Named Color Spaces (Deprecated)"` (page 54), which are deprecated in Mac OS X v10.4.

Declared In

`CGColorSpace.h`

Color Space Models

Models for color spaces.

```
enum CGColorSpaceModel {
    kCGColorSpaceModelUnknown = -1,
    kCGColorSpaceModelMonochrome,
    kCGColorSpaceModelRGB,
    kCGColorSpaceModelCMYK,
    kCGColorSpaceModelLab,
    kCGColorSpaceModelDeviceN,
    kCGColorSpaceModelIndexed,
    kCGColorSpaceModelPattern
};
typedef int32_t CGColorSpaceModel;
```

Constants

`kCGColorSpaceModelUnknown`
An unknown color space model.
Available in Mac OS X v10.5 and later.
Declared in `CGColorSpace.h`.

`kCGColorSpaceModelMonochrome`
A monochrome color space model.
Available in Mac OS X v10.5 and later.
Declared in `CGColorSpace.h`.

`kCGColorSpaceModelRGB`
An RGB color space model.
Available in Mac OS X v10.5 and later.
Declared in `CGColorSpace.h`.

`kCGColorSpaceModelCMYK`
A CMYK color space model.
Available in Mac OS X v10.5 and later.
Declared in `CGColorSpace.h`.

`kCGColorSpaceModelLab`
A Lab color space model.
Available in Mac OS X v10.5 and later.
Declared in `CGColorSpace.h`.

`kCGColorSpaceModelDeviceN`
A DeviceN color space model.
Available in Mac OS X v10.5 and later.
Declared in `CGColorSpace.h`.

`kCGColorSpaceModelIndexed`
An indexed color space model.
Available in Mac OS X v10.5 and later.
Declared in `CGColorSpace.h`.

`kCGColorSpaceModelPattern`
A pattern color space model.
Available in Mac OS X v10.5 and later.
Declared in `CGColorSpace.h`.

Declared In
`CGColorSpace.h`

Color Rendering Intents

Handling options for colors that are not located within the destination color space of a graphics context.

```
enum CGColorRenderingIntent {
    kCGRenderingIntentDefault,
    kCGRenderingIntentAbsoluteColorimetric,
    kCGRenderingIntentRelativeColorimetric,
    kCGRenderingIntentPerceptual,
    kCGRenderingIntentSaturation
};
typedef enum CGColorRenderingIntent CGColorRenderingIntent;
```

Constants

`kCGRenderingIntentDefault`
The default rendering intent for the graphics context.

Available in Mac OS X v10.0 and later.

Declared in `CGColorSpace.h`.

`kCGRenderingIntentAbsoluteColorimetric`
Map colors outside of the gamut of the output device to the closest possible match inside the gamut of the output device. This can produce a clipping effect, where two different color values in the gamut of the graphics context are mapped to the same color value in the output device's gamut. Unlike the relative colorimetric, absolute colorimetric does not modify colors inside the gamut of the output device.

Available in Mac OS X v10.0 and later.

Declared in `CGColorSpace.h`.

`kCGRenderingIntentRelativeColorimetric`
Map colors outside of the gamut of the output device to the closest possible match inside the gamut of the output device. This can produce a clipping effect, where two different color values in the gamut of the graphics context are mapped to the same color value in the output device's gamut. The relative colorimetric shifts all colors (including those within the gamut) to account for the difference between the white point of the graphics context and the white point of the output device.

Available in Mac OS X v10.0 and later.

Declared in `CGColorSpace.h`.

`kCGRenderingIntentPerceptual`
Preserve the visual relationship between colors by compressing the gamut of the graphics context to fit inside the gamut of the output device. Perceptual intent is good for photographs and other complex, detailed images.

Available in Mac OS X v10.0 and later.

Declared in `CGColorSpace.h`.

`kCGRenderingIntentSaturation`
Preserve the relative saturation value of the colors when converting into the gamut of the output device. The result is an image with bright, saturated colors. Saturation intent is good for reproducing images with low detail, such as presentation charts and graphs.

Available in Mac OS X v10.0 and later.

Declared in `CGColorSpace.h`.

Discussion
The rendering intent specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context. It determines the exact method used to map colors from one color space to another. If you do not explicitly set the rendering intent by calling the function `CGContextSetRenderingIntent` (page 114), the graphics context uses the relative colorimetric rendering intent, except when drawing sampled images.

Declared In
`CGColorSpace.h`

Named Color Spaces (Deprecated)

Color spaces used in the Preferences application.

```
#define kCGColorSpaceUserCMYK CFSTR("kCGColorSpaceUserCMYK")
#define kCGColorSpaceUserGray CFSTR("kCGColorSpaceUserGray")
#define kCGColorSpaceUserRGB CFSTR("kCGColorSpaceUserRGB")
```

Constants

`kCGColorSpaceUserCMYK`
A user-defined CMYK color space.

`kCGColorSpaceUserGray`
A user-defined gray color space.

`kCGColorSpaceUserRGB`
A user-defined RGB color space.

Discussion
These constants are deprecated in Mac OS X v10.4. Instead use "`Color Space Names`" (page 51).

The named color spaces are user-configurable in the "Default Profiles for Documents" pane, located in Mac OS 10.2 in the ColorSync preference panel, and in Mac OS 10.3 in the Displays Color Preference panel. See also `CGColorSpaceCreateWithName` (page 46).

Availability
Available in Mac OS X v10.2 and later but deprecated in Mac OS X v10.4.

Declared In
`CGColorSpace.h`

CGContext Reference

Derived From:	*CFType Reference*
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGContext.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGContextRef` opaque type represents a Quartz 2D drawing destination. A graphics context contains drawing parameters and all device-specific information needed to render the paint on a page to the destination, whether the destination is a window in an application, a bitmap image, a PDF document, or a printer. You can obtain a graphics context by using Quartz graphics context creation functions or by using higher-level functions provided in the Carbon, Cocoa, or Printing frameworks. Quartz provides creation functions for various flavors of Quartz graphics contexts including bitmap images and PDF. The Carbon and Cocoa frameworks provide functions for obtaining window graphics contexts. The Printing framework provides functions that obtain a graphics context appropriate for the destination printer.

Functions by Task

Managing Graphics Contexts

`CGContextFlush` (page 89)
: Forces all pending drawing operations in a window context to be rendered immediately to the destination device.

`CGContextGetTypeID` (page 93)
: Returns the type identifier for Quartz graphics contexts.

`CGContextRelease` (page 96)
: Decrements the retain count of a graphics context.

`CGContextRetain` (page 97)
: Increments the retain count of a graphics context.

`CGContextSynchronize` (page 130)
: Marks a window context for update.

Saving and Restoring the Current Graphics State

`CGContextSaveGState` (page 98)
Pushes a copy of the current graphics state onto the graphics state stack for the context.

`CGContextRestoreGState` (page 97)
Sets the current graphics state to the state most recently saved.

Getting and Setting Graphics State Parameters

`CGContextGetInterpolationQuality` (page 91)
Returns the current level of interpolation quality for a graphics context.

`CGContextSetFlatness` (page 107)
Sets the accuracy of curved paths in a graphics context.

`CGContextSetInterpolationQuality` (page 110)
Sets the level of interpolation quality for a graphics context.

`CGContextSetLineCap` (page 110)
Sets the style for the endpoints of lines drawn in a graphics context.

`CGContextSetLineDash` (page 111)
Sets the pattern for dashed lines in a graphics context.

`CGContextSetLineJoin` (page 112)
Sets the style for the joins of connected lines in a graphics context.

`CGContextSetLineWidth` (page 112)
Sets the line width for a graphics context.

`CGContextSetMiterLimit` (page 113)
Sets the miter limit for the joins of connected lines in a graphics context.

`CGContextSetPatternPhase` (page 113)
Sets the pattern phase of a context.

`CGContextSetFillPattern` (page 106)
Sets the fill pattern in the specified graphics context.

`CGContextSetRenderingIntent` (page 114)
Sets the rendering intent in the current graphics state.

`CGContextSetShouldAntialias` (page 118)
Sets anti-aliasing on or off for a graphics context.

`CGContextSetShouldSmoothFonts` (page 118)
Enables or disables font smoothing in a graphics context.

`CGContextSetStrokePattern` (page 120)
Sets the stroke pattern in the specified graphics context.

`CGContextSetBlendMode` (page 101)
Sets how Quartz composites sample values for a graphics context.

`CGContextSetAllowsAntialiasing` (page 100)
Sets whether or not to allow anti-aliasing for a graphics context.

Constructing Paths

These functions are used to define the geometry of the current path.

`CGContextAddArc` (page 62)
Adds an arc of a circle to the current path, using a center point, radius, and end point.

`CGContextAddArcToPoint` (page 63)
Adds an arc of a circle to the current path, using a radius and tangent points.

`CGContextAddCurveToPoint` (page 64)
Appends a cubic Bézier curve from the current point, using the provided control points and end point .

`CGContextAddLines` (page 66)
Adds a sequence of connected straight-line segments to the current path.

`CGContextAddLineToPoint` (page 67)
Appends a straight line segment from the current point to the provided point .

`CGContextAddPath` (page 67)
Adds a previously created Quartz path object to the current path in a graphics context.

`CGContextAddQuadCurveToPoint` (page 68)
Appends a quadratic Bézier curve from the current point, using a control point and an end point you specify.

`CGContextAddRect` (page 69)
Adds a rectangular path to the current path.

`CGContextAddRects` (page 69)
Adds a set rectangular paths to the current path.

`CGContextBeginPath` (page 70)
Creates a new empty path in a graphics context.

`CGContextClosePath` (page 75)
Closes and terminates an open path.

`CGContextMoveToPoint` (page 94)
Begins a new path at the point you specify.

`CGContextAddEllipseInRect` (page 65)
Adds an ellipse that fits inside the specified rectangle.

Painting Paths

These functions are used to stroke along or fill in the current path.

`CGContextClearRect` (page 72)
Paints a transparent rectangle.

`CGContextDrawPath` (page 81)
Draws the current path using the provided drawing mode.

`CGContextEOFillPath` (page 87)
Paints the area within the current path, using the even-odd fill rule.

`CGContextFillPath` (page 88)
Paints the area within the current path, using the nonzero winding number rule.

`CGContextFillRect` (page 88)
Paints the area contained within the provided rectangle, using the fill color in the current graphics state.

`CGContextFillRects` (page 89)
Paints the areas contained within the provided rectangles, using the fill color in the current graphics state.

`CGContextFillEllipseInRect` (page 87)
Paints the area of the ellipse that fits inside the provided rectangle, using the fill color in the current graphics state.

`CGContextStrokePath` (page 128)
Paints a line along the current path.

`CGContextStrokeRect` (page 128)
Paints a rectangular path.

`CGContextStrokeRectWithWidth` (page 129)
Paints a rectangular path, using the specified line width.

`CGContextReplacePathWithStrokedPath` (page 96)
Replaces the path in the graphics context with the stroked version of the path.

`CGContextStrokeEllipseInRect` (page 127)
Strokes an ellipse that fits inside the specified rectangle.

`CGContextStrokeLineSegments` (page 127)
Strokes a sequence of line segments.

Getting Information About Paths

`CGContextIsPathEmpty` (page 94)
Indicates whether the current path contains any subpaths.

`CGContextGetPathCurrentPoint` (page 92)
Returns the current point in a non-empty path.

`CGContextGetPathBoundingBox` (page 91)
Returns the smallest rectangle that contains the current path.

`CGContextPathContainsPoint` (page 95)
Checks to see whether the specified point is contained in the current path.

Modifying Clipping Paths

`CGContextClip` (page 73)
Modifies the current clipping path, using the nonzero winding number rule.

`CGContextEOClip` (page 86)
Modifies the current clipping path, using the even-odd rule.

`CGContextClipToRect` (page 74)
Sets the clipping path to the intersection of the current clipping path with the area defined by the specified rectangle.

`CGContextClipToRects` (page 75)
Sets the clipping path to the intersection of the current clipping path with the region defined by an array of rectangles.

`CGContextGetClipBoundingBox` (page 90)
Returns the bounding box of a clipping path.

`CGContextClipToMask` (page 73)
Maps a mask into the specified rectangle and intersects it with the current clipping area of the graphics context.

Setting Color, Color Space, and Shadow Values

`CGContextSetAlpha` (page 101)
Sets the opacity level for objects drawn in a graphics context.

`CGContextSetCMYKFillColor` (page 102)
Sets the current fill color to a value in the DeviceCMYK color space.

`CGContextSetFillColor` (page 105)
Sets the current fill color.

`CGContextSetCMYKStrokeColor` (page 104)
Sets the current stroke color to a value in the DeviceCMYK color space.

`CGContextSetFillColorSpace` (page 105)
Sets the fill color space in a graphics context.

`CGContextSetFillColorWithColor` (page 106)
Sets the current fill color in a graphics context, using a Quartz color.

`CGContextSetGrayFillColor` (page 108)
Sets the current fill color to a value in the DeviceGray color space.

`CGContextSetGrayStrokeColor` (page 109)
Sets the current stroke color to a value in the DeviceGray color space.

`CGContextSetRGBFillColor` (page 114)
Sets the current fill color to a value in the DeviceRGB color space.

`CGContextSetRGBStrokeColor` (page 115)
Sets the current stroke color to a value in the DeviceRGB color space.

`CGContextSetShadow` (page 116)
Enables shadowing in a graphics context.

`CGContextSetShadowWithColor` (page 117)
Enables shadowing with color a graphics context.

`CGContextSetStrokeColor` (page 119)
Sets the current stroke color.

`CGContextSetStrokeColorSpace` (page 119)
Sets the stroke color space in a graphics context.

`CGContextSetStrokeColorWithColor` (page 120)
Sets the current stroke color in a context, using a Quartz color.

Transforming User Space

These functions allow you to examine and change the current transformation matrix (CTM) in a graphics context.

`CGContextConcatCTM` (page 76)
Transforms the user coordinate system in a context using a specified matrix.

`CGContextGetCTM` (page 90)
Returns the current transformation matrix.

`CGContextRotateCTM` (page 98)
Rotates the user coordinate system in a context.

`CGContextScaleCTM` (page 99)
Changes the scale of the user coordinate system in a context.

`CGContextTranslateCTM` (page 130)
Changes the origin of the user coordinate system in a context.

Using Transparency Layers

`CGContextBeginTransparencyLayer` (page 71)
Begins a transparency layer.

`CGContextBeginTransparencyLayerWithRect` (page 72)
Begins a transparency layer whose contents are bounded by the specified rectangle.

`CGContextEndTransparencyLayer` (page 86)
Ends a transparency layer.

Drawing an Image to a Graphics Context

`CGContextDrawTiledImage` (page 84)
Repeatedly draws an image, scaled to the provided rectangle, to fill the current clip region.

`CGContextDrawImage` (page 80)
Draws an image into a graphics context.

Drawing PDF Content to a Graphics Context

`CGContextDrawPDFDocument` (page 82)
Draws a page of a PDF document into a graphics context.

`CGContextDrawPDFPage` (page 82)
Draws a page in the current user space of a PDF context.

Drawing With a Gradient

`CGContextDrawLinearGradient` (page 80)
Paints a gradient fill that varies along the line defined by the provided starting and ending points.

CGContextDrawRadialGradient (page 83)
Paints a gradient fill that varies along the area defined by the provided starting and ending circles.

Drawing With a Shading

CGContextDrawShading (page 84)
Fills the clipping path of a context with the specified shading.

Setting Up a Page-Based Graphics Context

CGContextBeginPage (page 70)
Starts a new page in a page-based graphics context.

CGContextEndPage (page 85)
Ends the current page in a page-based graphics context.

Drawing Glyphs

CGContextShowGlyphs (page 123)
Displays an array of glyphs at the current text position.

CGContextShowGlyphsAtPoint (page 123)
Displays an array of glyphs at a position you specify.

CGContextShowGlyphsWithAdvances (page 124)
Draws an array of glyphs with varying offsets.

CGContextShowGlyphsAtPositions (page 124)
Draws glyphs at the provided position.

Drawing Text

CGContextGetTextMatrix (page 92)
Returns the current text matrix.

CGContextGetTextPosition (page 93)
Returns the location at which text is drawn.

CGContextSelectFont (page 100)
Sets the font and font size in a graphics context.

CGContextSetCharacterSpacing (page 102)
Sets the current character spacing.

CGContextSetFont (page 107)
Sets the platform font in a graphics context.

CGContextSetFontSize (page 108)
Sets the current font size.

CGContextSetTextDrawingMode (page 121)
Sets the current text drawing mode.

`CGContextSetTextMatrix` (page 121)
Sets the current text matrix.

`CGContextSetTextPosition` (page 122)
Sets the location at which text is drawn.

`CGContextShowText` (page 125)
Displays a character array at the current text position, a point specified by the current text matrix.

`CGContextShowTextAtPoint` (page 126)
Displays a character string at a position you specify.

Converting Between Device Space and User Space

`CGContextGetUserSpaceToDeviceSpaceTransform` (page 94)
Returns an affine transform that maps user space coordinates to device space coordinates.

`CGContextConvertPointToDeviceSpace` (page 77)
Returns a point that is transformed from user space coordinates to device space coordinates.

`CGContextConvertPointToUserSpace` (page 77)
Returns a point that is transformed from device space coordinates to user space coordinates.

`CGContextConvertSizeToDeviceSpace` (page 79)
Returns a size that is transformed from user space coordinates to device space coordinates.

`CGContextConvertSizeToUserSpace` (page 79)
Returns a size that is transformed from device space coordinates to user space coordinates

`CGContextConvertRectToDeviceSpace` (page 78)
Returns a rectangle that is transformed from user space coordinate to device space coordinates.

`CGContextConvertRectToUserSpace` (page 78)
Returns a rectangle that is transformed from device space coordinate to user space coordinates.

Functions

CGContextAddArc

Adds an arc of a circle to the current path, using a center point, radius, and end point.

```
void CGContextAddArc (
   CGContextRef c,
   CGFloat x,
   CGFloat y,
   CGFloat radius,
   CGFloat startAngle,
   CGFloat endAngle,
   int clockwise
);
```

Parameters

context
A graphics context.

x
The x-value, in user space coordinates, for the center of the arc.

y
The y-value, in user space coordinates, for the center of the arc.

radius
The radius of the arc, in user space coordinates.

startAngle
The angle to the starting point of the arc, measured in radians from the positive x-axis.

endAngle
The angle to the end point of the arc, measured in radians from the positive x-axis.

clockwise
Pass `1` to draw the arc clockwise; `0` otherwise.

Discussion
When you call this function, Quartz builds an arc of a circle centered on the point you provide. The arc is of the specified radius and extends between the start and end point. (You can also use `CGContextAddArc` as a convenient way to draw a circle, by setting the start point to `0` and the end point to `2*Pi`.)

If the current path already contains a subpath, Quartz additionally appends a straight line segment from the current point to the starting point of the arc. If the current path is empty, Quartz creates a new subpath for the arc and does not add the initial straight line segment.

After adding the arc, the current point is reset to the end point of arc (the second tangent point).

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextAddArcToPoint` (page 63)

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextAddArcToPoint

Adds an arc of a circle to the current path, using a radius and tangent points.

```
void CGContextAddArcToPoint (
   CGContextRef c,
   CGFloat x1,
   CGFloat y1,
   CGFloat x2,
   CGFloat y2,
   CGFloat radius
);
```

Parameters

context
A graphics context whose current path is not empty.

x1

The x-value, in user space coordinates, for the end point of the first tangent line. The first tangent line is drawn from the current point to (x1,y1).

y1

The y-value, in user space coordinates, for the end point of the first tangent line. The first tangent line is drawn from the current point to (x1,y1).

x2

The x-value, in user space coordinates, for the end point of the second tangent line. The second tangent line is drawn from (x1,y1) to (x2,y2).

y2

The y-value, in user space coordinates, for the end point of the second tangent line. The second tangent line is drawn from (x1,y1) to (x2,y2).

radius

The radius of the arc, in user space coordinates.

Discussion

This function draws an arc that is tangent to the line from the current point to `(x1,y1)` and to the line from `(x1,y1)` to *(x2,y2)*. The start and end points of the arc are located on the first and second tangent lines, respectively. The start and end points of the arc are also the "tangent points" of the lines.

If the current point and the first tangent point of the arc (the starting point) are not equal, Quartz appends a straight line segment from the current point to the first tangent point. After adding the arc, the current point is reset to the end point of arc (the second tangent point).

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextAddArc` (page 62)

`CGContextAddArcToPoint` (page 63)

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextAddCurveToPoint

Appends a cubic Bézier curve from the current point, using the provided control points and end point .

```
void CGContextAddCurveToPoint (
   CGContextRef c,
   CGFloat cp1x,
   CGFloat cp1y,
   CGFloat cp2x,
   CGFloat cp2y,
   CGFloat x,
   CGFloat y
);
```

Parameters

context

A graphics context whose current path is not empty.

cp1x

The x-value, in user space coordinates, for the first control point of the curve.

cp1y

The y-value, in user space coordinates, for the first control point of the curve.

cp2x

The x-value, in user space coordinates, for the second control point of the curve.

cp2y

The y-value, in user space coordinates, for the second control point of the curve.

x

The x-value, in user space coordinates, at which to end the curve.

y

The y-value, in user space coordinates, at which to end the curve.

Discussion

This function appends a cubic curve to the current path. After adding the segment, the current point is reset from the beginning of the new segment to the end point of that segment.

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextAddQuadCurveToPoint` (page 68)

`CGContextAddArcToPoint` (page 63)

Declared In

`CGContext.h`

CGContextAddEllipseInRect

Adds an ellipse that fits inside the specified rectangle.

```
void CGContextAddEllipseInRect (
   CGContextRef context,
   CGRect rect
);
```

Parameters

context

A graphics context.

rect

A rectangle that defines the area for the ellipse to fit in.

Discussion
The ellipse is approximated by a sequence of Bézier curves. Its center is the midpoint of the rectangle defined by the `rect` parameter. If the rectangle is square, then the ellipse is circular with a radius equal to one-half the width (or height) of the rectangle. If the `rect` parameter specifies a rectangular shape, then the major and minor axes of the ellipse are defined by the width and height of the rectangle.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGContext.h`

CGContextAddLines

Adds a sequence of connected straight-line segments to the current path.

```
void CGContextAddLines (
   CGContextRef c,
   const CGPoint points[],
   size_t count
);
```

Parameters

context

A graphics context .

points

An array of values that specify the start and end points of the line segments to draw. Each point in the array specifies a position in user space. The first point is the array specifies the initial starting point.

count

The number of elements in the `points` array.

Discussion
This is a convenience function that adds a sequence of connected line segments to the current path in a graphics context. Quartz connects each point in the array with the subsequent point in the array, using straight line segments.

On return, the current point is the last point in the array. This function does not automatically close the path created by the line segments. If you want to close the path, you must call `CGContextClosePath` (page 75).

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextAddLineToPoint` (page 67)

Declared In
`CGContext.h`

CGContextAddLineToPoint

Appends a straight line segment from the current point to the provided point .

```
void CGContextAddLineToPoint (
   CGContextRef c,
   CGFloat x,
   CGFloat y
);
```

Parameters

context

A graphics context whose current path is not empty.

x

The x-value, in user space coordinates, for the end of the line segment.

y

The y-value, in user space coordinates, for the end of the line segment.

Discussion

After adding the line segment, the current point is reset from the beginning of the new line segment to the endpoint of that line segment.

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextAddLines` (page 66)

Related Sample Code

CALayerEssentials

CarbonSketch

HID Calibrator

HID Explorer

Declared In

`CGContext.h`

CGContextAddPath

Adds a previously created Quartz path object to the current path in a graphics context.

```
void CGContextAddPath (
   CGContextRef context,
   CGPathRef path
);
```

Parameters

context

A graphics context .

path

A previously created Quartz path object. See *CGPath Reference*.

Discussion
Quartz applies the current transformation matrix (CTM) to the points in the new path before they are added to the current path in the graphics context.

Availability
Available in Mac OS X version 10.2 and later.

Related Sample Code
CALayerEssentials

Declared In
`CGContext.h`

CGContextAddQuadCurveToPoint

Appends a quadratic Bézier curve from the current point, using a control point and an end point you specify.

```
void CGContextAddQuadCurveToPoint (
   CGContextRef c,
   CGFloat cpx,
   CGFloat cpy,
   CGFloat x,
   CGFloat y
);
```

Parameters

context
A graphics context whose current path is not empty.

cpx
The x-coordinate of the user space for the control point of the curve.

cpy
The y-coordinate of the user space for the control point of the curve.

x
The x-coordinate of the user space at which to end the curve.

y
The y-coordinate of the user space at which to end the curve.

Discussion
This function appends a quadratic curve to the current subpath. After adding the segment, the current point is reset from the beginning of the new segment to the end point of that segment.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextAddCurveToPoint` (page 64)
`CGContextAddArcToPoint` (page 63)

Declared In
`CGContext.h`

CGContextAddRect

Adds a rectangular path to the current path.

```
void CGContextAddRect (
   CGContextRef c,
   CGRect rect
);
```

Parameters

context

A graphics context.

rect

A rectangle, specified in user space coordinates.

Availability

Available in Mac OS X version 10.0 and later.

See Also

CGContextAddRects (page 69)

Related Sample Code

CarbonSketch

Declared In

CGContext.h

CGContextAddRects

Adds a set rectangular paths to the current path.

```
void CGContextAddRects (
   CGContextRef c,
   const CGRect rects[],
   size_t count
);
```

Parameters

context

A graphics context.

rects

An array of rectangles, specified in user space coordinates.

count

The number of rectangles in the `rects` array.

Availability

Available in Mac OS X version 10.0 and later.

See Also

CGContextAddRect (page 69)

Declared In

CGContext.h

CGContextBeginPage

Starts a new page in a page-based graphics context.

```
void CGContextBeginPage (
   CGContextRef c,
   const CGRect *mediaBox
);
```

Parameters

context

A page-based graphics context such as a PDF context. If you specify a context that does not support multiple pages, this function does nothing.

mediaBox

A Quartz rectangle defining the bounds of the new page, expressed in units of the default user space, or `NULL`. These bounds supersede any supplied for the media box when you created the context. If you pass `NULL`, Quartz uses the rectangle you supplied for the media box when the graphics context was created.

Discussion

When using a graphics context that supports multiple pages, you should call this function together with `CGContextEndPage` (page 85) to delineate the page boundaries in the output. In other words, each page should be bracketed by calls to `CGContextBeginPage` and `CGContextEndPage`. Quartz ignores all drawing operations performed outside a page boundary in a page-based context.

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextBeginPath

Creates a new empty path in a graphics context.

```
void CGContextBeginPath (
   CGContextRef c
);
```

Parameters

context

A graphics context.

Discussion

A graphics context can have only a single path in use at any time. If the specified context already contains a current path when you call this function, Quartz replaces the previous current path with the new path. In this case, Quartz discards the old path and any data associated with it.

The current path is not part of the graphics state. Consequently, saving and restoring the graphics state has no effect on the current path.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextClosePath` (page 75)

Related Sample Code
CarbonSketch
HID Calibrator
HID Explorer

Declared In
`CGContext.h`

CGContextBeginTransparencyLayer

Begins a transparency layer.

```
void CGContextBeginTransparencyLayer (
   CGContextRef context,
   CFDictionaryRef auxiliaryInfo
);
```

Parameters

context
A graphics context.

auxiliaryInfo
A dictionary that specifies any additional information, or `NULL`.

Discussion
Until a corresponding call to `CGContextEndTransparencyLayer` (page 86), all subsequent drawing operations in the specified context are composited into a fully transparent backdrop (which is treated as a separate destination buffer from the context).

After a call to `CGContextEndTransparencyLayer`, the result is composited into the context using the global alpha and shadow state of the context. This operation respects the clipping region of the context.

After a call to this function, all of the parameters in the graphics state remain unchanged with the exception of the following:

- The global alpha is set to `1`.
- The shadow is turned off.

Ending the transparency layer restores these parameters to their previous values. Quartz maintains a transparency layer stack for each context, and transparency layers may be nested.

Tip: For best performance, make sure that you set the smallest possible clipping area for the objects in the transparency layer prior to calling `CGContextBeginTransparencyLayer`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
CGContext.h

CGContextBeginTransparencyLayerWithRect

Begins a transparency layer whose contents are bounded by the specified rectangle.

```
void CGContextBeginTransparencyLayerWithRect(CGContextRef context, CGRect rect,
CFDictionaryRef auxiliaryInfo);
```

Parameters

context
A graphics context.

rect
The rectangle, specified in user space, that bounds the transparency layer.

auxiliaryInfo
A dictionary that specifies any additional information, or NULL.

Discussion
This function is identical to CGContextBeginTransparencyLayer (page 71) except that the content of the transparency layer is within the bounds of the provided rectangle.

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

Declared In
CGContext.h

CGContextClearRect

Paints a transparent rectangle.

```
void CGContextClearRect (
   CGContextRef c,
   CGRect rect
);
```

Parameters

context
The graphics context in which to paint the rectangle.

rect
The rectangle, in user space coordinates.

Discussion
If the provided context is a window or bitmap context, Quartz effectively clears the rectangle. For other context types, Quartz fills the rectangle in a device-dependent manner. However, you should not use this function in contexts other than window or bitmap contexts.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
CGContext.h

CGContextClip

Modifies the current clipping path, using the nonzero winding number rule.

```
void CGContextClip (
   CGContextRef c
);
```

Parameters

context

A graphics context that contains a path. If the context does not have a current path, the function does nothing.

Discussion
The function uses the nonzero winding number rule to calculate the intersection of the current path with the current clipping path. Quartz then uses the path resulting from the intersection as the new current clipping path for subsequent painting operations.

Unlike the current path, the current clipping path is part of the graphics state. Therefore, to re-enlarge the paintable area by restoring the clipping path to a prior state, you must save the graphics state before you clip and restore the graphics state after you've completed any clipped drawing.

After determining the new clipping path, the function resets the context's current path to an empty path.

See alsoCGContextEOClip (page 86)

Availability
Available in Mac OS X version 10.0 and later.

Declared In
CGContext.h

CGContextClipToMask

Maps a mask into the specified rectangle and intersects it with the current clipping area of the graphics context.

```
void CGContextClipToMask (
   CGContextRef c,
   CGRect rect,
   CGImageRef mask
);
```

Parameters

c

A graphics context.

rect

The rectangle to map the `mask` parameter to.

mask

An image or an image mask. If `mask` is an image, then it must be in the DeviceGray color space, may not have an alpha component, and may not be masked by an image mask or masking color.

Discussion
If the *mask* parameter is an image mask, then Quartz clips in a manner identical to the behavior seen with the function `CGContextDrawImage`—the mask indicates an area to be left unchanged when drawing. The source samples of the image mask determine which points of the clipping area are changed, acting as an "inverse alpha" value. If the value of a source sample in the image mask is `S`, then the corresponding point in the current clipping area is multiplied by an alpha value of (`1-S`). For example, if `S` is `1` then the point in the clipping area becomes transparent. If `S` is `0`, the point in the clipping area is unchanged.

If the *mask* parameter is an image, then *mask* acts like an alpha mask and is blended with the current clipping area. The source samples of mask determine which points of the clipping area are changed. If the value of the source sample in mask is `S`, then the corresponding point in the current clipping area is multiplied by an alpha of `S`. For example, if `S` is `0`, then the point in the clipping area becomes transparent. If `S` is `1`, the point in the clipping area is unchanged.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGContext.h`

CGContextClipToRect

Sets the clipping path to the intersection of the current clipping path with the area defined by the specified rectangle.

```
void CGContextClipToRect (
   CGContextRef c,
   CGRect rect
);
```

Parameters

context

The graphics context for which to set the clipping path.

rect

A `CGRect` value that specifies, in the user space, the location and dimensions of the rectangle to be used in determining the new clipping path.

Discussion
This function sets the specified graphics context's clipping region to the area which intersects both the current clipping path and the specified rectangle.

After determining the new clipping path, the `CGContextClipToRect` function resets the context's current path to an empty path.

See also `CGContextClipToRects` (page 75).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
CGContext.h

CGContextClipToRects

Sets the clipping path to the intersection of the current clipping path with the region defined by an array of rectangles.

```
void CGContextClipToRects (
   CGContextRef c,
   const CGRect rects[],
   size_t count
);
```

Parameters

context
The graphics context for which to set the clipping path.

rects
An array of rectangles. The locations and dimensions of the rectangles are specified in the user space coordinate system.

count
The total number of array entries in the `rects` parameter.

Discussion
This function sets the clipping path to the intersection of the current clipping path and the region within the specified rectangles.

After determining the new clipping path, the function resets the context's current path to an empty path.

See also `CGContextClipToRect` (page 74).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
CGContext.h

CGContextClosePath

Closes and terminates an open path.

```
void CGContextClosePath (
   CGContextRef c
);
```

Parameters

context
A graphics context.

Discussion
If a path is open, this function closes and terminate the path. Quartz closes a path by drawing a straight line that connects the current point to the starting point. If the current point and the starting point are the same, you must still call this function to close the path. After Quartz terminates the path, the current point is no longer defined. If there is no open path, this function does nothing.

When you fill or clip an open path, Quartz implicitly closes the subpath for you.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextAddPath` (page 67)

Related Sample Code
CALayerEssentials
CarbonSketch
HID Explorer

Declared In
`CGContext.h`

CGContextConcatCTM

Transforms the user coordinate system in a context using a specified matrix.

```
void CGContextConcatCTM (
   CGContextRef c,
   CGAffineTransform transform
);
```

Parameters

context
A graphics context.

transform
The transformation matrix to apply to the specified context's current transformation matrix.

Discussion
When you call the function `CGContextConcatCTM`, it concatenates (that is, it combines) two matrices, by multiplying them together. The order in which matrices are concatenated is important, as the operations are not commutative. When you call `CGContextConcatCTM`, the resulting CTM in the context is: `CTMnew = transform * CTMcontext.`

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextConvertPointToDeviceSpace

Returns a point that is transformed from user space coordinates to device space coordinates.

```
CGPoint CGContextConvertPointToDeviceSpace (
   CGContextRef c,
   CGPoint point
);
```

Parameters

c

A graphics context.

point

The point, in user space coordinates, to transform.

Return Value

The coordinates of the point in device space coordinates.

Availability

Available in Mac OS X v10.4 and later.

See Also

CGContextConvertPointToUserSpace (page 77)

Declared In

CGContext.h

CGContextConvertPointToUserSpace

Returns a point that is transformed from device space coordinates to user space coordinates.

```
CGPoint CGContextConvertPointToUserSpace (
   CGContextRef c,
   CGPoint point
);
```

Parameters

c

A graphics context.

point

The point, in device space coordinates, to transform.

Return Value

The coordinates of the point in user space coordinates.

Availability

Available in Mac OS X v10.4 and later.

See Also

CGContextConvertPointToDeviceSpace (page 77)

Declared In

CGContext.h

CGContextConvertRectToDeviceSpace

Returns a rectangle that is transformed from user space coordinate to device space coordinates.

```
CGRect CGContextConvertRectToDeviceSpace (
   CGContextRef c,
   CGRect rect
);
```

Parameters

context

A graphics context.

rect

The rectangle, in user space coordinates, to transform.

Return Value

The rectangle in device space coordinates.

Discussion

In general affine transforms do not preserve rectangles. As a result, this function returns the smallest rectangle that contains the transformed corner points of the rectangle.

Availability

Available in Mac OS X v10.4 and later.

See Also

CGContextConvertRectToUserSpace (page 78)

Declared In

CGContext.h

CGContextConvertRectToUserSpace

Returns a rectangle that is transformed from device space coordinate to user space coordinates.

```
CGRect CGContextConvertRectToUserSpace (
   CGContextRef c,
   CGRect rect
);
```

Parameters

context

A graphics context.

rect

The rectangle, in device space coordinates, to transform.

Return Value

The rectangle in user space coordinates.

Discussion

In general, affine transforms do not preserve rectangles. As a result, this function returns the smallest rectangle that contains the transformed corner points of the rectangle.

Availability

Available in Mac OS X v10.4 and later.

See Also
CGContextConvertRectToDeviceSpace (page 78)

Declared In
CGContext.h

CGContextConvertSizeToDeviceSpace

Returns a size that is transformed from user space coordinates to device space coordinates.

```
CGSize CGContextConvertSizeToDeviceSpace (
   CGContextRef c,
   CGSize size
);
```

Parameters

c
A graphics context.

size
The size, in user space coordinates, to transform.

Return Value
The size in device space coordinates.

Availability
Available in Mac OS X v10.4 and later.

See Also
CGContextConvertSizeToUserSpace (page 79)

Declared In
CGContext.h

CGContextConvertSizeToUserSpace

Returns a size that is transformed from device space coordinates to user space coordinates

```
CGSize CGContextConvertSizeToUserSpace (
   CGContextRef c,
   CGSize size
);
```

Parameters

context
A graphics context.

size
The size, in device space coordinates, to transform.

Return Value
The size in user space coordinates.

Availability
Available in Mac OS X v10.4 and later.

See Also
CGContextConvertSizeToDeviceSpace (page 79)

Declared In
CGContext.h

CGContextDrawImage

Draws an image into a graphics context.

```
void CGContextDrawImage (
   CGContextRef c,
   CGRect rect,
   CGImageRef image
);
```

Parameters

context
The graphics context in which to draw the image.

rect
The location and dimensions in user space of the bounding box in which to draw the image.

image
The image to draw.

Discussion
Quartz scales the image—disproportionately, if necessary—to fit the bounds specified by the `rect` parameter.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonCocoa_PictureCursor
WhackedTV

Declared In
CGContext.h

CGContextDrawLinearGradient

Paints a gradient fill that varies along the line defined by the provided starting and ending points.

```
void CGContextDrawLinearGradient(
    CGContextRef context,
    CGGradientRef gradient,
    CGPoint startPoint,
    CGPoint endPoint,
    CGGradientDrawingOptions options
);
```

Parameters

context
A Quartz graphics context.

gradient
A CGGradient object.

startPoint
The coordinate that defines the starting point of the gradient.

endPoint
The coordinate that defines the ending point of the gradient.

options
Option flags (kCGGradientDrawsBeforeStartLocation (page 201) or kCGGradientDrawsAfterEndLocation (page 201)) that control whether the fill is extended beyond the starting or ending point.

Discussion
The color at location 0 in the CGGradient object is mapped to the starting point. The color at location 1 in the CGGradient object is mapped to the ending point. Colors are linearly interpolated between these two points based on the location values of the gradient. The option flags control whether the gradient is drawn before the start point or after the end point.

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

Declared In
CGContext.h

CGContextDrawPath

Draws the current path using the provided drawing mode.

```
void CGContextDrawPath (
   CGContextRef c,
   CGPathDrawingMode mode
);
```

Parameters

context
A graphics context that contains a path to paint.

mode
A path drawing mode constant—kCGPathFill, kCGPathEOFill, kCGPathStroke, kCGPathFillStroke, or kCGPathEOFillStroke. For a discussion of these constants, see *CGPath Reference*.

Discussion
This function draws the current path using the specified drawing mode. If the current path contains several disjoint portions (or subpaths), Quartz fills each one independently. Any subpath that you did not explicitly close by calling CGContextClosePath (page 75) is closed implicitly by the fill routines.

Availability
Available in Mac OS X version 10.0 and later.

See Also
CGContextFillPath (page 88)
CGContextEOFillPath (page 87)
CGContextStrokePath (page 128)

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextDrawPDFDocument

Draws a page of a PDF document into a graphics context.

```
void CGContextDrawPDFDocument (
   CGContextRef c,
   CGRect rect,
   CGPDFDocumentRef document,
   int page
);
```

Parameters

context

The graphics context in which to draw the PDF page.

rect

A `CGRect` value that specifies the dimensions and location of the area in which to draw the PDF page, in units of the user space. When drawn, Quartz scales the media box of the page to fit the rectangle you specify.

document

The PDF document to draw.

page

A value that specifies the PDF page number to draw. If the specified page does not exist, the function does nothing.

Special Considerations

For applications running in Mac OS X version 10.3 and later, it is recommended that you replace this function with `CGContextDrawPDFPage` (page 82). If you do so, and want to specify the drawing rectangle, you should use `CGPDFPageGetDrawingTransform` (page 349) to get an appropriate transform, concatenate it with the current transformation matrix, clip to the rectangle, and then call `CGContextDrawPDFPage` (page 82).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextDrawPDFPage

Draws a page in the current user space of a PDF context.

```
void CGContextDrawPDFPage (
   CGContextRef c,
   CGPDFPageRef page
);
```

Parameters

context

The graphics context in which to draw the PDF page.

page

A Quartz PDF page.

Discussion

This function works in conjunction with the opaque type `CGPDFPageRef` to draw individual pages into a PDF context.

For applications running in Mac OS X version 10.3 and later, this function is recommended as a replacement for the older function `CGContextDrawPDFDocument`.

Availability

Available in Mac OS X version 10.3 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextDrawRadialGradient

Paints a gradient fill that varies along the area defined by the provided starting and ending circles.

```
void CGContextDrawRadialGradient(
    CGContextRef context,
    CGGradientRef gradient,
    CGPoint startCenter,
    CGFloat startRadius,
    CGPoint endCenter,
    CGFloat endRadius,
    CGGradientDrawingOptions options
);
```

Parameters

context

A Quartz graphics context.

gradient

A CGGradient object.

startCenter

The coordinate that defines the center of the starting circle.

startRadius

The radius of the starting circle.

endCenter

The coordinate that defines the center of the ending circle.

endRadius
The radius of the ending circle.

options
Option flags (`kCGGradientDrawsBeforeStartLocation` (page 201) or `kCGGradientDrawsAfterEndLocation` (page 201)) that control whether the gradient is drawn before the starting circle or after the ending circle.

Discussion
The color at location 0 in the CGGradient object is mapped to the circle defined by `startCenter` and `startRadius`. The color at location 1 in the CGGradient object is mapped to the circle defined by `endCenter` and `endRadius`. Colors are linearly interpolated between the starting and ending circles based on the location values of the gradient. The option flags control whether the gradient is drawn before the start point or after the end point.

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

Declared In
`CGContext.h`

CGContextDrawShading

Fills the clipping path of a context with the specified shading.

```
void CGContextDrawShading (
   CGContextRef c,
   CGShadingRef shading
);
```

Parameters

context
The graphics context in which to draw the shading.

shading
A Quartz shading. Quartz retains this object; upon return, you may safely release it.

Discussion
In Mac OS X v10.5 and later, the preferred way to draw gradients is to use a CGGradient object. See *CGGradient Reference*.

Availability
Available in Mac OS X v10.2 and later.

See Also
`CGContextDrawLinearGradient` (page 80)
`CGContextDrawRadialGradient` (page 83)

Declared In
`CGContext.h`

CGContextDrawTiledImage

Repeatedly draws an image, scaled to the provided rectangle, to fill the current clip region.

```
void CGContextDrawTiledImage(
    CGContextRef context,
    CGRect rect,
    CGImageRef image
);
```

Parameters

context

The graphics context in which to draw the image.

rect

A rectangle that specifies the tile size. Quartz scales the image—disproportionately, if necessary—to fit the bounds specified by the `rect` parameter.

image

The image to draw.

Discussion

Quartz draws the scaled image starting at the origin of user space, then moves to a new point (horizontally by the width of the tile and/or vertically by the height of the tile), draws the scaled image, moves again, draws again, and so on, until the current clip region is tiled with copies of the image. Unlike patterns, the image is tiled in user space, so transformations applied to the CTM affect the final result.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CGContext.h`

CGContextEndPage

Ends the current page in a page-based graphics context.

```
void CGContextEndPage (
   CGContextRef c
);
```

Parameters

context

A page-based graphics context.

Discussion

When using a graphics context that supports multiple pages, you should call this function to terminate drawing in the current page.

For more information, see `CGContextBeginPage` (page 70).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextEndTransparencyLayer

Ends a transparency layer.

```
void CGContextEndTransparencyLayer (
   CGContextRef context
);
```

Parameters

context

A graphics context.

Discussion

See the discussion for `CGContextBeginTransparencyLayer` (page 71).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGContext.h`

CGContextEOClip

Modifies the current clipping path, using the even-odd rule.

```
void CGContextEOClip (
   CGContextRef c
);
```

Parameters

context

A graphics context containing a path. If the context does not have a current path, the function does nothing.

Discussion

The function uses the even-odd rule to calculate the intersection of the current path with the current clipping path. Quartz then uses the path resulting from the intersection as the new current clipping path for subsequent painting operations.

Unlike the current path, the current clipping path is part of the graphics state. Therefore, to re-enlarge the paintable area by restoring the clipping path to a prior state, you must save the graphics state before you clip and restore the graphics state after you've completed any clipped drawing.

After determining the new clipping path, the function resets the context's current path to an empty path.

See also `CGContextClip` (page 73).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextEOFillPath

Paints the area within the current path, using the even-odd fill rule.

```
void CGContextEOFillPath (
   CGContextRef c
);
```

Parameters

context

A graphics context that contains a path to fill.

Discussion

If the current path contains several disjoint portions (or subpaths), Quartz fills each one independently. Any subpath that you did not explicitly close by calling `CGContextClosePath` (page 75) is closed implicitly by the fill routines.

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextFillPath` (page 88)

`CGContextStrokePath` (page 128)

`CGContextDrawPath` (page 81)

Related Sample Code

CALayerEssentials

Declared In

`CGContext.h`

CGContextFillEllipseInRect

Paints the area of the ellipse that fits inside the provided rectangle, using the fill color in the current graphics state.

```
void CGContextFillEllipseInRect (
   CGContextRef context,
   CGRect rect
);
```

Parameters

context

A graphics context.

rect

A rectangle that defines the area for the ellipse to fit in.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

HID Calibrator

HID Config Save

HID Explorer

Declared In
CGContext.h

CGContextFillPath

Paints the area within the current path, using the nonzero winding number rule.

```
void CGContextFillPath (
   CGContextRef c
);
```

Parameters

context
A graphics context that contains a path to fill.

Discussion
If the current path contains several disjoint portions (or subpaths), Quartz fills each one independently. Any subpath that you did not explicitly close by calling `CGContextClosePath` (page 75) is closed implicitly by the fill routines.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextEOFillPath` (page 87)
`CGContextStrokePath` (page 128)
`CGContextDrawPath` (page 81)

Declared In
CGContext.h

CGContextFillRect

Paints the area contained within the provided rectangle, using the fill color in the current graphics state.

```
void CGContextFillRect (
   CGContextRef c,
   CGRect rect
);
```

Parameters

context
A graphics context.

rect
A rectangle, in user space coordinates.

Discussion
As a side effect when you call this function, Quartz clears the current path.

Availability
Available in Mac OS X version 10.0 and later.

See Also
CGContextFillRects (page 89)

Related Sample Code
CALayerEssentials
CarbonSketch
HID Calibrator
HID Explorer

Declared In
CGContext.h

CGContextFillRects

Paints the areas contained within the provided rectangles, using the fill color in the current graphics state.

```
void CGContextFillRects (
   CGContextRef c,
   const CGRect rects[],
   size_t count
);
```

Parameters

context
A graphics context .

rects
An array of rectangles, in user space coordinates.

count
The number rectangles in the `rects` array.

Discussion
As a side effect when you call this function, Quartz clears the current path.

Availability
Available in Mac OS X version 10.0 and later.

See Also
CGContextFillRect (page 88)

Declared In
CGContext.h

CGContextFlush

Forces all pending drawing operations in a window context to be rendered immediately to the destination device.

```
void CGContextFlush (
   CGContextRef c
);
```

Parameters

context

The window context to flush. If you pass a PDF context or a bitmap context, this function does nothing.

Discussion

When you call this function, Quartz immediately flushes the current drawing to the destination device (for example, a screen). Because the system software flushes a context automatically at the appropriate times, calling this function could have an adverse effect on performance. Under normal conditions, you do not need to call this function.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextGetClipBoundingBox

Returns the bounding box of a clipping path.

```
CGRect CGContextGetClipBoundingBox (
   CGContextRef c
);
```

Parameters

context

The graphics context to modify.

Return Value

The bounding box of the clipping path, specified in user space.

Discussion

The bounding box is the smallest rectangle completely enclosing all points in the clipping path, including control points for any Bezier curves in the path.

Availability

Available in Mac OS X version 10.3 and later.

Related Sample Code

CALayerEssentials

Declared In

`CGContext.h`

CGContextGetCTM

Returns the current transformation matrix.

```
CGAffineTransform CGContextGetCTM (
   CGContextRef c
);
```

Parameters

context

A graphics context.

Return Value
The transformation matrix for the current graphics state of the specified context.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
CGContext.h

CGContextGetInterpolationQuality

Returns the current level of interpolation quality for a graphics context.

```
CGInterpolationQuality CGContextGetInterpolationQuality (
   CGContextRef c
);
```

Parameters

context

The graphics context to examine.

Return Value
The current level of interpolation quality.

Discussion
Interpolation quality is a graphics state parameter that provides a hint for the level of quality to use for image interpolation (for example, when scaling the image). Not all contexts support all interpolation quality levels.

Availability
Available in Mac OS X version 10.1 and later.

See Also
CGContextSetInterpolationQuality (page 110)

Declared In
CGContext.h

CGContextGetPathBoundingBox

Returns the smallest rectangle that contains the current path.

```
CGRect CGContextGetPathBoundingBox (
   CGContextRef c
);
```

Parameters

context

The graphics context, containing a path, to examine.

Return Value

A `CGRect` value that specifies the dimensions and location, in user space, of the bounding box of the path. If there is no path, the function returns `CGRectNull`.

Discussion

The bounding box is the smallest rectangle completely enclosing all points in a path, including control points for Bézier cubic and quadratic curves.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextGetPathCurrentPoint

Returns the current point in a non-empty path.

```
CGPoint CGContextGetPathCurrentPoint (
   CGContextRef c
);
```

Parameters

context

The graphics context containing the path to examine.

Return Value

A `CGPoint` value that specifies the location, in user space, of current point in the context's path. If there is no path, the function returns `CGPointZero`.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextGetTextMatrix

Returns the current text matrix.

```
CGAffineTransform CGContextGetTextMatrix (
   CGContextRef c
);
```

Parameters

context

The graphics context for which to obtain the text matrix.

Return Value
The current text matrix.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
CGContext.h

CGContextGetTextPosition

Returns the location at which text is drawn.

```
CGPoint CGContextGetTextPosition (
   CGContextRef c
);
```

Parameters

context

The graphics context from which to obtain the current text position.

Return Value
Returns a `CGPoint` value that specifies the x and y values at which text is to be drawn, in user space coordinates.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
CGContext.h

CGContextGetTypeID

Returns the type identifier for Quartz graphics contexts.

```
CFTypeID CGContextGetTypeID (
   void
);
```

Return Value
The identifier for the opaque type `CGContextRef` (page 131).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CGContext.h

CGContextGetUserSpaceToDeviceSpaceTransform

Returns an affine transform that maps user space coordinates to device space coordinates.

```
CGAffineTransform CGContextGetUserSpaceToDeviceSpaceTransform (
   CGContextRef c
);
```

Parameters

c

A graphics context.

Return Value

The affine transforms that maps the user space of the graphics context to the device space.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGContext.h`

CGContextIsPathEmpty

Indicates whether the current path contains any subpaths.

```
bool CGContextIsPathEmpty (
   CGContextRef c
);
```

Parameters

context

The graphics context containing the path to examine.

Return Value

Returns `1` if the context's path contains no subpaths, otherwise returns `0`.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextMoveToPoint

Begins a new path at the point you specify.

```
void CGContextMoveToPoint (
   CGContextRef c,
   CGFloat x,
   CGFloat y
);
```

Parameters

context

A graphics context.

x
The x-value, in user space coordinates, for the point.

y
The y-value, in user space coordinates, for the point.

Discussion
This point you specifies becomes the current point. It defines the starting point of the next line segment.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CALayerEssentials
CarbonSketch
HID Calibrator
HID Explorer

Declared In
`CGContext.h`

CGContextPathContainsPoint

Checks to see whether the specified point is contained in the current path.

```
bool CGContextPathContainsPoint (
   CGContextRef context,
   CGPoint point,
   CGPathDrawingMode mode
);
```

Parameters

context
A graphics context.

point
The point to check, specified in user space units.

mode
A path drawing mode—`kCGPathFill`, `kCGPathEOFill`, `kCGPathStroke`, `kCGPathFillStroke`, or `kCGPathEOFillStroke`. See `CGPathDrawingMode` for more information on these modes.

Return Value
Returns `true` if `point` is inside the current path of the graphics context; `false` otherwise.

Discussion
A point is contained within the path of a graphics context if the point is inside the painted region when the path is stroked or filled with opaque colors using the specified path drawing mode. A point can be inside a path only if the path is explicitly closed by calling the function `CGContextClosePath` (page 75), for paths drawn directly to the current context, or `CGPathCloseSubpath` (page 265), for paths first created as `CGPath` objects and then drawn to the current context.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGContext.h

CGContextRelease

Decrements the retain count of a graphics context.

```
void CGContextRelease (
   CGContextRef c
);
```

Parameters

context
The graphics context to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the `context` parameter is `NULL`.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
CGContext.h

CGContextReplacePathWithStrokedPath

Replaces the path in the graphics context with the stroked version of the path.

```
void CGContextReplacePathWithStrokedPath (
   CGContextRef c
);
```

Parameters

c
A graphics context.

Discussion
Quartz creates a stroked path using the parameters of the current graphics context. You can use this path in the same way you use the path of any context. For example, you can clip to the stroked version of a path by calling this function followed by a call to the function `CGContextClip` (page 73).

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGContext.h

CGContextRestoreGState

Sets the current graphics state to the state most recently saved.

```
void CGContextRestoreGState (
   CGContextRef c
);
```

Parameters

context

The graphics context whose state you want to modify.

Discussion

Quartz removes the graphics state that is at the top of the stack so that the most recently saved state becomes the current graphics state.

Availability

Available in Mac OS X version 10.0 and later.

See Also

`CGContextSaveGState` (page 98)

Related Sample Code

CarbonSketch

HID Calibrator

Declared In

`CGContext.h`

CGContextRetain

Increments the retain count of a graphics context.

```
CGContextRef CGContextRetain (
   CGContextRef c
);
```

Parameters

context

The graphics context to retain.

Return Value

The same graphics context you passed in as the `context` parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `context` parameter is `NULL`.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextRotateCTM

Rotates the user coordinate system in a context.

```
void CGContextRotateCTM (
   CGContextRef c,
   CGFloat angle
);
```

Parameters

context

A graphics context.

angle

The angle, in radians, by which to rotate the coordinate space of the specified context. (Positive values rotate counterclockwise.)

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextSaveGState

Pushes a copy of the current graphics state onto the graphics state stack for the context.

```
void CGContextSaveGState (
   CGContextRef c
);
```

Parameters

context

The graphics context whose current graphics state you want to save.

Discussion
Each graphics context maintains a stack of graphics states. Note that not all aspects of the current drawing environment are elements of the graphics state. For example, the current path is not considered part of the graphics state and is therefore not saved when you call the `CGContextSaveGState` function. The graphics state parameters that *are* saved are:

- CTM (current transformation matrix)
- clip region
- image interpolation quality
- line width
- line join
- miter limit
- line cap
- line dash
- flatness
- should anti-alias

- rendering intent
- fill color space
- stroke color space
- fill color
- stroke color
- alpha value
- font
- font size
- character spacing
- text drawing mode
- shadow parameters
- the pattern phase
- the font smoothing parameter
- blend mode

To restore your drawing environment to a previously saved state, you can use the function `CGContextRestoreGState` (page 97).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch
HID Calibrator

Declared In
`CGContext.h`

CGContextScaleCTM

Changes the scale of the user coordinate system in a context.

```
void CGContextScaleCTM (
   CGContextRef c,
   CGFloat sx,
   CGFloat sy
);
```

Parameters

context
A graphics context.

sx
The factor by which to scale the x-axis of the coordinate space of the specified context.

sy
The factor by which to scale the y-axis of the coordinate space of the specified context.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextSelectFont

Sets the font and font size in a graphics context.

```
void CGContextSelectFont (
   CGContextRef c,
   const char *name,
   CGFloat size,
   CGTextEncoding textEncoding
);
```

Parameters

context
The graphics context for which to set the font and font size.

name
A null-terminated string that contains the PostScript name of the font to set.

size
A value that specifies the font size to set, in text space units.

textEncoding
A `CGTextEncoding` value that specifies the encoding used for the font. For a description of the available values, see "Text Encodings" (page 140).

Discussion
For information about when to use this function, see `CGContextShowText` (page 125) and `CGContextShowTextAtPoint` (page 126).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
HID Calibrator

Declared In
`CGContext.h`

CGContextSetAllowsAntialiasing

Sets whether or not to allow anti-aliasing for a graphics context.

```
void CGContextSetAllowsAntialiasing (
   CGContextRef context,
   bool allowsAntialiasing
);
```

Parameters

context

A graphics context.

allowsAntialiasing

A Boolean value that specifies whether or not to allow antialiasing. Pass `true` to allow antialiasing; `false` otherwise. This parameter is not part of the graphics state.

Discussion

Quartz performs antialiasing for a graphics context if both the `allowsAntialiasing` parameter and the graphics state parameter `shouldAntialias` are `true`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGContext.h`

CGContextSetAlpha

Sets the opacity level for objects drawn in a graphics context.

```
void CGContextSetAlpha (
   CGContextRef c,
   CGFloat alpha
);
```

Parameters

context

The graphics context for which to set the current graphics state's alpha value parameter.

alpha

A value that specifies the opacity level. Values can range from `0.0` (transparent) to `1.0` (opaque). Values outside this range are clipped to `0.0` or `1.0`.

Discussion

This function sets the alpha value parameter for the specified graphics context. To clear the contents of the drawing canvas, you should use the function `CGContextClearRect` (page 72).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetBlendMode

Sets how Quartz composites sample values for a graphics context.

```
void CGContextSetBlendMode (
   CGContextRef context,
   CGBlendMode mode
);
```

Parameters

context

The graphics context to modify.

mode

A blend mode. See "Blend Modes" (page 131) for a list of the constants you can supply.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGContext.h

CGContextSetCharacterSpacing

Sets the current character spacing.

```
void CGContextSetCharacterSpacing (
   CGContextRef c,
   CGFloat spacing
);
```

Parameters

context

The graphics context for which to set the character spacing.

spacing

A value that represents the amount of additional space to place between glyphs, in text space coordinates.

Discussion
Quartz adds the additional space to the advance between the origin of one character and the origin of the next character. For information about the text coordinate system, see `CGContextSetTextMatrix` (page 121).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
CGContext.h

CGContextSetCMYKFillColor

Sets the current fill color to a value in the DeviceCMYK color space.

```
void CGContextSetCMYKFillColor (
   CGContextRef c,
   CGFloat cyan,
   CGFloat magenta,
   CGFloat yellow,
   CGFloat black,
   CGFloat alpha
);
```

Parameters

context

The graphics context for which to set the current fill color.

cyan

The cyan intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

magenta

The magenta intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

yellow

The yellow intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

black

The black intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from 0.0 (does not absorb the secondary color) to 1.0 (fully absorbs the secondary color).

alpha

A value that specifies the opacity level. Values can range from `0.0` (transparent) to `1.0` (opaque). Values outside this range are clipped to `0.0` or `1.0`.

Discussion

Quartz provides convenience functions for each of the device color spaces that allow you to set the fill or stroke color space and the fill or stroke color with one function call.

When you call this function, two things happen:

- Quartz sets the current fill color space to DeviceCMYK.
- Quartz sets the current fill color to the value specified by the `cyan`, `magenta`, `yellow`, `black`, and `alpha` parameters.

See also `CGContextSetCMYKStrokeColor` (page 104).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetCMYKStrokeColor

Sets the current stroke color to a value in the DeviceCMYK color space.

```
void CGContextSetCMYKStrokeColor (
   CGContextRef c,
   CGFloat cyan,
   CGFloat magenta,
   CGFloat yellow,
   CGFloat black,
   CGFloat alpha
);
```

Parameters

context

The graphics context for which to set the current stroke color.

cyan

The cyan intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from `0.0` (does not absorb the secondary color) to `1.0` (fully absorbs the secondary color).

magenta

The magenta intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from `0.0` (does not absorb the secondary color) to `1.0` (fully absorbs the secondary color).

yellow

The yellow intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from `0.0` (does not absorb the secondary color) to `1.0` (fully absorbs the secondary color).

black

The black intensity value for the color to set. The DeviceCMYK color space permits the specification of a value ranging from `0.0` (does not absorb the secondary color) to `1.0` (fully absorbs the secondary color).

alpha

A value that specifies the opacity level. Values can range from `0.0` (transparent) to `1.0` (opaque). Values outside this range are clipped to `0.0` or `1.0`.

Discussion

When you call this function, two things happen:

- Quartz sets the current stroke color space to DeviceCMYK.
- Quartz sets the current stroke color to the value specified by the `cyan`, `magenta`, `yellow`, `black`, and `alpha` parameters.

See also `CGContextSetCMYKFillColor` (page 102).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetFillColor

Sets the current fill color.

```
void CGContextSetFillColor (
   CGContextRef c,
   const CGFloat components[]
);
```

Parameters

context

The graphics context for which to set the current fill color.

components

An array of intensity values describing the color to set. The number of array elements must equal the number of components in the current fill color space, plus an additional component for the alpha value.

Discussion

The current fill color space must not be a pattern color space. For information on setting the fill color when using a pattern color space, see `CGContextSetFillPattern` (page 106). Note that the preferred API to use is now `CGContextSetFillColorWithColor` (page 106).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGContext.h`

CGContextSetFillColorSpace

Sets the fill color space in a graphics context.

```
void CGContextSetFillColorSpace (
   CGContextRef c,
   CGColorSpaceRef colorspace
);
```

Parameters

context

The graphics context for which to set the fill color space.

colorspace

The new fill color space. Quartz retains this object; upon return, you may safely release it.

Discussion

As a side effect of this function, Quartz assigns an appropriate initial value to the fill color, based on the specified color space. To change this value, call `CGContextSetFillColor` (page 105). Note that the preferred API to use is now `CGContextSetFillColorWithColor` (page 106).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
CGContext.h

CGContextSetFillColorWithColor

Sets the current fill color in a graphics context, using a Quartz color.

```
void CGContextSetFillColorWithColor (
   CGContextRef c,
   CGColorRef color
);
```

Parameters

context
The graphics context for which to set the fill color.

color
The new fill color.

Discussion
See also `CGContextSetFillColor` (page 105).

Availability
Available in Mac OS X version 10.3 and later.

Related Sample Code
CALayerEssentials

Declared In
CGContext.h

CGContextSetFillPattern

Sets the fill pattern in the specified graphics context.

```
void CGContextSetFillPattern (
   CGContextRef c,
   CGPatternRef pattern,
   const CGFloat components[]
);
```

Parameters

context
The graphics context to modify.

pattern
A fill pattern. Quartz retains this object; upon return, you may safely release it.

components

If the pattern is an uncolored (or a masking) pattern, pass an array of intensity values that specify the color to use when the pattern is painted. The number of array elements must equal the number of components in the base space of the fill pattern color space, plus an additional component for the alpha value.

If the pattern is a colored pattern, pass an alpha value.

Discussion

The current fill color space must be a pattern color space. Otherwise, the result of calling this function is undefined. If you want to set a fill color, not a pattern, then call the function `CGContextSetFillColorWithColor` (page 106).

Availability

Available in Mac OS X version 10.1 and later.

Declared In

`CGContext.h`

CGContextSetFlatness

Sets the accuracy of curved paths in a graphics context.

```
void CGContextSetFlatness (
   CGContextRef c,
   CGFloat flatness
);
```

Parameters

context

The graphics context to modify.

flatness

The largest permissible distance, measured in device pixels, between a point on the true curve and a point on the approximated curve.

Discussion

This function controls how accurately curved paths are rendered. Setting the flatness value to less than `1.0` renders highly accurate curves, but lengthens rendering times.

In most cases, you should not change the flatness value. Customizing the flatness value for the capabilities of a particular output device impairs the ability of your application to render to other devices.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetFont

Sets the platform font in a graphics context.

```
void CGContextSetFont (
   CGContextRef c,
   CGFontRef font
);
```

Parameters

context

The graphics context for which to set the font.

font

A Quartz font.

Discussion

For information about when to use this function, see `CGFontCreateWithPlatformFont` (page 177).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetFontSize

Sets the current font size.

```
void CGContextSetFontSize (
   CGContextRef c,
   CGFloat size
);
```

Parameters

context

A graphics context.

size

A font size, expressed in text space units.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetGrayFillColor

Sets the current fill color to a value in the DeviceGray color space.

```
void CGContextSetGrayFillColor (
   CGContextRef c,
   CGFloat gray,
   CGFloat alpha
);
```

Parameters

context

The graphics context for which to set the current fill color.

gray

A value that specifies the desired gray level. The DeviceGray color space permits the specification of a value ranging from `0.0` (absolute black) to `1.0` (absolute white). Values outside this range are clamped to `0.0` or `1.0`.

alpha

A value that specifies the opacity level. Values can range from `0.0` (transparent) to `1.0` (opaque). Values outside this range are clipped to `0.0` or `1.0`.

Discussion
When you call this function, two things happen:

- Quartz sets the current fill color space to DeviceGray.
- Quartz sets the current fill color to the value you specify in the `gray` and `alpha` parameters.

See also `CGContextSetGrayStrokeColor` (page 109).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextSetGrayStrokeColor

Sets the current stroke color to a value in the DeviceGray color space.

```
void CGContextSetGrayStrokeColor (
   CGContextRef c,
   CGFloat gray,
   CGFloat alpha
);
```

Parameters

context

The graphics context for which to set the current stroke color.

gray

A value that specifies the desired gray level. The DeviceGray color space permits the specification of a value ranging from `0.0` (absolute black) to `1.0` (absolute white). Values outside this range are clamped to `0.0` or `1.0`.

alpha

A value that specifies the opacity level. Values can range from `0.0` (transparent) to `1.0` (opaque). Values outside this range are clipped to `0.0` or `1.0`.

Discussion
When you call this function, two things happen:

- Quartz sets the current stroke color space to DeviceGray. The DeviceGray color space is a single-dimension space in which color values are specified solely by the intensity of a gray value (from absolute black to absolute white).
- Quartz sets the current stroke color to the value you specify in the `gray` and `alpha` parameters.

See also `CGContextSetGrayFillColor` (page 108).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextSetInterpolationQuality

Sets the level of interpolation quality for a graphics context.

```
void CGContextSetInterpolationQuality (
   CGContextRef c,
   CGInterpolationQuality quality
);
```

Parameters

context
The graphics context to modify.

quality
A `CGInterpolationQuality` constant that specifies the required level of interpolation quality. For possible values, see "`Interpolation Qualities`" (page 136).

Discussion
Interpolation quality is merely a hint to the context—not all contexts support all interpolation quality levels.

Availability
Available in Mac OS X version 10.1 and later.

See Also
`CGContextGetInterpolationQuality` (page 91)

Declared In
`CGContext.h`

CGContextSetLineCap

Sets the style for the endpoints of lines drawn in a graphics context.

```
void CGContextSetLineCap (
   CGContextRef c,
   CGLineCap cap
);
```

Parameters

context

The graphics context to modify.

cap

A line cap style constant—`kCGLineCapButt` (page 137) (the default), `kCGLineCapRound` (page 137), or `kCGLineCapSquare` (page 137). See "`Line Cap Styles`" (page 137).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextSetLineDash

Sets the pattern for dashed lines in a graphics context.

```
void CGContextSetLineDash (
   CGContextRef c,
   CGFloat phase,
   const CGFloat lengths[],
   size_t count
);
```

Parameters

context

The graphics context to modify.

phase

A value that specifies how far into the dash pattern the line starts, in units of the user space. For example, passing a value of `3` means the line is drawn with the dash pattern starting at three units from its beginning. Passing a value of `0` draws a line starting with the beginning of a dash pattern.

lengths

An array of values that specify the lengths of the painted segments and unpainted segments, respectively, of the dash pattern—or `NULL` for no dash pattern.

For example, passing an array with the values `[2,3]` sets a dash pattern that alternates between a 2-user-space-unit-long painted segment and a 3-user-space-unit-long unpainted segment. Passing the values `[1,3,4,2]` sets the pattern to a 1-unit painted segment, a 3-unit unpainted segment, a 4-unit painted segment, and a 2-unit unpainted segment.

count

If the `lengths` parameter specifies an array, pass the number of elements in the array. Otherwise, pass `0`.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
CGContext.h

CGContextSetLineJoin

Sets the style for the joins of connected lines in a graphics context.

```
void CGContextSetLineJoin (
   CGContextRef c,
   CGLineJoin join
);
```

Parameters

context

The graphics context to modify.

join

A line join value—kCGLineJoinMiter (page 138) (the default), kCGLineJoinRound (page 138), or kCGLineJoinBevel (page 138). See "Line Joins" (page 138).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
CGContext.h

CGContextSetLineWidth

Sets the line width for a graphics context.

```
void CGContextSetLineWidth (
   CGContextRef c,
   CGFloat width
);
```

Parameters

context

The graphics context to modify.

width

The new line width to use, in user space units. The value must be greater than 0.

Discussion
The default line width is 1 unit. When stroked, the line straddles the path, with half of the total width on either side.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextSetMiterLimit

Sets the miter limit for the joins of connected lines in a graphics context.

```
void CGContextSetMiterLimit (
   CGContextRef c,
   CGFloat limit
);
```

Parameters

context
The graphics context to modify.

limit
The miter limit to use.

Discussion
If the current line join style is set to `kCGLineJoinMiter` (see `CGContextSetLineJoin` (page 112)), Quartz uses the miter limit to determine whether the lines should be joined with a bevel instead of a miter. Quartz divides the length of the miter by the line width. If the result is greater than the miter limit, Quartz converts the style to a bevel.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextSetPatternPhase

Sets the pattern phase of a context.

```
void CGContextSetPatternPhase (
   CGContextRef c,
   CGSize phase
);
```

Parameters

context
The graphics context to modify.

phase
A pattern phase, specified in user space.

Discussion
The pattern phase is a translation that Quartz applies prior to drawing a pattern in the context. The pattern phase is part of the graphics state of a context, and the default pattern phase is `(0,0)`. Setting the pattern phase has the effect of temporarily changing the pattern matrix of any pattern you draw. For example, setting the context's pattern phase to `(2,3)` has the effect of moving the start of pattern cell tiling to the point `(2,3)` in default user space.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGContext.h`

CGContextSetRenderingIntent

Sets the rendering intent in the current graphics state.

```
void CGContextSetRenderingIntent (
   CGContextRef c,
   CGColorRenderingIntent intent
);
```

Parameters

context
The graphics context to modify.

intent
A rendering intent constant—`kCGRenderingIntentDefault` (page 53), `kCGRenderingIntentAbsoluteColorimetric` (page 53), `kCGRenderingIntentRelativeColorimetric` (page 53), `kCGRenderingIntentPerceptual` (page 53), or `kCGRenderingIntentSaturation` (page 53). For a discussion of these constants, see *CGColorSpace Reference*.

Discussion
The rendering intent specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context. If you do not explicitly set the rendering intent, Quartz uses perceptual rendering intent for drawing sampled images and relative colorimetric rendering intent for all other drawing.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextSetRGBFillColor

Sets the current fill color to a value in the DeviceRGB color space.

```
void CGContextSetRGBFillColor (
   CGContextRef c,
   CGFloat red,
   CGFloat green,
   CGFloat blue,
   CGFloat alpha
);
```

Parameters

context

The graphics context for which to set the current fill color.

red

The red intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from `0.0` (zero intensity) to `1.0` (full intensity).

green

The green intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from `0.0` (zero intensity) to `1.0` (full intensity).

blue

The blue intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from `0.0` (zero intensity) to `1.0` (full intensity).

alpha

A value that specifies the opacity level. Values can range from `0.0` (transparent) to `1.0` (opaque). Values outside this range are clipped to `0.0` or `1.0`.

Discussion

When you call this function, two things happen:

- Quartz sets the current fill color space to DeviceRGB.
- Quartz sets the current fill color to the value specified by the `red`, `green`, `blue`, and `alpha` parameters.

See also `CGContextSetRGBStrokeColor` (page 115).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CALayerEssentials
CarbonSketch
HID Calibrator
HID Config Save
HID Explorer

Declared In

`CGContext.h`

CGContextSetRGBStrokeColor

Sets the current stroke color to a value in the DeviceRGB color space.

```
void CGContextSetRGBStrokeColor (
   CGContextRef c,
   CGFloat red,
   CGFloat green,
   CGFloat blue,
   CGFloat alpha
);
```

Parameters

context

The graphics context for which to set the current stroke color.

red

The red intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from `0.0` (zero intensity) to `1.0` (full intensity).

green

The green intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from `0.0` (zero intensity) to `1.0` (full intensity).

blue

The blue intensity value for the color to set. The DeviceRGB color space permits the specification of a value ranging from `0.0` (zero intensity) to `1.0` (full intensity).

alpha

A value that specifies the opacity level. Values can range from `0.0` (transparent) to `1.0` (opaque). Values outside this range are clipped to `0.0` or `1.0`.

Discussion

When you call this function, two things happen:

- Quartz sets the current stroke color space to DeviceRGB.
- Quartz sets the current stroke color to the value specified by the `red`, `green`, `blue`, and `alpha` parameters.

See also `CGContextSetRGBFillColor` (page 114).

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch
HID Calibrator
HID Config Save
HID Explorer

Declared In

`CGContext.h`

CGContextSetShadow

Enables shadowing in a graphics context.

```
void CGContextSetShadow (
   CGContextRef context,
   CGSize offset,
   CGFloat blur
);
```

Parameters

context

A graphics context.

offset

Specifies a translation of the context's coordinate system, to establish an offset for the shadow (`{0,0}` specifies a light source immediately above the screen).

blur

A non-negative number specifying the amount of blur.

Discussion

Shadow parameters are part of the graphics state in a context. After shadowing is set, all objects drawn are shadowed using a black color with 1/3 alpha (i.e., RGBA = `{0, 0, 0, 1.0/3.0}`) in the DeviceRGB color space.

To turn off shadowing:

- Use the standard save/restore mechanism for the graphics state.
- Use `CGContextSetShadowWithColor` (page 117) to set the shadow color to a fully transparent color (or pass `NULL` as the color).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGContext.h`

CGContextSetShadowWithColor

Enables shadowing with color a graphics context.

```
void CGContextSetShadowWithColor (
   CGContextRef context,
   CGSize offset,
   CGFloat blur,
   CGColorRef color
);
```

Parameters

context

The graphics context to modify.

offset

Specifies a translation in base-space.

blur

A non-negative number specifying the amount of blur.

color

Specifies the color of the shadow, which may contain a non-opaque alpha value. If `NULL`, then shadowing is disabled.

Discussion
See also `CGContextSetShadow` (page 116).

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGContext.h`

CGContextSetShouldAntialias

Sets anti-aliasing on or off for a graphics context.

```
void CGContextSetShouldAntialias (
   CGContextRef c,
   bool shouldAntialias
);
```

Parameters

context

The graphics context to modify.

shouldAntialias

A Boolean value that specifies whether anti-aliasing should be turned on. Anti-aliasing is turned on by default when a window or bitmap context is created. It is turned off for other types of contexts.

Discussion
Anti-aliasing is a graphics state parameter.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextSetShouldSmoothFonts

Enables or disables font smoothing in a graphics context.

```
void CGContextSetShouldSmoothFonts (
   CGContextRef c,
   bool shouldSmoothFonts
);
```

Parameters

context

The graphics context to modify.

shouldSmoothFonts

A Boolean value that specifies whether to enable font smoothing.

Discussion
There are cases, such as rendering to a bitmap, when font smoothing is not appropriate and should be disabled. Note that some contexts (such as PostScript contexts) do not support font smoothing.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGContext.h`

CGContextSetStrokeColor

Sets the current stroke color.

```
void CGContextSetStrokeColor (
   CGContextRef c,
   const CGFloat components[]
);
```

Parameters

context

The graphics context for which to set the current stroke color.

components

An array of intensity values describing the color to set. The number of array elements must equal the number of components in the current stroke color space, plus an additional component for the alpha value.

Discussion
The current stroke color space must not be a pattern color space. For information on setting the stroke color when using a pattern color space, see `CGContextSetStrokePattern` (page 120). Note that the preferred API is now `CGContextSetStrokeColorWithColor` (page 120).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextSetStrokeColorSpace

Sets the stroke color space in a graphics context.

```
void CGContextSetStrokeColorSpace (
   CGContextRef c,
   CGColorSpaceRef colorspace
);
```

Parameters

context

The graphics context for the new stroke color space.

colorspace
The new stroke color space. Quartz retains this object; upon return, you may safely release it.

Discussion
As a side effect when you call this function, Quartz assigns an appropriate initial value to the stroke color, based on the color space you specify. To change this value, call `CGContextSetStrokeColor` (page 119). Note that the preferred API is now `CGContextSetStrokeColorWithColor` (page 120).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

CGContextSetStrokeColorWithColor

Sets the current stroke color in a context, using a Quartz color.

```
void CGContextSetStrokeColorWithColor (
   CGContextRef c,
   CGColorRef color
);
```

Parameters

context
The graphics context to modify.

color
The new stroke color.

Discussion
See also `CGContextSetStrokeColor` (page 119).

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGContext.h`

CGContextSetStrokePattern

Sets the stroke pattern in the specified graphics context.

```
void CGContextSetStrokePattern (
   CGContextRef c,
   CGPatternRef pattern,
   const CGFloat components[]
);
```

Parameters

context
The graphics context to modify.

pattern

A pattern for stroking. Quartz retains this object; upon return, you may safely release it.

components

If the specified pattern is an uncolored (or masking) pattern, pass an array of intensity values that specify the color to use when the pattern is painted. The number of array elements must equal the number of components in the base space of the stroke pattern color space, plus an additional component for the alpha value.

If the specified pattern is a colored pattern, pass an alpha value.

Discussion

The current stroke color space must be a pattern color space. Otherwise, the result of calling this function is undefined. If you want to set a stroke color, not a stroke pattern, then call the function `CGContextSetStrokeColorWithColor` (page 120).

Availability

Available in Mac OS X version 10.1 and later.

Declared In

`CGContext.h`

CGContextSetTextDrawingMode

Sets the current text drawing mode.

```
void CGContextSetTextDrawingMode (
   CGContextRef c,
   CGTextDrawingMode mode
);
```

Parameters

context

A graphics context.

mode

A text drawing mode (such as `kCGTextFill` (page 139) or `kCGTextStroke` (page 139)) that specifies how Quartz renders individual glyphs in a graphics context. See "`Text Drawing Modes`" (page 138) for a complete list.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextSetTextMatrix

Sets the current text matrix.

```
void CGContextSetTextMatrix (
   CGContextRef c,
   CGAffineTransform t
);
```

Parameters

context

A graphics context.

transform

The text matrix to set.

Discussion

The text matrix specifies the transform from text space to user space. To produce the final text rendering matrix that is used to actually draw the text on the page, Quartz concatenates the text matrix with the current transformation matrix and other parameters from the graphics state.

Note that the text matrix is *not* a part of the graphics state—saving or restoring the graphics state has no effect on the text matrix. The text matrix is an attribute of the graphics context, not of the current font.

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

HID Calibrator

Declared In

`CGContext.h`

CGContextSetTextPosition

Sets the location at which text is drawn.

```
void CGContextSetTextPosition (
   CGContextRef c,
   CGFloat x,
   CGFloat y
);
```

Parameters

context

A graphics context.

x

A value for the x-coordinate at which to draw the text, in user space coordinates.

y

A value for the y-coordinate at which to draw the text.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextShowGlyphs

Displays an array of glyphs at the current text position.

```
void CGContextShowGlyphs (
   CGContextRef c,
   const CGGlyph g[],
   size_t count
);
```

Parameters

context

The graphics context in which to display the glyphs.

glyphs

An array of glyphs to display.

count

The total number of glyphs passed in the `g` parameter.

Discussion

This function displays an array of glyphs at the current text position, a point specified by the current text matrix.

See also `CGContextShowGlyphsAtPoint` (page 123), `CGContextShowText` (page 125), `CGContextShowTextAtPoint` (page 126), and `CGContextShowGlyphsWithAdvances` (page 124).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGContext.h`

CGContextShowGlyphsAtPoint

Displays an array of glyphs at a position you specify.

```
void CGContextShowGlyphsAtPoint (
   CGContextRef c,
   CGFloat x,
   CGFloat y,
   const CGGlyph glyphs[],
   size_t count
);
```

Parameters

context

The graphics context in which to display the glyphs.

x

A value for the x-coordinate of the user space at which to display the glyphs.

y

A value for the y-coordinate of the user space at which to display the glyphs.

glyphs

An array of glyphs to display.

count
The total number of glyphs passed in the `glyphs` parameter.

Discussion
This function displays an array of glyphs at the specified position in the text space.

See also `CGContextShowText` (page 125), `CGContextShowGlyphs` (page 123), `CGContextShowGlyphs` (page 123), and `CGContextShowGlyphsWithAdvances` (page 124).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGContext.h`

CGContextShowGlyphsAtPositions

Draws glyphs at the provided position.

```
void CGContextShowGlyphsAtPositions(
    CGContextRef context,
    const CGGlyph glyphs[],
    const CGPoint positions[],
    size_t count
);
```

Parameters
context
The graphics context in which to display the glyphs.

glyphs
An array of Quartz glyphs.

positions
The positions for the glyphs. Each item in this array matches with the glyph at the corresponding index in the `glyphs` array. The position of each glyph is specified in text space, and, as a consequence, is transformed through the text matrix to user space.

count
The number of items in the `glyphs` array.

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

Declared In
`CGContext.h`

CGContextShowGlyphsWithAdvances

Draws an array of glyphs with varying offsets.

```
void CGContextShowGlyphsWithAdvances (
   CGContextRef c,
   const CGGlyph glyphs[],
   const CGSize advances[],
   size_t count
);
```

Parameters

context

The graphics context in which to display the glyphs.

glyphs

An array of Quartz glyphs.

advances

An array of offset values associated with each glyph in the array. Each value specifies the offset from the previous glyph's origin to the origin of the corresponding glyph. Offsets are specified in user space.

count

The number of glyphs in the specified array.

Discussion

This function draws an array of glyphs at the current point specified by the text matrix.

See also `CGContextShowText` (page 125), `CGContextShowGlyphs` (page 123), and `CGContextShowGlyphs` (page 123), and `CGContextShowGlyphsAtPoint` (page 123).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGContext.h`

CGContextShowText

Displays a character array at the current text position, a point specified by the current text matrix.

```
void CGContextShowText (
   CGContextRef c,
   const char *string,
   size_t length
);
```

Parameters

context

A graphics context.

string

An array of characters to draw.

length

The length of the array specified in the `bytes` parameter.

Discussion
Quartz uses font data provided by the system to map each byte of the array through the encoding vector of the current font to obtain the glyph to display. Note that the font must have been set using `CGContextSelectFont` (page 100). Don't use `CGContextShowTextAtPoint` in conjunction with `CGContextSetFont` (page 107).

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextShowTextAtPoint` (page 126)
`CGContextShowGlyphs` (page 123)
`CGContextShowGlyphsAtPoint` (page 123)
`CGContextShowGlyphsWithAdvances` (page 124)

Declared In
`CGContext.h`

CGContextShowTextAtPoint

Displays a character string at a position you specify.

```
void CGContextShowTextAtPoint (
   CGContextRef c,
   CGFloat x,
   CGFloat y,
   const char *string,
   size_t length
);
```

Parameters

context
A graphics context .

x
A value for the x-coordinate of the text space at which to display the text.

y
A value for the y-coordinate of the text space at which to display the text.

string
An array of characters to draw.

length
The length of the array specified in the `bytes` parameter.

Discussion
Quartz uses font data provided by the system to map each byte of the array through the encoding vector of the current font to obtain the glyph to display. Note that the font must have been set using `CGContextSelectFont` (page 100). Don't use `CGContextShowTextAtPoint` in conjunction with `CGContextSetFont` (page 107).

Availability
Available in Mac OS X version 10.0 and later.

See Also
CGContextShowText (page 125)
CGContextShowGlyphs (page 123)
CGContextShowGlyphsAtPoint (page 123)
CGContextShowGlyphsWithAdvances (page 124)

Related Sample Code
HID Calibrator

Declared In
CGContext.h

CGContextStrokeEllipseInRect

Strokes an ellipse that fits inside the specified rectangle.

```
void CGContextStrokeEllipseInRect (
   CGContextRef context,
   CGRect rect
);
```

Parameters

context
A graphics context.

rect
A rectangle that defines the area for the ellipse to fit in.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGContext.h

CGContextStrokeLineSegments

Strokes a sequence of line segments.

```
void CGContextStrokeLineSegments (
   CGContextRef c,
   const CGPoint points[],
   size_t count
);
```

Parameters

c
A graphics context.

points
An array of points, organized as pairs—the starting point of a line segment followed by the ending point of a line segment. For example, the first point in the array specifies the starting position of the first line, the second point specifies the ending position of the first line, the third point specifies the starting position of the second line, and so forth.

count

The number of points in the `points` array.

Discussion
This function is equivalent to the following code:

```
CGContextBeginPath (context);
for (k = 0; k < count; k += 2) {
    CGContextMoveToPoint(context, s[k].x, s[k].y);
    CGContextAddLineToPoint(context, s[k+1].x, s[k+1].y);
}
CGContextStrokePath(context);
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGContext.h`

CGContextStrokePath

Paints a line along the current path.

```
void CGContextStrokePath (
   CGContextRef c
);
```

Parameters

context

A graphics context.

Discussion
Quartz uses the line width and stroke color of the graphics state to paint the path. As a side effect when you call this function, Quartz clears the current path.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextDrawPath` (page 81)
`CGContextFillPath` (page 88)
`CGContextEOFillPath` (page 87)

Related Sample Code
CarbonSketch
HID Calibrator
HID Explorer

Declared In
`CGContext.h`

CGContextStrokeRect

Paints a rectangular path.

```
void CGContextStrokeRect (
   CGContextRef c,
   CGRect rect
);
```

Parameters

context

A graphics context .

rect

A rectangle, specified in user space coordinates.

Discussion
Quartz uses the line width and stroke color of the graphics state to paint the path.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextStrokeRectWithWidth` (page 129)

Related Sample Code
CarbonSketch
HID Calibrator
HID Config Save

Declared In
`CGContext.h`

CGContextStrokeRectWithWidth

Paints a rectangular path, using the specified line width.

```
void CGContextStrokeRectWithWidth (
   CGContextRef c,
   CGRect rect,
   CGFloat width
);
```

Parameters

context

A graphics context.

rect

A rectangle, in user space coordinates.

width

A value, in user space units, that is greater than zero. This value does not affect the line width values in the current graphics state.

Discussion
Aside from the line width value, Quartz uses the current attributes of the graphics state (such as stroke color) to paint the line. The line straddles the path, with half of the total width on either side. As a side effect when you call this function, Quartz clears the current path.

Availability
Available in Mac OS X version 10.0 and later.

See Also
CGContextStrokeRect (page 128)

Declared In
CGContext.h

CGContextSynchronize

Marks a window context for update.

```
void CGContextSynchronize (
   CGContextRef c
);
```

Parameters

context

The window context to synchronize. If you pass a PDF context or a bitmap context, this function does nothing.

Discussion
When you call this function, all drawing operations since the last update are flushed at the next regular opportunity. Under normal conditions, you do not need to call this function.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
CGContext.h

CGContextTranslateCTM

Changes the origin of the user coordinate system in a context.

```
void CGContextTranslateCTM (
   CGContextRef c,
   CGFloat tx,
   CGFloat ty
);
```

Parameters

context

A graphics context.

tx

The amount to displace the x-axis of the coordinate space, in units of the user space, of the specified context.

ty

The amount to displace the y-axis of the coordinate space, in units of the user space, of the specified context.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGContext.h`

Data Types

CGContextRef

An opaque type that represents a Quartz 2D drawing environment.

```
typedef struct CGContext * CGContextRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGContext.h`

Constants

Blend Modes

Compositing operations for images.

```
enum CGBlendMode {
    kCGBlendModeNormal,
    kCGBlendModeMultiply,
    kCGBlendModeScreen,
    kCGBlendModeOverlay,
    kCGBlendModeDarken,
    kCGBlendModeLighten,
    kCGBlendModeColorDodge,
    kCGBlendModeColorBurn,
    kCGBlendModeSoftLight,
    kCGBlendModeHardLight,
    kCGBlendModeDifference,
    kCGBlendModeExclusion,
    kCGBlendModeHue,
    kCGBlendModeSaturation,
    kCGBlendModeColor,
    kCGBlendModeLuminosity,
    kCGBlendModeClear,
    kCGBlendModeCopy,
    kCGBlendModeSourceIn,
    kCGBlendModeSourceOut,
    kCGBlendModeSourceAtop,
    kCGBlendModeDestinationOver,
    kCGBlendModeDestinationIn,
    kCGBlendModeDestinationOut,
    kCGBlendModeDestinationAtop,
    kCGBlendModeXOR,
    kCGBlendModePlusDarker,
    kCGBlendModePlusLighter
};
typedef enum CGBlendMode CGBlendMode;
```

Constants

`kCGBlendModeNormal`

Paints the source image samples over the background image samples.

Available in Mac OS X v10.4 and later.

Declared in `CGContext.h`.

`kCGBlendModeMultiply`

Multiplies the source image samples with the background image samples. This results in colors that are at least as dark as either of the two contributing sample colors.

Available in Mac OS X v10.4 and later.

Declared in `CGContext.h`.

`kCGBlendModeScreen`

Multiplies the inverse of the source image samples with the inverse of the background image samples. This results in colors that are at least as light as either of the two contributing sample colors.

Available in Mac OS X v10.4 and later.

Declared in `CGContext.h`.

`kCGBlendModeOverlay`
: Either multiplies or screens the source image samples with the background image samples, depending on the background color. The result is to overlay the existing image samples while preserving the highlights and shadows of the background. The background color mixes with the source image to reflect the lightness or darkness of the background.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeDarken`
: Creates the composite image samples by choosing the darker samples (either from the source image or the background). The result is that the background image samples are replaced by any source image samples that are darker. Otherwise, the background image samples are left unchanged.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeLighten`
: Creates the composite image samples by choosing the lighter samples (either from the source image or the background). The result is that the background image samples are replaced by any source image samples that are lighter. Otherwise, the background image samples are left unchanged.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeColorDodge`
: Brightens the background image samples to reflect the source image samples. Source image sample values that specify black do not produce a change.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeColorBurn`
: Darkens the background image samples to reflect the source image samples. Source image sample values that specify white do not produce a change.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeSoftLight`
: Either darkens or lightens colors, depending on the source image sample color. If the source image sample color is lighter than 50% gray, the background is lightened, similar to dodging. If the source image sample color is darker than 50% gray, the background is darkened, similar to burning. If the source image sample color is equal to 50% gray, the background is not changed. Image samples that are equal to pure black or pure white produce darker or lighter areas, but do not result in pure black or white. The overall effect is similar to what you'd achieve by shining a diffuse spotlight on the source image. Use this to add highlights to a scene.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeHardLight`
: Either multiplies or screens colors, depending on the source image sample color. If the source image sample color is lighter than 50% gray, the background is lightened, similar to screening. If the source image sample color is darker than 50% gray, the background is darkened, similar to multiplying. If the source image sample color is equal to 50% gray, the source image is not changed. Image samples that are equal to pure black or pure white result in pure black or white. The overall effect is similar to what you'd achieve by shining a harsh spotlight on the source image. Use this to add highlights to a scene.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeDifference`
: Subtracts either the source image sample color from the background image sample color, or the reverse, depending on which sample has the greater brightness value. Source image sample values that are black produce no change; white inverts the background color values.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeExclusion`
: Produces an effect similar to that produced by `kCGBlendModeDifference`, but with lower contrast. Source image sample values that are black don't produce a change; white inverts the background color values.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeHue`
: Uses the luminance and saturation values of the background with the hue of the source image.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeSaturation`
: Uses the luminance and hue values of the background with the saturation of the source image. Areas of the background that have no saturation (that is, pure gray areas) don't produce a change.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeColor`
: Uses the luminance values of the background with the hue and saturation values of the source image. This mode preserves the gray levels in the image. You can use this mode to color monochrome images or to tint color images.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeLuminosity`
: Uses the hue and saturation of the background with the luminance of the source image. This mode creates an effect that is inverse to the effect created by `kCGBlendModeColor`.

  Available in Mac OS X v10.4 and later.

  Declared in `CGContext.h`.

`kCGBlendModeClear`
`R = 0`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeCopy`
`R = S`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeSourceIn`
`R = S*Da`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeSourceOut`
`R = S*(1 - Da)`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeSourceAtop`
`R = S*Da + D*(1 - Sa)`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeDestinationOver`
`R = S*(1 - Da) + D`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeDestinationIn`
`R = D*Sa`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeDestinationOut`
`R = D*(1 - Sa)`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeDestinationAtop`
`R = S*(1 - Da) + D*Sa`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModeXOR`
`R = S*(1 - Da) + D*(1 - Sa)`. This XOR mode is only nominally related to the classical bitmap XOR operation, which is not supported by Quartz 2D.

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModePlusDarker`
`R = MAX(0, (1 - D) + (1 - S))`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

`kCGBlendModePlusLighter`
`R = MIN(1, S + D)`

Available in Mac OS X v10.5 and later.

Declared in `CGContext.h`.

Discussion
The blend mode constants introduced in Mac OS X v10.5 represent the Porter-Duff blend modes. The symbols in the equations for these blend modes are:

- R is the premultiplied result
- S is the source color, and includes alpha
- D is the destination color, and includes alpha
- Ra, Sa, and Da are the alpha components of R, S, and D

You can find more information on blend modes, including examples of images produced using them, and many mathematical descriptions of the modes, in *PDF Reference, Fourth Edition*, Version 1.5, Adobe Systems, Inc. If you are a former QuickDraw developer, it may be helpful for you to think of blend modes as an alternative to transfer modes

For examples of using blend modes see "Setting Blend Modes" and "Using Blend Modes With Images" in *Quartz 2D Programming Guide*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGContext.h`

Interpolation Qualities

Levels of interpolation quality for rendering an image.

```
enum CGInterpolationQuality {
    kCGInterpolationDefault,
    kCGInterpolationNone,
    kCGInterpolationLow,
    kCGInterpolationHigh
};
typedef enum CGInterpolationQuality CGInterpolationQuality;
```

Constants
`kCGInterpolationDefault`

The default level of quality.

Available in Mac OS X v10.1 and later.

Declared in `CGContext.h`.

`kCGInterpolationNone`
No interpolation.

Available in Mac OS X v10.1 and later.

Declared in `CGContext.h`.

`kCGInterpolationLow`
A low level of interpolation quality. This setting may speed up image rendering.

Available in Mac OS X v10.1 and later.

Declared in `CGContext.h`.

`kCGInterpolationHigh`
A high level of interpolation quality. This setting may slow down image rendering.

Available in Mac OS X v10.1 and later.

Declared in `CGContext.h`.

Discussion
You use the function `CGContextSetInterpolationQuality` (page 110) to set the interpolation quality in a graphics context.

Declared In
`CGContext.h`

Line Cap Styles

Styles for rendering the endpoint of a stroked line.

```
enum CGLineCap {
    kCGLineCapButt,
    kCGLineCapRound,
    kCGLineCapSquare
};
typedef enum CGLineCap CGLineCap;
```

Constants
`kCGLineCapButt`
A line with a squared-off end. Quartz draws the line to extend only to the exact endpoint of the path. This is the default.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGLineCapRound`
A line with a rounded end. Quartz draws the line to extend beyond the endpoint of the path. The line ends with a semicircular arc with a radius of 1/2 the line's width, centered on the endpoint.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGLineCapSquare`
A line with a squared-off end. Quartz extends the line beyond the endpoint of the path for a distance equal to half the line width.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

Discussion
A line cap specifies the method used by `CGContextStrokePath` (page 128) to draw the endpoint of the line. To change the line cap style in a graphics context, you use the function `CGContextSetLineCap` (page 110).

Declared In
`CGContext.h`

Line Joins

Junction types for stroked lines.

```
enum CGLineJoin {
    kCGLineJoinMiter,
    kCGLineJoinRound,
    kCGLineJoinBevel
};
typedef enum CGLineJoin CGLineJoin;
```

Constants

`kCGLineJoinMiter`
A join with a sharp (angled) corner. Quartz draws the outer sides of the lines beyond the endpoint of the path, until they meet. If the length of the miter divided by the line width is greater than the miter limit, a bevel join is used instead. This is the default. To set the miter limit, see `CGContextSetMiterLimit` (page 113)

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGLineJoinRound`
A join with a rounded end. Quartz draws the line to extend beyond the endpoint of the path. The line ends with a semicircular arc with a radius of 1/2 the line's width, centered on the endpoint.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGLineJoinBevel`
A join with a squared-off end. Quartz draws the line to extend beyond the endpoint of the path, for a distance of 1/2 the line's width.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

Discussion
A line join specifies how `CGContextStrokePath` (page 128) draws the junction between connected line segments. To set the line join style in a graphics context, you use the function `CGContextSetLineJoin` (page 112).

Declared In
`CGContext.h`

Text Drawing Modes

Modes for rendering text.

```
enum CGTextDrawingMode {
    kCGTextFill,
    kCGTextStroke,
    kCGTextFillStroke,
    kCGTextInvisible,
    kCGTextFillClip,
    kCGTextStrokeClip,
    kCGTextFillStrokeClip,
    kCGTextClip
};
typedef enum CGTextDrawingMode CGTextDrawingMode;
```

Constants

`kCGTextFill`
Perform a fill operation on the text.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGTextStroke`
Perform a stroke operation on the text.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGTextFillStroke`
Perform fill, then stroke operations on the text.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGTextInvisible`
Do not draw the text, but do update the text position.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGTextFillClip`
Perform a fill operation, then intersect the text with the current clipping path.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGTextStrokeClip`
Perform a stroke operation, then intersect the text with the current clipping path.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGTextFillStrokeClip`
Perform fill then stroke operations, then intersect the text with the current clipping path.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGTextClip`
Specifies to intersect the text with the current clipping path. This mode does not paint the text.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

Discussion
You provide a text drawing mode constant to the function `CGContextSetTextDrawingMode` (page 121) to set the current text drawing mode for a graphics context. Text drawing modes determine how Quartz renders individual glyphs onscreen. For example, you can set a text drawing mode to draw text filled in or outlined (stroked) or both. You can also create special effects with the text clipping drawing modes, such as clipping an image to a glyph shape.

Declared In
`CGContext.h`

Text Encodings

Text encodings for fonts.

```
enum CGTextEncoding {
    kCGEncodingFontSpecific,
    kCGEncodingMacRoman
};
typedef enum CGTextEncoding CGTextEncoding;
```

Constants

`kCGEncodingFontSpecific`
The built-in encoding of the font.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

`kCGEncodingMacRoman`
The MacRoman encoding. MacRoman is an ASCII variant originally created for use in the Mac OS, in which characters 127 and lower are ASCII, and characters 128 and higher are non-English characters and symbols.

Available in Mac OS X v10.0 and later.

Declared in `CGContext.h`.

Discussion
For more information on setting the font in a graphics context, see `CGContextSelectFont` (page 100).

Declared In
`CGContext.h`

CGDataConsumer Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGDataConsumer.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGDataConsumerRef` opaque type abstracts the data-writing task and eliminates the need for applications to manage data through a raw memory buffer. You can use data consumer objects to write image or PDF data and all, except for `CGDataConsumerCreateWithCFData` (page 142), are available in Mac OS X v10.0 or later.

If your application runs in Mac OS X v10.4 or later, you should use CGImageDestination objects rather than data consumers. See *CGImageDestination Reference*.

Functions by Task

Creating Data Consumers

`CGDataConsumerCreate` (page 142)
Creates a data consumer that uses callback functions to write data.

`CGDataConsumerCreateWithURL` (page 143)
Creates a data consumer that writes data to a location specified by a URL.

`CGDataConsumerCreateWithCFData` (page 142)
Creates a data consumer that writes to a CFData object.

Getting the CFType ID

`CGDataConsumerGetTypeID` (page 143)
Returns the Core Foundation type identifier for Quartz data consumers.

Retaining and Releasing Data Consumers

`CGDataConsumerRelease` (page 144)
Decrements the retain count of a data consumer.

`CGDataConsumerRetain` (page 144)
Increments the retain count of a data consumer.

Functions

CGDataConsumerCreate

Creates a data consumer that uses callback functions to write data.

```
CGDataConsumerRef CGDataConsumerCreate (
   void *info,
   const CGDataConsumerCallbacks *callbacks
);
```

Parameters

info
A pointer to data of any type or `NULL`. When Quartz calls the functions specified in the `callbacks` parameter, it passes this pointer as the `info` parameter.

callbacks
A pointer to a `CGDataConsumerCallbacks` structure that specifies the callback functions you implement to copy data sent to the consumer and to handle the consumer's basic memory management. For a complete description, see `CGDataConsumerCallbacks` (page 146).

Return Value
A new data consumer object. You are responsible for releasing this object using `CGDataConsumerRelease` (page 144).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGDataConsumer.h`

CGDataConsumerCreateWithCFData

Creates a data consumer that writes to a CFData object.

```
CGDataConsumerRef CGDataConsumerCreateWithCFData (
   CFMutableDataRef data
);
```

Parameters

data

The CFData object to write to.

Return Value

A new data consumer object. You are responsible for releasing this object using `CGDataConsumerRelease` (page 144).

Discussion

You can use this function when you need to represent Quartz data as a CFData type. For example, you might create a CFData object that you then copy to the pasteboard.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGDataConsumer.h`

CGDataConsumerCreateWithURL

Creates a data consumer that writes data to a location specified by a URL.

```
CGDataConsumerRef CGDataConsumerCreateWithURL (
   CFURLRef url
);
```

Parameters

url

A CFURL object that specifies the data destination.

Return Value

A new data consumer object. You are responsible for releasing this object using `CGDataConsumerRelease` (page 144).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGDataConsumer.h`

CGDataConsumerGetTypeID

Returns the Core Foundation type identifier for Quartz data consumers.

```
CFTypeID CGDataConsumerGetTypeID (
   void
);
```

Return Value

The Core Foundation identifier for the opaque type `CGDataConsumerRef` (page 147).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGDataConsumer.h`

CGDataConsumerRelease

Decrements the retain count of a data consumer.

```
void CGDataConsumerRelease (
   CGDataConsumerRef consumer
);
```

Parameters

consumer
The data consumer to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the `consumer` parameter is `NULL`.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGDataConsumer.h`

CGDataConsumerRetain

Increments the retain count of a data consumer.

```
CGDataConsumerRef CGDataConsumerRetain (
   CGDataConsumerRef consumer
);
```

Parameters

consumer
The data consumer to retain.

Return Value
The same data consumer you passed in as the `consumer` parameter.

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `consumer` parameter is `NULL`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGDataConsumer.h`

Callbacks

CGDataConsumerPutBytesCallback

Copies data from a Quartz-supplied buffer into a data consumer.

```
size_t (*CGDataConsumerPutBytesCallback) (
    void *info,
    const void *buffer,
    size_t count
);
```

If you name your function `MyConsumerPutBytes`, you would declare it like this:

```
size_t MyConsumerPutBytes (
    void *info,
    const void *buffer,
    size_t count
);
```

Parameters

info
: A generic pointer to private data shared among your callback functions. This is the pointer supplied to `CGDataConsumerCreate` (page 142).

buffer
: The Quartz-supplied buffer from which you copy the specified number of bytes.

count
: The number of bytes to copy.

Return Value
The number of bytes copied. If no more data can be written to the consumer, you should return `0`.

Discussion
When Quartz is ready to send data to the consumer, your function is called. It should copy the specified number of bytes from `buffer` into some resource under your control—for example, a file.

For information on how to associate your callback function with a data consumer, see `CGDataConsumerCreate` (page 142) and `CGDataConsumerCallbacks` (page 146).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGDataConsumer.h`

CGDataConsumerReleaseInfoCallback

Releases any private data or resources associated with the data consumer.

```
void (*CGDataConsumerReleaseInfoCallback) (
    void *info
);
```

If you name your function `MyConsumerReleaseInfo`, you would declare it like this:

```
void MyConsumerReleaseInfo (
    void *info
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataConsumerCreate` (page 142).

Discussion

When Quartz frees a data consumer that has an associated release function, the release function is called.

For information on how to associate your callback function with a data consumer, see `CGDataConsumerCreate` (page 142) and `CGDataConsumerCallbacks` (page 146).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGDataConsumer.h`

Data Types

CGDataConsumerCallbacks

A structure that contains pointers to callback functions that manage the copying of data for a data consumer.

```
struct CGDataConsumerCallbacks {
 CGDataConsumerPutBytesCallback putBytes;
 CGDataConsumerReleaseInfoCallback releaseConsumer;
};
typedef struct CGDataConsumerCallbacks CGDataConsumerCallbacks;
```

Fields

`putBytes`

A pointer to a function that copies data to the data consumer. For more information, see `CGDataConsumerPutBytesCallback` (page 145).

`releaseConsumer`

A pointer to a function that handles clean-up for the data consumer, or `NULL`. For more information, see `CGDataConsumerReleaseInfoCallback` (page 146)

Discussion
The functions specified by the `CGDataConsumerCallbacks` structure are responsible for copying data that Quartz sends to your consumer and for handling the consumer's basic memory management. You supply a `CGDataConsumerCallbacks` structure to the function `CGDataConsumerCreate` (page 142) to create a data consumer.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDataConsumer.h`

CGDataConsumerRef

An opaque type that handles the storage of data supplied by Quartz functions.

```
typedef struct CGDataConsumer *CGDataConsumerRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDataConsumer.h`

CGDataProvider Reference

Derived From:	*CFType Reference*
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGDataProvider.h

Overview

The CGDataProvider header file declares a data type that supplies Quartz functions with data. Data provider objects abstract the data-access task and eliminate the need for applications to manage data through a raw memory buffer.

For information on how to use CGDataProvider functions, see *Quartz 2D Programming Guide* Programming Guide.

See also *CGDataConsumer Reference*.

Functions

CGDataProviderCopyData

Returns a copy of the provider's data.

```
CFDataRef CGDataProviderCopyData(
   CGDataProviderRef provider
);
```

Parameters

provider

The data provider whose data you want to copy.

Return Value

A new data object containing a copy of the provider's data. You are responsible for releasing this object.

Availability

Available in Mac OS X v10.5 and later.

Declared In

CGDataProvider.h

CGDataProviderCreate

Creates a Quartz sequential-access data provider. (Deprecated in Mac OS X v10.5.)

```
CGDataProviderRef CGDataProviderCreate (
   void *info,
   const CGDataProviderCallbacks *callbacks
);
```

Parameters

info

A pointer to data of any type or `NULL`. When Quartz calls the functions specified in the `callbacks` parameter, it sends each of the functions this data.

callbacks

A pointer to a `CGDataProviderCallbacks` structure that specifies the callback functions you implement to handle the data provider's basic memory management. For a complete description, see `CGDataProviderCallbacks` (page 164).

Return Value

A new data provider. You are responsible for releasing this object using `CGDataProviderRelease` (page 155).

Discussion

You use this function to create a sequential-access data provider that uses callback functions to read data from your program in a stream.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Declared In

`CGDataProvider.h`

CGDataProviderCreateDirect

Creates a Quartz direct-access data provider.

```
CGDataProviderRef CGDataProviderCreateDirect (
   void *info,
   off_t size,
   const CGDataProviderDirectCallbacks *callbacks
);
```

Parameters

info

A pointer to data of any type or `NULL`. When Quartz calls the functions specified in the `callbacks` parameter, it sends each of the functions this pointer.

size

The number of bytes of data to provide.

callbacks

A pointer to a `CGDataProviderDirectCallbacks` structure that specifies the callback functions you implement to handle the data provider's basic memory management.

Return Value
A new data provider. You are responsible for releasing this object using `CGDataProviderRelease` (page 155).

Discussion
You use this function to create a direct-access data provider that uses callback functions to read data from your program in a single block.

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

Declared In
`CGDataProvider.h`

CGDataProviderCreateDirectAccess

Creates a Quartz direct-access data provider. (Deprecated in Mac OS X v10.5.)

```
CGDataProviderRef CGDataProviderCreateDirectAccess (
   void *info,
   size_t size,
   const CGDataProviderDirectAccessCallbacks *callbacks
);
```

Parameters

info

A pointer to data of any type or `NULL`. When Quartz calls the functions specified in the `callbacks` parameter, it sends each of the functions this pointer.

size

A value that specifies the number of bytes that the data provider contains.

callbacks

A pointer to a `CGDataProviderDirectAccessCallbacks` structure that specifies the callback functions you implement to handle the data provider's basic memory management. For a complete description, see `CGDataProviderDirectAccessCallbacks` (page 165).

Return Value
A new data provider. You are responsible for releasing this object using `CGDataProviderRelease` (page 155).

Discussion
You use this function to create a direct-access data provider that uses callback functions to read data from your program in a single block.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

Declared In
`CGDataProvider.h`

CGDataProviderCreateSequential

Creates a Quartz sequential-access data provider.

```
CGDataProviderRef CGDataProviderCreateSequential (
   void *info,
   const CGDataProviderSequentialCallbacks *callbacks
);
```

Parameters

info

A pointer to data of any type or `NULL`. When Quartz calls the functions specified in the `callbacks` parameter, it sends each of the functions this pointer.

callbacks

A pointer to a `CGDataProviderSequentialCallbacks` structure that specifies the callback functions you implement to handle the data provider's basic memory management.

Return Value

A new data provider. You are responsible for releasing this object using `CGDataProviderRelease` (page 155).

Discussion

You use this function to create a sequential-access data provider that uses callback functions to read data from your program in a single block.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CGDataProvider.h`

CGDataProviderCreateWithCFData

Creates a Quartz data provider that reads from a CFData object.

```
CGDataProviderRef CGDataProviderCreateWithCFData (
   CFDataRef data
);
```

Parameters

data

The CFData object to read from.

Return Value

A new data provider. You are responsible for releasing this object using `CGDataProviderRelease` (page 155).

Discussion

You can use this function when you need to represent Quartz data as a CFData type. For example, you might create a CFData object when reading data from the pasteboard.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGDataProvider.h`

CGDataProviderCreateWithData

Creates a Quartz direct-access data provider that uses data your program supplies.

```
CGDataProviderRef CGDataProviderCreateWithData (
   void *info,
   const void *data,
   size_t size,
   CGDataProviderReleaseDataCallback releaseData
);
```

Parameters

info

A pointer to data of any type, or `NULL`. When Quartz calls the function specified in the `releaseData` parameter, Quartz sends it this pointer as its first argument.

data

A pointer to the array of data that the provider contains.

size

A value that specifies the number of bytes that the data provider contains.

releaseData

A pointer to a release callback for the data provider, or `NULL`. Your release function is called when Quartz frees the data provider. For more information, see `CGDataProviderReleaseDataCallback` (page 160).

Return Value

A new data provider. You are responsible for releasing this object using `CGDataProviderRelease` (page 155).

Discussion

You use this function to create a direct-access data provider that uses callback functions to read data from your program an entire block at one time.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGDataProvider.h`

CGDataProviderCreateWithFilename

Creates a Quartz direct-access data provider that uses a file to supply data.

```
CGDataProviderRef CGDataProviderCreateWithFilename(
   const char *filename
);
```

Parameters

filename

The full or relative pathname to use for the data provider. When you supply Quartz data via the provider, it reads the data from the specified file.

Return Value
A new data provider or `NULL` if the file could not be opened. You are responsible for releasing this object using `CGDataProviderRelease` (page 155).

Discussion
You use this function to create a direct-access data provider that supplies data from a file. When you supply Quartz with a direct-access data provider, Quartz obtains data from your program in a single block.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDataProvider.h`

CGDataProviderCreateWithURL

Creates a Quartz direct-access data provider that uses a URL to supply data.

```
CGDataProviderRef CGDataProviderCreateWithURL (
   CFURLRef url
);
```

Parameters

url

A CFURL object to use for the data provider. When you supply Quartz data via the provider, it reads the data from the URL address.

Return Value
A new data provider or `NULL` if the data from the URL could not be accessed. You are responsible for releasing this object using `CGDataProviderRelease` (page 155).

Discussion
You use this function to create a direct-access data provider that supplies data from a URL. When you supply Quartz with a direct-access data provider, Quartz obtains data from your program in a single entire block.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDataProvider.h`

CGDataProviderGetTypeID

Returns the Core Foundation type identifier for Quartz data providers.

```
CFTypeID CGDataProviderGetTypeID (
   void
);
```

Return Value
The identifier for the opaque type `CGDataProviderRef` (page 164).

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGDataProvider.h

CGDataProviderRelease

Decrements the retain count of a data provider.

```
void CGDataProviderRelease (
   CGDataProviderRef provider
);
```

Parameters

provider
The data provider to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the `provider` parameter is `NULL`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDataProvider.h

CGDataProviderRetain

Increments the retain count of a data provider.

```
CGDataProviderRef CGDataProviderRetain (
   CGDataProviderRef provider
);
```

Parameters

provider
The data provider to retain.

Return Value
The same data provider you passed in as the `provider` parameter.

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `provider` parameter is `NULL`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDataProvider.h

Callbacks by Task

Sequential-Access Data Provider Callbacks

`CGDataProviderGetBytesCallback` (page 159)
A callback function that copies from a provider data stream into a Quartz-supplied buffer.

`CGDataProviderReleaseInfoCallback` (page 161)
A callback function that releases any private data or resources associated with the data provider.

`CGDataProviderRewindCallback` (page 162)
A callback function that moves the current position in the data stream back to the beginning.

`CGDataProviderSkipBytesCallback` (page 162)
A callback function that advances the current position in the data stream supplied by the provider.

`CGDataProviderSkipForwardCallback` (page 163)
A callback function that advances the current position in the data stream supplied by the provider.

Direct-Access Data Provider Callbacks

`CGDataProviderGetBytePointerCallback` (page 156)
A callback function that returns a generic pointer to the provider data.

`CGDataProviderGetBytesAtOffsetCallback` (page 157)
A callback function that copies data from the provider into a Quartz buffer.

`CGDataProviderReleaseBytePointerCallback` (page 160)
A callback function that releases the pointer Quartz obtained by calling `CGDataProviderGetBytePointerCallback` (page 156).

`CGDataProviderReleaseDataCallback` (page 160)
A callback function that releases data you supply to the function `CGDataProviderCreateWithData` (page 153).

`CGDataProviderGetBytesAtPositionCallback` (page 158)
A callback function that copies data from the provider into a Quartz buffer.

Callbacks

CGDataProviderGetBytePointerCallback

A callback function that returns a generic pointer to the provider data.

```
const void * (*CGDataProviderGetBytePointerCallback) (
   void *info
);
```

If you name your function `MyProviderGetBytePointer`, you would declare it like this:

```
void *MyProviderGetBytePointer (
```

```
    void *info
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreateDirectAccess` (page 151).

Return Value

A generic pointer to your provider data. By suppling this pointer, you are giving Quartz read-only access to both the pointer and the underlying provider data. You must not move or modify the provider data until Quartz calls your `CGDataProviderReleaseBytePointerCallback` (page 160) function.

Discussion

When Quartz needs direct access to your provider data, this function is called.

For information on how to associate your function with a direct-access data provider, see `CGDataProviderCreateDirectAccess` (page 151) and `CGDataProviderDirectAccessCallbacks` (page 165).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGDataProvider.h`

CGDataProviderGetBytesAtOffsetCallback

A callback function that copies data from the provider into a Quartz buffer.

```
typedef size_t (*CGDataProviderGetBytesAtOffsetCallback) (
    void *info,
    void *buffer,
    size_t offset,
    size_t count
);
```

If you name your function `MyProviderGetBytesWithOffset`, you would declare it like this:

```
size_t MyProviderGetBytesWithOffset (
    void *info,
    void *buffer,
    size_t offset,
    size_t count
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreateDirectAccess` (page 151).

buffer

The Quartz-supplied buffer into which you copy the specified number of bytes.

offset
Specifies the relative location in the data provider at which to begin copying data.

count
The number of bytes to copy.

Return Value
The number of bytes copied. If no more data can be written to the buffer, you should return 0.

Discussion
When Quartz is ready to receive data from the provider, your function is called.

For information on how to associate your function with a direct-access data provider, see `CGDataProviderCreateDirectAccess` (page 151) and `CGDataProviderDirectAccessCallbacks` (page 165).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGDataProvider.h`

CGDataProviderGetBytesAtPositionCallback

A callback function that copies data from the provider into a Quartz buffer.

```
typedef size_t (*CGDataProviderGetBytesAtPositionCallback) (
    void *info,
    void *buffer,
    off_t position,
    size_t count
);
```

If you name your function `MyProviderGetBytesAtPosition`, you would declare it like this:

```
size_t MyProviderGetBytesAtPosition (
    void *info,
    void *buffer,
    off_t position,
    size_t count
);
```

Parameters

info
A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreateDirect` (page 150).

buffer
The Quartz-supplied buffer into which you copy the specified number of bytes.

position
Specifies the relative location in the data provider at which to begin copying data.

count
The number of bytes to copy.

Return Value
The number of bytes copied. If no more data can be written to the buffer, you should return 0.

Discussion
When Quartz is ready to receive data from the provider, your function is called.

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

Declared In
`CGDataProvider.h`

CGDataProviderGetBytesCallback

A callback function that copies from a provider data stream into a Quartz-supplied buffer.

```
size_t (*CGDataProviderGetBytesCallback) (
    void *info,
    void *buffer,
    size_t count
);
```

If you name your function `MyProviderGetBytes`, you would declare it like this:

```
size_t MyProviderGetBytes (
    void *info,
    void *buffer,
    size_t count
);
```

Parameters

info
A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate` (page 150).

buffer
The Quartz-supplied buffer into which you copy the specified number of bytes.

count
The number of bytes to copy.

Return Value
The number of bytes copied. If no more data can be written to the buffer, you should return `0`.

Discussion
When Quartz is ready to receive data from the provider data stream, your function is called. It should copy the specified number of bytes into `buffer`.

For information on how to associate your callback function with a data provider, see `CGDataProviderCreate` (page 150) and `CGDataProviderCallbacks` (page 164).

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGDataProvider.h

CGDataProviderReleaseBytePointerCallback

A callback function that releases the pointer Quartz obtained by calling CGDataProviderGetBytePointerCallback (page 156).

```
typedef void (*CGDataProviderReleaseBytePointerCallback) (
    void *info,
    const void *pointer
);
```

If you name your function MyProviderReleaseBytePointer, you would declare it like this:

```
void MyProviderReleaseBytePointer (
    void *info,
    const void *pointer
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to CGDataProviderCreateDirectAccess (page 151).

pointer

A pointer to your provider data. This is the same pointer you returned in CGDataProviderGetBytePointerCallback (page 156).

Discussion
When Quartz no longer needs direct access to your provider data, your function is called. You may safely modify, move, or release your provider data at this time.

For information on how to associate your function with a direct-access data provider, see CGDataProviderCreateDirectAccess (page 151) and CGDataProviderDirectAccessCallbacks (page 165).

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGDataProvider.h

CGDataProviderReleaseDataCallback

A callback function that releases data you supply to the function CGDataProviderCreateWithData (page 153).

```
typedef void (*CGDataProviderReleaseDataCallback) (
    void *info,
    const void *data
    size_t size
);
```

If you name your function `MyProviderReleaseData`, you would declare it like this:

```
void MyProviderReleaseData (
    void *info,
    const void *data
    size_t size
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreateWithData` (page 153).

data

A pointer to your provider data.

size

The size of the data.

Discussion

When Quartz no longer needs direct access to your provider data, your function is called. You may safely modify, move, or release your provider data at this time.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CGDataProvider.h`

CGDataProviderReleaseInfoCallback

A callback function that releases any private data or resources associated with the data provider.

```
void (*CGDataProviderReleaseInfoCallback) (
    void *info
);
```

If you name your function `MyProviderReleaseInfo`, you would declare it like this:

```
void MyProviderReleaseInfo (
    void *info
);
```

Parameters

info

A generic pointer to private information shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate` (page 150).

Discussion

When Quartz frees a data provider that has an associated release function, the release function is called.

For information on how to associate your callback function with a data provider, see `CGDataProviderCreate` (page 150) and `CGDataProviderCallbacks` (page 164).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGDataProvider.h`

CGDataProviderRewindCallback

A callback function that moves the current position in the data stream back to the beginning.

```
void (*CGDataProviderRewindCallback) (
    void *info
);
```

If you name your function `MyProviderRewind`, you would declare it like this:

```
void MyProviderRewind (
    void *info
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate` (page 150).

Discussion

When Quartz needs to read from the beginning of the provider's data stream, your function is called.

For information on how to associate your callback function with a data provider, see `CGDataProviderCreate` (page 150) and `CGDataProviderCallbacks` (page 164).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGDataProvider.h`

CGDataProviderSkipBytesCallback

A callback function that advances the current position in the data stream supplied by the provider.

```
void (*CGDataProviderSkipBytesCallback) (
    void *info,
    size_t count
);
```

If you name your function `MyProviderSkipBytes`, you would declare it like this:

```
void MyProviderSkipBytes (
    void *info,
    size_t count
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate` (page 150).

count

The number of bytes to skip.

Discussion

When Quartz needs to advance forward in the provider's data stream, your function is called.

For information on how to associate your callback function with a data provider, see `CGDataProviderCreate` (page 150) and `CGDataProviderCallbacks` (page 164).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGDataProvider.h`

CGDataProviderSkipForwardCallback

A callback function that advances the current position in the data stream supplied by the provider.

```
off_t (*CGDataProviderSkipForwardCallback) (
    void *info,
    off_t count
);
```

If you name your function `MyProviderSkipForwardBytes`, you would declare it like this:

```
off_t MyProviderSkipForwardBytes (
    void *info,
    off_t count
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGDataProviderCreate` (page 150).

count
The number of bytes to skip.

Return Value
The number of bytes that were actually skipped.

Discussion
When Quartz needs to advance forward in the provider's data stream, your function is called.

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

Declared In
`CGDataProvider.h`

Data Types

CGDataProviderRef

Defines an opaque type that supplies Quartz with data.

```
typedef struct CGDataProvider *CGDataProviderRef;
```

Discussion
Some Quartz routines supply blocks of data to your program. Rather than reading through a raw memory buffer, data provider objects of type `CGDataProviderRef` allow you to supply Quartz functions with data.

In Mac OS X version 10.2 and later, `CGDataProviderRef` is derived from `CFTypeRef` and inherits the properties that all Core Foundation types have in common. For more information, see *CFType Reference*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDataProvider.h`

CGDataProviderCallbacks

Defines a structure containing pointers to client-defined callback functions that manage the sending of data for a sequential-access data provider.

```
struct CGDataProviderCallbacks {
    CGDataProviderGetBytesCallback getBytes;
    CGDataProviderSkipBytesCallback skipBytes;
    CGDataProviderRewindCallback rewind;
    CGDataProviderReleaseInfoCallback releaseProvider;
};
typedef struct CGDataProviderCallbacks CGDataProviderCallbacks;
```

Fields

`getBytes`

A pointer to a function that copies data from the provider. For more information, see `CGDataProviderGetBytesCallback` (page 159).

`skipBytes`

A pointer to a function that Quartz calls to advance the stream of data supplied by the provider. For more information, see `CGDataProviderSkipBytesCallback` (page 162).

`rewind`

A pointer to a function Quartz calls to return the provider to the beginning of the data stream. For more information, see `CGDataProviderRewindCallback` (page 162).

`releaseProvider`

A pointer to a function that handles clean-up for the data provider, or `NULL`. For more information, see `CGDataProviderReleaseInfoCallback` (page 161).

Discussion

The functions specified by the `CGDataProviderCallbacks` structure are responsible for sequentially copying data to a memory buffer for Quartz to use. The functions are also responsible for handling the data provider's basic memory management. You supply a `CGDataProviderCallbacks` structure to the function `CGDataProviderCreate` (page 150) to create a sequential-access data provider.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDataProvider.h`

CGDataProviderDirectAccessCallbacks

Defines pointers to client-defined callback functions that manage the sending of data for a direct-access data provider.

```
struct CGDataProviderDirectAccessCallbacks {
    CGDataProviderGetBytePointerCallback getBytePointer;
    CGDataProviderReleaseBytePointerCallback releaseBytePointer;
    CGDataProviderGetBytesAtOffsetCallback getBytes;
    CGDataProviderReleaseInfoCallback releaseProvider;
};
typedef struct CGDataProviderDirectAccessCallbacks
CGDataProviderDirectAccessCallbacks;
```

Fields

`getBytePointer`

A pointer to a function that returns a pointer to the provider's data. For more information, see `CGDataProviderGetBytePointerCallback` (page 156).

releaseBytePointer
: A pointer to a function that Quartz calls to release a pointer to the provider's data. For more information, see CGDataProviderReleaseBytePointerCallback (page 160).

getBytes
: A pointer to a function that copies data from the provider. For more information, see CGDataProviderGetBytesAtOffsetCallback (page 157).

releaseProvider
: A pointer to a function that handles clean-up for the data provider, or NULL. For more information, see CGDataProviderReleaseInfoCallback (page 161).

Discussion

You supply a CGDataProviderDirectAccessCallbacks structure to the function CGDataProviderCreateDirectAccess (page 151) to create a data provider for direct access. The functions specified by the CGDataProviderDirectAccessCallbacks structure are responsible for copying data a block at a time to a memory buffer for Quartz to use. The functions are also responsible for handling the data provider's basic memory management. For the callback to work, one of the getBytePointer and getBytes parameters must be non-NULL. If both are non-NULL, then getBytePointer is used to access the data.

Availability

Available in Mac OS X v10.0 and later.

Declared In

CGDataProvider.h

CGDataProviderDirectCallbacks

Defines pointers to client-defined callback functions that manage the sending of data for a direct-access data provider.

```
struct CGDataProviderDirectCallbacks {
     unsigned int version;
    CGDataProviderGetBytePointerCallback getBytePointer;
    CGDataProviderReleaseBytePointerCallback releaseBytePointer;
    CGDataProviderGetBytesAtPositionCallback getBytesAtPosition;
    CGDataProviderReleaseInfoCallback releaseInfo;
};
typedef struct CGDataProviderDirectCallbacks  CGDataProviderDirectCallbacks;
```

Fields

version
: The version of this structure. It should be set to 0.

getBytePointer
: A pointer to a function that returns a pointer to the provider's data. For more information, see CGDataProviderGetBytePointerCallback (page 156).

releaseBytePointer
: A pointer to a function that Quartz calls to release a pointer to the provider's data. For more information, see CGDataProviderReleaseBytePointerCallback (page 160).

getBytesAtPosition
: A pointer to a function that copies data from the provider.

releaseInfo
A pointer to a function that handles clean-up for the data provider, or NULL. For more information, see CGDataProviderReleaseInfoCallback (page 161).

Discussion
You supply a CGDataProviderDirectCallbacks structure to the function CGDataProviderCreateDirect (page 150) to create a data provider for direct access. The functions specified by the CGDataProviderDirectCallbacks structure are responsible for copying data a block at a time to a memory buffer for Quartz to use. The functions are also responsible for handling the data provider's basic memory management. For the callback to work, one of the getBytePointer and getBytesAtPosition parameters must be non-NULL. If both are non-NULL, then getBytePointer is used to access the data.

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

Declared In
CGDataProvider.h

CGDataProviderSequentialCallbacks

Defines a structure containing pointers to client-defined callback functions that manage the sending of data for a sequential-access data provider.

```
struct CGDataProviderSequentialCallbacks {
    unsigned int version;
    CGDataProviderGetBytesCallback getBytes;
    CGDataProviderSkipForwardCallback skipForward;
    CGDataProviderRewindCallback rewind;
    CGDataProviderReleaseInfoCallback releaseInfo;
};
typedef struct CGDataProviderSequentialCallbacks CGDataProviderSequentialCallbacks;
```

Fields
version
The version of this structure. It should be set to 0.

getBytes
A pointer to a function that copies data from the provider. For more information, see CGDataProviderGetBytesCallback (page 159).

skipForward
A pointer to a function that Quartz calls to advance the stream of data supplied by the provider.

rewind
A pointer to a function Quartz calls to return the provider to the beginning of the data stream. For more information, see CGDataProviderRewindCallback (page 162).

releaseInfo
A pointer to a function that handles clean-up for the data provider, or NULL. For more information, see CGDataProviderReleaseInfoCallback (page 161).

Discussion
The functions specified by the CGDataProviderSequentialCallbacks structure are responsible for sequentially copying data to a memory buffer for Quartz to use. The functions are also responsible for handling the data provider's basic memory management. You supply a CGDataProviderCallbacks structure to the function CGDataProviderCreateSequential (page 151) to create a sequential-access data provider.

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

Declared In
`CGDataProvider.h`

CGFont Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGFont.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGFontRef` opaque type encapsulates font information. A font is a set of shapes or glyphs associated with a character set. A glyph can represent a single character (such as 'b'), more than one character (such as the "fi" ligature), or a special character such as a space. Quartz retrieves the glyphs for the font from ATS (Apple Type Services) and paints the glyphs based on the relevant parameters of the current graphics state.

Quartz provides a limited, low-level interface for drawing text. For information on text-drawing functions, see *CGContext Reference*. For full Unicode and text-layout support, use the services provided by Core Text or ATSUI).

Functions by Task

Retaining and Releasing a CGFont Object

`CGFontRelease` (page 184)
Decrements the retain count of a Quartz font.

`CGFontRetain` (page 185)
Increments the retain count of a Quartz font.

Creating a CGFont Object

`CGFontCreateWithDataProvider` (page 176)
Creates a font object from data supplied from a data provider.

`CGFontCreateWithFontName` (page 177)
Creates a font object corresponding to the font specified by a PostScript or full name.

`CGFontCreateCopyWithVariations` (page 175)
Creates a copy of a font using a variation specification dictionary.

`CGFontCreateWithPlatformFont` (page 177)
Creates a font object from an Apple Type Services (ATS) font.

Working With PostScript Fonts

`CGFontCopyPostScriptName` (page 172)
Obtains the PostScript name of a font.

`CGFontCanCreatePostScriptSubset` (page 171)
Determines whether Quartz can create a subset of the font in PostScript format.

`CGFontCreatePostScriptSubset` (page 176)
Creates a subset of the font in the specified PostScript format.

`CGFontCreatePostScriptEncoding` (page 175)
Creates a PostScript encoding of a font.

Working With Font Tables

`CGFontCopyTableTags` (page 173)
Returns an array of tags that correspond to the font tables for a font.

`CGFontCopyTableForTag` (page 173)
Returns the font table that corresponds to the provided tag.

Getting Font Information

`CGFontGetTypeID` (page 183)
Returns the Core Foundation type identifier for Quartz fonts.

`CGFontCopyVariationAxes` (page 174)
Returns an array of the variation axis dictionaries for a font.

`CGFontCopyVariations` (page 174)
Returns the variation specification dictionary for a font.

`CGFontCopyFullName` (page 171)
Returns the full name associated with a font object.

`CGFontGetAscent` (page 178)
Returns the ascent of a font.

`CGFontGetDescent` (page 179)
Returns the descent of a font.

`CGFontGetLeading` (page 182)
Returns the leading of a font.

`CGFontGetCapHeight` (page 178)
Returns the cap height of a font.

`CGFontGetXHeight` (page 184)
Returns the x-height of a font.

`CGFontGetFontBBox` (page 179)
Returns the bounding box of a font.

`CGFontGetItalicAngle` (page 182)
Returns the italic angle of a font.

`CGFontGetStemV` (page 183)
Returns the thickness of the dominant vertical stems of glyphs in a font.

`CGFontGetGlyphBBoxes` (page 181)
Get the bounding box of each glyph in an array.

`CGFontGetGlyphWithGlyphName` (page 181)
Returns the glyph for the font name associated with the specified font object.

`CGFontCopyGlyphNameForGlyph` (page 172)
Returns the glyph name associated with a font object.

`CGFontGetNumberOfGlyphs` (page 182)
Returns the number of glyphs in a font.

`CGFontGetGlyphAdvances` (page 180)
Gets the bound box of each glyph in the provided array.

`CGFontGetUnitsPerEm` (page 184)
Returns the number of glyph space units per em for the provided font.

Functions

CGFontCanCreatePostScriptSubset

Determines whether Quartz can create a subset of the font in PostScript format.

```
bool CGFontCanCreatePostScriptSubset (
   CGFontRef font,
   CGFontPostScriptFormat format
);
```

Parameters

font
A font object.

Return Value

Returns `true` if a subset in the PostScript format can be created for the font; `false` otherwise.

Discussion

For more information on PostScript format, see *Adobe Type 1 Font Format*, which is available from http://partners.adobe.com/.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGFont.h`

CGFontCopyFullName

Returns the full name associated with a font object.

```
CFStringRef CGFontCopyFullName (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value
The full name associated with the font.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
CGFont.h

CGFontCopyGlyphNameForGlyph

Returns the glyph name associated with a font object.

```
CFStringRef CGFontCopyGlyphNameForGlyph (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value
A glyph name, or `NULL` if there isn't a glyph associated with the font object.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
CGFont.h

CGFontCopyPostScriptName

Obtains the PostScript name of a font.

```
CFStringRef CGFontCopyPostScriptName (
   CGFontRef font
);
```

Parameters

font

A font object.

Return Value
The PostScript name of the font.

Discussion

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGFont.h`

CGFontCopyTableForTag

Returns the font table that corresponds to the provided tag.

```
CFDataRef CGFontCopyTableForTag(
    CGFontRef font,
    uint32_t tag
);
```

Parameters

font
A font object.

tag
The tag for the table you want to obtain.

Return Value
The font table that corresponds to the tag, or `NULL` if no such table exists.

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

Declared In
`CGFont.h`

CGFontCopyTableTags

Returns an array of tags that correspond to the font tables for a font.

```
CFArrayRef CGFontCopyTableTags(
    CGFontRef font
);
```

Parameters

font
A CGFont object.

Return Value
An array of font table tags.

Discussion
Each entry in the returned array is a four-byte value that represents a single TrueType or OpenType font table tag. To obtain a tag at index `k` in a manner that is appropriate for 32-bit and 64-bit architectures, you need to use code similar to the following:

```
tag = (uint32_t)(uintptr_t)CFArrayGetValue(table, k);
```

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

Declared In
`CGFont.h`

CGFontCopyVariationAxes

Returns an array of the variation axis dictionaries for a font.

```
CFArrayRef CGFontCopyVariationAxes (
   CGFontRef font
);
```

Parameters

font
A CGFont object.

Return Value
An array of the variation axis dictionaries. Returns `NULL` if the font doesn't support variations.

Discussion
A variation axis is a range included in a font by the font designer that allows a font to produce different type styles. Each variation axis dictionary contains key-value pairs that specify the variation axis name and the minimum, maximum, and default values for that variation axis.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGFont.h`

CGFontCopyVariations

Returns the variation specification dictionary for a font.

```
CFDictionaryRef CGFontCopyVariations (
   CGFontRef font
);
```

Parameters

font
A font object.

Return Value
The variation specification dictionary for the font. Returns `NULL` if the font doesn't support variations.

Discussion
The variation specification dictionary contains keys that correspond to the variation axis names of the font. Each key is a variation axis name. The value for each key is the value specified for that particular variation axis represented as a CFNumber object.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGFont.h`

CGFontCreateCopyWithVariations

Creates a copy of a font using a variation specification dictionary.

```
CGFontRef CGFontCreateCopyWithVariations (
   CGFontRef font,
   CFDictionaryRef variations
);
```

Parameters

font

The Quartz font to copy.

variations

A variation specification dictionary that contains keys corresponding to the variation axis names of the font. Each key in the dictionary is a variation axis name. The value for each key is the value specified for that particular variation axis represented as a CFNumber object. If a variation axis name is not specified in `variations`, then the current value from `font` is used.

Return Value
The font object.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGFont.h`

CGFontCreatePostScriptEncoding

Creates a PostScript encoding of a font.

```
CFDataRef CGFontCreatePostScriptEncoding (
   CGFontRef font,
   const CGGlyph encoding[256]
);
```

Parameters

font

A CGFont object.

encoding

The encoding to use.

Return Value
A PostScript encoding of the font that contains glyphs in the specified encoding.

Discussion
For more information on PostScript format, see *Adobe Type 1 Font Format*, which is available from http://partners.adobe.com/.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGFont.h`

CGFontCreatePostScriptSubset

Creates a subset of the font in the specified PostScript format.

```
CFDataRef CGFontCreatePostScriptSubset (
   CGFontRef font,
   CFStringRef subsetName,
   CGFontPostScriptFormat format,
   const CGGlyph glyphs[],
   size_t count,
   const CGGlyph encoding[256]
);
```

Parameters

font
A font object.

subsetName
The name of the subset.

format
The PostScript format of the font.

glyphs
An array that contains the glyphs in the subset.

count
The number of glyphs specified by the `glyphs` array.

encoding
The default encoding for the subset. You can pass `NULL` if you do not want to specify an encoding.

Return Value
A subset of the font created from the supplied parameters.

Discussion
For more information on PostScript format, see *Adobe Type 1 Font Format*, which is available from http://partners.adobe.com/.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGFont.h`

CGFontCreateWithDataProvider

Creates a font object from data supplied from a data provider.

```
CGFontRef CGFontCreateWithDataProvider (
    CGDataProviderRef provider
);
```

Parameters

provider

A data provider.

Return Value

The font object or `NULL` if the font can't be created. You are responsible for releasing this object using `CGFontRelease` (page 184).

Discussion

Before drawing text in a Quartz context, you must set the font in the current graphics state by calling the function `CGContextSetFontSize` (page 108).

Availability

Available in Mac OS X version 10.5 and later.

Declared In

`CGFont.h`

CGFontCreateWithFontName

Creates a font object corresponding to the font specified by a PostScript or full name.

```
CGFontRef CGFontCreateWithFontName (
    CFStringRef name
);
```

Parameters

name

The PostScript or full name of a font.

Return Value

The font object or `NULL` if the font can't be created. You are responsible for releasing this object using `CGFontRelease` (page 184).

Discussion

Before drawing text in a Quartz context, you must set the font in the current graphics state by calling the function `CGContextSetFont` (page 107).

Availability

Available in Mac OS X version 10.5 and later.

Declared In

`CGFont.h`

CGFontCreateWithPlatformFont

Creates a font object from an Apple Type Services (ATS) font.

```
CGFontRef CGFontCreateWithPlatformFont (
   void *platformFontReference
);
```

Parameters

platformFontReference

A generic pointer to a font object. The font should be of a type appropriate to the platform on which your program is running. For Mac OS X, you should pass a pointer to an ATS font.

Return Value

The font object, or `NULL` if the platform font could not be located. You are responsible for releasing this object using `CGFontRelease` (page 184).

Discussion

Before drawing text in a Quartz context, you must set the font in the current graphics state. For ATS Fonts, call this function to create a Quartz font, and pass it to `CGContextSetFont` (page 107).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGFont.h`

CGFontGetAscent

Returns the ascent of a font.

```
int CGFontGetAscent (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value

The ascent of the font.

Discussion

The ascent is the maximum distance above the baseline of glyphs in a font. The value is specified in glyph space units.

Availability

Available in Mac OS X version 10.5 and later.

Declared In

`CGFont.h`

CGFontGetCapHeight

Returns the cap height of a font.

```
int CGFontGetCapHeight (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value
The cap height of the font.

Discussion
The cap height is the distance above the baseline of the top of flat capital letters of glyphs in a font. The value is specified in glyph space units.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
`CGFont.h`

CGFontGetDescent

Returns the descent of a font.

```
int CGFontGetDescent (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value
The descent of the font .

Discussion
The descent is the maximum distance below the baseline of glyphs in a font. The value is specified in glyph space units.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
`CGFont.h`

CGFontGetFontBBox

Returns the bounding box of a font.

```
CGRect CGFontGetFontBBox (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value
The bounding box of the font.

Discussion
The font bounding box is the union of all of the bounding boxes for all the glyphs in a font. The value is specified in glyph space units.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
`CGFont.h`

CGFontGetGlyphAdvances

Gets the bound box of each glyph in the provided array.

```
bool CGFontGetGlyphAdvances (
    CGFontRef font,
    const CGGlyph glyphs[],
    size_t count,
    int advances[]
);
```

Parameters

font

The font object associated with the provided glyphs.

glyphs

An array of glyphs.

count

The number of glyphs in the array.

advances

On output, an array of of advances for the provided glyphs.

Return Value
`TRUE` unless the advances can't be provided for some reason.

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

Declared In
`CGFont.h`

CGFontGetGlyphBBoxes

Get the bounding box of each glyph in an array.

```
bool CGFontGetGlyphBBoxes (
    CGFontRef font,
    const CGGlyph glyphs[],
    size_t count,
    CGRect bboxes[]
);
```

Parameters

font

A font object.

glyphs

A array of glyphs.

count

The number of items in the `glyphs` array.

bboxes

On return, the bounding boxes for each glyph.

Return Value

`false` if bounding boxes can't be retrieved for any reason; `true` otherwise.

Availability

Available in Mac OS X version 10.5 and later.

Declared In

`CGFont.h`

CGFontGetGlyphWithGlyphName

Returns the glyph for the font name associated with the specified font object.

```
CGGlyph CGFontGetGlyphWithGlyphName (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value

A glyph, or `0` if there isn't a name associated with the font object.

Availability

Available in Mac OS X version 10.5 and later.

Declared In

`CGFont.h`

CGFontGetItalicAngle

Returns the italic angle of a font.

```
CGFloat CGFontGetItalicAngle (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value

The italic angle of the font, measured in degrees counter-clockwise from the vertical.

Availability

Available in Mac OS X version 10.5 and later.

Declared In

CGFont.h

CGFontGetLeading

Returns the leading of a font.

```
int CGFontGetLeading (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value

The leading of the font.

Discussion

The leading is the spacing between consecutive lines of text in a font. The value is specified in glyph space units.

Availability

Available in Mac OS X version 10.5 and later.

Declared In

CGFont.h

CGFontGetNumberOfGlyphs

Returns the number of glyphs in a font.

```
size_t CGFontGetNumberOfGlyphs (
    CGFontRef font
);
```

Parameters

font

A `CGFont` object.

Return Value
The number of glyphs in the provided font.

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

Declared In
`CGFont.h`

CGFontGetStemV

Returns the thickness of the dominant vertical stems of glyphs in a font.

```
CGFloat CGFontGetItalicAngle (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value
The thickness of the dominant vertical stems of glyphs in a font.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
`CGFont.h`

CGFontGetTypeID

Returns the Core Foundation type identifier for Quartz fonts.

```
CFTypeID CGFontGetTypeID (
   void
);
```

Return Value
The Core Foundation identifier for the opaque type `CGFontRef` (page 185).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGFont.h`

CGFontGetUnitsPerEm

Returns the number of glyph space units per em for the provided font.

```
int CGFontGetUnitsPerEm (
    CGFontRef font
);
```

Parameters

font

A `CGFont` object.

Return Value

The number of glyph space units per em for the provided font.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CGFont.h`

CGFontGetXHeight

Returns the x-height of a font.

```
int CGFontGetXHeight (
    CGFontRef font
);
```

Parameters

font

A font object.

Return Value

The x-height of the font.

Discussion

The x-height is the distance above the baseline of the top of flat, non-ascending lowercase letters (such as `x`) of glyphs in a font. The value is specified in glyph space units.

Availability

Available in Mac OS X version 10.5 and later.

Declared In

`CGFont.h`

CGFontRelease

Decrements the retain count of a Quartz font.

```
void CGFontRelease (
   CGFontRef font
);
```

Parameters

font

The Quartz font to release.

Discussion

This function is equivalent to `CFRelease`, except that it does not cause an error if the `font` parameter is `NULL`.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGFont.h`

CGFontRetain

Increments the retain count of a Quartz font.

```
CGFontRef CGFontRetain (
   CGFontRef font
);
```

Parameters

font

The Quartz font to retain.

Return Value

The same font you specified in the `font` parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `font` parameter is `NULL`.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGFont.h`

Data Types

CGFontRef

An opaque type that encapsulates font information.

```
typedef struct CGFont *CGFontRef;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In
CGFont.h

CGFontIndex

An index into a font table.

```
typedef unsigned short CGFontIndex;
```

Discussion
This integer type provides an additional way to specify a glyph identifier. `CGFontIndex` is equivalent to `CGGlyph` (page 186), and you can use constants of either type interchangeably.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CGFont.h

CGGlyph

An index into the internal glyph table of a font.

```
typedef unsigned short CGGlyph;
```

Discussion
When drawing text, you typically specify a sequence of characters. However, Quartz also allows you to use `CGGlyph` values to specify glyphs. In either case, Quartz renders the text using font data provided by the Apple Type Services (ATS) framework.

You provide `CGGlyph` values to the functions `CGContextShowGlyphs` (page 123) and `CGContextShowGlyphsAtPoint` (page 123). These functions display an array of glyphs at the current text position or at a position you specify, respectively.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGFont.h

Constants

CGFontPostScriptFormat

Possible formats for a PostScript font subset.

```
enum CGFontPostScriptFormat {
    kCGFontPostScriptFormatType1 = 1,
    kCGFontPostScriptFormatType3 = 3,
    kCGFontPostScriptFormatType42 = 42
};
typedef enum CGFontPostScriptFormat CGFontPostScriptFormat;
```

Constants

`kCGFontPostScriptFormatType1`

This is documented in *Adobe Type 1 Font Format*, which is available from http://partners.adobe.com/.

Available in Mac OS X v10.4 and later.

Declared in `CGFont.h`.

`kCGFontPostScriptFormatType3`

This is documented in *PostScript Language Reference, 3rd edition*, which is available from http://partners.adobe.com/.

Available in Mac OS X v10.4 and later.

Declared in `CGFont.h`.

`kCGFontPostScriptFormatType42`

This is documented in *Adobe Technical Note 5012, The Type 42 Font Format Specification*, which is available from http://partners.adobe.com/.

Available in Mac OS X v10.4 and later.

Declared in `CGFont.h`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGFont.h`

Font Table Index Values

Possible values for an index into a font table.

```
enum {
    kCGFontIndexMax = ((1 << 16) - 2),
    kCGFontIndexInvalid = ((1 << 16) - 1),
    kCGGlyphMax = kCGFontIndexMax
};
```

Constants

`kCGFontIndexMax`

The maximum allowed value for `CGFontIndex` (page 186).

Available in Mac OS X v10.1 and later.

Declared in `CGFont.h`.

`kCGFontIndexInvalid`

An invalid font index (a value which never represents a valid glyph).

Available in Mac OS X v10.1 and later.

Declared in `CGFont.h`.

`kCGGlyphMax`
The same as `kCGFontIndexMax`.

Available in Mac OS X v10.1 and later.

Declared in `CGFont.h`.

Discussion
See `CGFontIndex` (page 186).

Declared In
`CGFont.h`

Font Variation Axis Keys

Keys used for a font variation axis dictionary.

```
const CFStringRef kCGFontVariationAxisName
const CFStringRef kCGFontVariationAxisMinValue
const CFStringRef kCGFontVariationAxisMaxValue
const CFStringRef kCGFontVariationAxisDefaultValue
```

Constants
`kCGFontVariationAxisName`
The key used to obtain the variation axis name from a variation axis dictionary. The value obtained with this key is a `CFStringRef` that specifies the name of the variation axis.

Available in Mac OS X v10.4 and later.

Declared in `CGFont.h`.

`kCGFontVariationAxisMinValue`
The key used to obtain the minimum variation axis value from a variation axis dictionary. The value obtained with this key is a `CFNumberRef` that specifies the minimum value of the variation axis.

Available in Mac OS X v10.4 and later.

Declared in `CGFont.h`.

`kCGFontVariationAxisMaxValue`
The key used to obtain the maximum variation axis value from a variation axis dictionary. The value obtained with this key is a `CFNumberRef` that specifies the maximum value of the variation axis.

Available in Mac OS X v10.4 and later.

Declared in `CGFont.h`.

`kCGFontVariationAxisDefaultValue`
The key used to obtain the default variation axis value from a variation axis dictionary. The value obtained with this key is a `CFNumberRef` that specifies the default value of the variation axis.

Available in Mac OS X v10.4 and later.

Declared in `CGFont.h`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGFont.h`

CGFunction Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGFunction.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGFunctionRef` opaque type provides a general facility for defining and using callback functions. These functions can take an arbitrary number of floating-point input values and pass back an arbitrary number of floating-point output values.

Quartz uses CGFunction objects to implement shadings. *CGShading Reference* describes the parameters and semantics required for the callbacks used by CGFunction objects.

Functions by Task

Creating a CGFunction Object

`CGFunctionCreate` (page 190)
Creates a Quartz function.

Retaining and Releasing CGFunction Objects

`CGFunctionRelease` (page 191)
Decrements the retain count of a function object.

`CGFunctionRetain` (page 191)
Increments the retain count of a function object.

Getting the CFType ID

`CGFunctionGetTypeID` (page 191)
Returns the type identifier for Quartz function objects.

Functions

CGFunctionCreate

Creates a Quartz function.

```
CGFunctionRef CGFunctionCreate (
   void *info,
   size_t domainDimension,
   const CGFloat *domain,
   size_t rangeDimension,
   const CGFloat *range,
   const CGFunctionCallbacks *callbacks
);
```

Parameters

info

A pointer to user-defined storage for data that you want to pass to your callbacks. You need to make sure that the data persists for as long as it's needed, which can be beyond the scope in which the Quartz function is used.

domainDimension

The number of inputs.

domain

An array of (`2*domainDimension`) floats used to specify the valid intervals of input values. For each `k` from `0` to `(domainDimension - 1)`, `domain[2*k]` must be less than or equal to `domain[2*k+1]`, and the `k`th input value will be clipped to lie in the interval `domain[2*k] ≤ input[k] ≤ domain[2*k+1]`. If this parameter is `NULL`, then the input values are not clipped.

rangeDimension

The number of outputs.

range

An array of `(2*rangeDimension)` floats that specifies the valid intervals of output values. For each `k` from `0` to `(rangeDimension - 1)`, `range[2*k]` must be less than or equal to `range[2*k+1]`, and the `k`th output value will be clipped to lie in the interval `range[2*k] ≤ output[k] ≤ range[2*k+1]`. If this parameter is `NULL`, then the output values are not clipped.

callbacks

A pointer to a callback function table. This table should contain pointers to the callbacks you provide to implement the semantics of this Quartz function. Quartz makes a copy of your table, so, for example, you could safely pass in a pointer to a structure on the stack.

Return Value

The new Quartz function. You are responsible for releasing this object using `CGFunctionRelease` (page 191).

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGFunction.h`

CGFunctionGetTypeID

Returns the type identifier for Quartz function objects.

```
CFTypeID CGFunctionGetTypeID (
    void
);
```

Return Value
The identifier for the opaque type `CGFunctionRef` (page 193).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGFunction.h`

CGFunctionRelease

Decrements the retain count of a function object.

```
void CGFunctionRelease (
   CGFunctionRef function
);
```

Parameters

`function`
The function object to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the `function` parameter is `NULL`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGFunction.h`

CGFunctionRetain

Increments the retain count of a function object.

```
CGFunctionRef CGFunctionRetain (
   CGFunctionRef function
);
```

Parameters

`function`
The same function object you passed in as the `function` parameter.

Return Value

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `function` parameter is `NULL`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGFunction.h`

Callbacks

CGFunctionEvaluateCallback

Performs custom operations on the supplied input data to produce output data.

```
typedef void (*CGFunctionEvaluateCallback) (
    void *info,
    const float *inData,
    float *outData
);
```

If you name your function `MyCGFunctionEvaluate`, you would declare it like this:

```
void MyCGFunctionEvaluate (
    void *info,
    const float *inData,
    float *outData
);
```

Parameters

info

The `info` parameter passed to `CGFunctionCreate` (page 190).

inData

An array of floats. The size of the array is that specified by the `domainDimension` parameter passed to the `CGFunctionCreate` (page 190) function.

outData

An array of floats. The size of the array is that specified by the `rangeDimension` parameter passed to the `CGFunctionCreate` (page 190) function.

Discussion
The callback you write is responsible for implementing the calculation of output values from the supplied input values. For example, if you want to implement a simple "squaring" function of one input argument to one output argument, your evaluation function might be:

```
void evaluateSquare(void *info, const float *inData, float *outData)
{
    outData[0] = inData[0] * inData[0];
}
```

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGFunction.h

CGFunctionReleaseInfoCallback

Performs custom clean-up tasks when Quartz deallocates a CGFunction object.

```
typedef void (*CGFunctionReleaseInfoCallback) (
    void *info
);
```

If you name your function `MyCGFunctionReleaseInfo`, you would declare it like this:

```
void MyCGFunctionReleaseInfo (
    void *info
);
```

Parameters

info

The `info` parameter passed to `CGFunctionCreate` (page 190).

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGFunction.h

Data Types

CGFunctionRef

An opaque type that represents a callback function.

```
typedef struct CGFunction *CGFunctionRef;
```

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CGFunction.h

CGFunctionCallbacks

A structure that contains callbacks needed by a CGFunction object.

```
struct CGFunctionCallbacks
{
  unsigned int version;
  CGFunctionEvaluateCallback evaluate;
  CGFunctionReleaseInfoCallback releaseInfo
};

typedef struct CGFunctionCallbacks CGFunctionCallbacks;
```

Fields

`version`

The structure version number. For this structure, the version should be `0`.

`evaluate`

The callback that evaluates the function.

`releaseInfo`

If non-`NULL`, the callback used to release the `info` parameter passed to `CGFunctionCreate` (page 190).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGFunction.h`

CGGLContext Reference

Derived From:	`CGContextRef` (page 131)
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGGLContext.h
Companion guide	Quartz 2D Programming Guide

Overview

The CGGLContext header file defines functions that create and update a graphics context for OpenGL drawing. A CGGLContext context is a type of `CGContextRef` (page 131) that is used for OpenGL content. However, its use is not recommended.

Functions

CGGLContextCreate

Creates a Quartz graphics context from an OpenGL context.

```
CGContextRef CGGLContextCreate (
   void *glContext,
   CGSize size,
   CGColorSpaceRef colorspace
);
```

Parameters

glContext

The context that the OpenGL system uses to manage OpenGL drawing.

size

The dimensions of the OpenGL viewport rectangle.

colorspace

An RGB color space that serves as the destination space when rendering device-independent colors. If `NULL`, Quartz uses the default RGB color space. Quartz retains the color space you pass in; on return, you may safely release it.

Return Value

A new Quartz graphics context. You are responsible for releasing this object by calling `CGContextRelease` (page 96).

Discussion
The use of this function is not recommended.

Creates a Quartz context from the OpenGL context `glContext`. The context establishes an OpenGL viewport rectangle with dimensions specified by the `size` parameter by calling `glViewport(3G)`. If non-`NULL`, the `colorspace` parameter should be an RGB profile that specifies the destination space when rendering device-independent colors.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGGLContext.h`

CGGLContextUpdateViewportSize

Updates the size of the viewport associated with an OpenGL context.

```
void CGGLContextUpdateViewportSize (
   CGContextRef c,
   CGSize size
);
```

Parameters

context
A Quartz graphics context obtained by calling `CGGLContextCreate` (page 195).

size
The new dimensions of the OpenGL viewport.

Discussion
The use of this function is not recommended.

You should call this function whenever the size of the associated OpenGL context changes.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGGLContext.h`

CGGradient Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGGradient.h
Companion guide	Quartz 2D Programming Guide

Overview

A gradient defines a smooth transition between colors across an area. The `CGGradientRef` opaque type, and the functions that operate on it, make creating and using radial and axial gradient fills an easy task. A CGGradient object has a color space, two or more colors, and a location for each color. The color space cannot be a pattern or indexed color space, otherwise it can be any Quartz color space (`CGColorSpaceRef` (page 50)).

Colors can be provided as component values (such as red, green, blue) or as Quartz color objects (`CGColorRef` (page 35)). In Quartz, component can vary from 0.0 to 1.0, designating the proportion of the component present in the color.

A location is a normalized value. When it comes time to paint the gradient, Quartz maps the normalized location values to the points in coordinate space that you provide.

If you want more precise control over gradients, or if your application runs in versions of Mac OS X that are earlier than v10.5, see *CGShading Reference*.

Functions by Task

Creating a CGGradient Object

`CGGradientCreateWithColorComponents` (page 198)
Creates a CGGradient object from a color space and the provided color components and locations.

`CGGradientCreateWithColors` (page 199)
Creates a CGGradient object from a color space and the provided color objects and locations.

Retaining and Releasing a CGGradient Object

`CGGradientRelease` (page 200)
Decrements the retain count of a CGGradient object.

`CGGradientRetain` (page 200)
Increments the retain count of a CGGradient object.

Getting the Type ID for a CGGradient Object

`CGGradientGetTypeID` (page 200)
Returns the Core Foundation type identifier for CGGradient objects.

Functions

CGGradientCreateWithColorComponents

Creates a CGGradient object from a color space and the provided color components and locations.

```
CGGradientRef CGGradientCreateWithColorComponents(
    CGColorSpaceRef space,
    const CGFloat components[],
    const CGFloat locations[],
    size_t count
);
```

Parameters

space
The color space to use for the gradient. You cannot use a pattern or indexed color space.

components
The color components for each color that defines the gradient. The components should be in the color space specified by `space`. If you are unsure of the number of components, you can call the function `CGColorSpaceGetNumberOfComponents` (page 48).

The number of items in this array should be the product of `count` and the number of components in the color space. For example, if the color space is an RGBA color space and you want to use two colors in the gradient (one for a starting location and another for an ending location), then you need to provide 8 values in `components`—red, green, blue, and alpha values for the first color, followed by red, green, blue, and alpha values for the second color.

locations
The location for each color provided in `components`. Each location must be a `CGFloat` value in the range of 0 to 1, inclusive. If 0 and 1 are not in the `locations` array, Quartz uses the colors provided that are closest to 0 and 1 for those locations.

If `locations` is `NULL`, the first color in `colors` is assigned to location `0`, the last color in`colors` is assigned to location `1`, and intervening colors are assigned locations that are at equal intervals in between.

count
The number of locations provided in the `locations` parameters.

Return Value
A CGGradient object.

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

See Also
`CGContextDrawLinearGradient` (page 80)
`CGContextDrawRadialGradient` (page 83)

Declared In
`CGGradient.h`

CGGradientCreateWithColors

Creates a CGGradient object from a color space and the provided color objects and locations.

```
CGGradientRef CGGradientCreateWithColors(
    CGColorSpaceRef space,
    CFArrayRef colors,
    const CGFloat locations[]
);
```

Parameters

space

The color space to use for the gradient. You cannot use a pattern or indexed color space.

colors

A non-empty array of CGColor objects that should be in the color space specified by `space`. If `space` is not `NULL`, each color will be converted (if necessary) to that color space and the gradient will drawn in that color space. Otherwise, each color will be converted to and drawn in the GenericRGB color space.

locations

The location for each color provided in `colors`; each location must be a `CGFloat` value in the range of 0 to 1, inclusive. If 0 and 1 are not in the `locations` array, Quartz uses the colors provided that are closest to 0 and 1 for those locations.

If `locations` is `NULL`, the first color in `colors` is assigned to location `0`, the last color in`colors` is assigned to location `1`, and intervening colors are assigned locations that are at equal intervals in between.

The `locations` array should contain the same number of items as the `colors` array.

Return Value
A CGGradient object.

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

See Also
`CGContextDrawLinearGradient` (page 80)
`CGContextDrawRadialGradient` (page 83)

Declared In
`CGGradient.h`

CGGradientGetTypeID

Returns the Core Foundation type identifier for CGGradient objects.

```
CFTypeID CGGradientGetTypeID (
   void
);
```

Return Value
The Core Foundation identifier for the opaque type `CGGradientRef`.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
`CGGradient.h`

CGGradientRelease

Decrements the retain count of a CGGradient object.

```
void CGGradientRelease (
   CGGradientRef gradient
);
```

Parameters
gradient
The gradient object to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the *gradient* parameter is `NULL`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGGradient.h`

CGGradientRetain

Increments the retain count of a CGGradient object.

```
CGGradientRef CGGradientRetain(
   CGGradientRef gradient
);
```

Parameters
gradient
The gradient object to retain.

Return Value
The same gradient object that you passed in as the `gradient` parameter.

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `gradient` parameter is `NULL`.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
`CGGradient.h`

Data Types

CGGradientRef

An opaque type that represents a Quartz gradient.

```
typedef struct CGGradient *CGGradientRef;
```

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

Declared In
`CGGradient.h`

Constants

Gradient Drawing Options

Drawing locations for gradients.

```
enum {
    kCGGradientDrawsBeforeStartLocation = (1 << 0),
    kCGGradientDrawsAfterEndLocation = (1 << 1)
};
typedef enum CGGradientDrawingOptions CGGradientDrawingOptions;
```

Constants

`kCGGradientDrawsBeforeStartLocation`
The fill should extend beyond the starting location. The color that extends beyond the starting point is the solid color defined by the CGGradient object to be at location 0.

Available in Mac OS X v10.5 and later.

Declared in `CGGradient.h`.

`kCGGradientDrawsAfterEndLocation`
The fill should extend beyond the ending location. The color that extends beyond the ending point is the solid color defined by the CGGradient object to be at location 1.

Available in Mac OS X v10.5 and later.

Declared in `CGGradient.h`.

Declared In
`CGGradient.h`

CGImage Reference

Derived From:	*CFType Reference*
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGImage.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGImageRef` opaque type represents bitmap images and bitmap image masks, based on sample data that you supply. A bitmap (or sampled) image is a rectangular array of pixels, with each pixel representing a single sample or data point in a source image.

Functions by Task

Creating Bitmap Images

`CGImageCreate` (page 205)
Creates a bitmap image from data supplied by a data provider.

`CGImageCreateCopy` (page 206)
Creates a copy of a bitmap image.

`CGImageCreateCopyWithColorSpace` (page 207)
Create a copy of a bitmap image, replacing its colorspace.

`CGImageCreateWithJPEGDataProvider` (page 208)
Creates a bitmap image using JPEG-encoded data supplied by a data provider.

`CGImageCreateWithPNGDataProvider` (page 210)
Creates a Quartz bitmap image using PNG-encoded data supplied by a data provider.

`CGImageCreateWithImageInRect` (page 207)
Creates a bitmap image using the data contained within a subregion of an existing bitmap image.

`CGImageCreateWithMask` (page 209)
Creates a bitmap image from an existing image and an image mask.

`CGImageCreateWithMaskingColors` (page 209)
Creates a bitmap image by masking an existing bitmap image with the provided color values.

Creating an Image Mask

`CGImageMaskCreate` (page 217)
Creates a bitmap image mask from data supplied by a data provider.

Retaining and Releasing Images

`CGImageRetain` (page 219)
Increments the retain count of a bitmap image.

`CGImageRelease` (page 218)
Decrements the retain count of a bitmap image.

Getting the CFType ID

`CGImageGetTypeID` (page 216)
Returns the type identifier for Quartz bitmap images.

Getting Information About an Image

`CGImageGetAlphaInfo` (page 211)
Returns the alpha channel information for a bitmap image.

`CGImageGetBitmapInfo` (page 211)
Returns the bitmap information for a bitmap image.

`CGImageGetBitsPerComponent` (page 212)
Returns the number of bits allocated for a single color component of a bitmap image.

`CGImageGetBitsPerPixel` (page 212)
Returns the number of bits allocated for a single pixel in a bitmap image.

`CGImageGetBytesPerRow` (page 213)
Returns the number of bytes allocated for a single row of a bitmap image.

`CGImageGetColorSpace` (page 213)
Return the color space for a bitmap image.

`CGImageGetDataProvider` (page 214)
Returns the data provider for a bitmap image.

`CGImageGetDecode` (page 214)
Returns the decode array for a bitmap image.

`CGImageGetHeight` (page 214)
Returns the height of a bitmap image.

`CGImageGetShouldInterpolate` (page 215)
Returns the interpolation setting for a bitmap image.

`CGImageGetRenderingIntent` (page 215)
Returns the rendering intent setting for a bitmap image.

`CGImageGetWidth` (page 216)
Returns the width of a bitmap image.

`CGImageIsMask` (page 217)
Returns whether a bitmap image is an image mask.

Functions

CGImageCreate

Creates a bitmap image from data supplied by a data provider.

```
CGImageRef CGImageCreate (
   size_t width,
   size_t height,
   size_t bitsPerComponent,
   size_t bitsPerPixel,
   size_t bytesPerRow,
   CGColorSpaceRef colorspace,
   CGBitmapInfo bitmapInfo,
   CGDataProviderRef provider,
   const CGFloat decode[],
   bool shouldInterpolate,
   CGColorRenderingIntent intent
);
```

Parameters

width
The width, in pixels, of the required image.

height
The height, in pixels, of the required image

bitsPerComponent
The number of bits for each component in a source pixel. For example, if the source image uses the RGBA-32 format, you would specify 8 bits per component.

bitsPerPixel
The total number of bits in a source pixel. This value must be at least `bitsPerComponent` times the number of components per pixel.

bytesPerRow
The number of bytes of memory for each horizontal row of the bitmap.

colorspace
The color space for the image. Quartz retains the color space you pass in; on return, you may safely release it.

bitmapInfo
A `CGBitmapInfo` constant that specifies whether the bitmap should contain an alpha channel and its relative location in a pixel, along with whether the components are floating-point or integer values.

provider
The source of data for the bitmap. For information about supported data formats, see the discussion below. Quartz retains this object; on return, you may safely release it.

decode

The decode array for the image. If you do not want to allow remapping of the image's color values, pass `NULL` for the decode array. For each color component in the image's color space, a decode array provides a pair of values denoting the upper and lower limits of a range. For example, the decode array for a source image in the RGB color space would contain six entries total, consisting of one pair each for red, green, and blue. When the image is rendered, Quartz uses a linear transform to map the original component value into a relative number within your designated range that is appropriate for the destination color space.

shouldInterpolate

A Boolean value that specifies whether interpolation should occur. The interpolation setting specifies whether Quartz should apply a pixel-smoothing algorithm to the image. Without interpolation, the image may appear jagged or pixelated when drawn on an output device with higher resolution than the image data.

intent

A rendering intent constant that specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context. The rendering intent determines the exact method used to map colors from one color space to another. For descriptions of the defined rendering-intent constants, see `Color Rendering Intents` (page 53).

Return Value
A new Quartz bitmap image. You are responsible for releasing this object by calling `CGImageRelease` (page 218).

Discussion
The data provider should provide raw data that matches the format specified by the other input parameters. To use encoded data (for example, from a file specified by a URL-based data provider), see `CGImageCreateWithJPEGDataProvider` (page 208) and `CGImageCreateWithPNGDataProvider` (page 210). In Mac OS X version 10.3 and later, you can also use the QuickTime function `GraphicsImportCreateCGImage` to decode an image file in any supported format and create a CGImage, in a single operation.

For information on supported pixel formats, see *Quartz 2D Programming Guide*.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

CGImageCreateCopy

Creates a copy of a bitmap image.

```
CGImageRef CGImageCreateCopy (
   CGImageRef image
);
```

Parameters

image

The image to copy.

Return Value
An copy of the image specified by the `image` parameter.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGImage.h`

CGImageCreateCopyWithColorSpace

Create a copy of a bitmap image, replacing its colorspace.

```
CGImageRef CGImageCreateCopyWithColorSpace (
   CGImageRef image,
   CGColorSpaceRef colorspace
);
```

Parameters

`image`
The graphics image to copy.

`colorspace`
The destination color space. The number of components in this color space must be the same as the number in the specified image.

Return Value
A new Quartz image that is a copy of the image passed as the `image` parameter but with its color space replaced by that specified by the `colorspace` parameter. Returns `NULL` if `image` is an image mask, or if the number of components of `colorspace` is not the same as the number of components of the colorspace of `image`. You are responsible for releasing this object using `CGImageRelease` (page 218).

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGImage.h`

CGImageCreateWithImageInRect

Creates a bitmap image using the data contained within a subregion of an existing bitmap image.

```
CGImageRef CGImageCreateWithImageInRect (
   CGImageRef image,
   CGRect rect
);
```

Parameters

`image`
The image to extract the subimage from.

`rect`
A rectangle whose coordinates specify the area to create an image from.

Return Value
A CGImage object that specifies a subimage of the image. If the `rect` parameter defines an area that is not in the image, returns `NULL`.

Discussion
Quartz performs these tasks to create the subimage:

- Adjusts the area specified by the `rect` parameter to integral bounds by calling the function `CGRectIntegral`.
- Intersects the result with a rectangle whose origin is `(0,0)` and size is equal to the size of the image specified by the `image` parameter.
- References the pixels within the resulting rectangle, treating the first pixel within the rectangle as the origin of the subimage.

If `W` and `H` are the width and height of image, respectively, then the point `(0,0)` corresponds to the first pixel of the image data. The point `(W-1, 0)` is the last pixel of the first row of the image data while `(0, H-1)` is the first pixel of the last row of the image data and `(W-1, H-1)` is the last pixel of the last row of the image data.

The resulting image retains a reference to the original image, which means you may release the original image after calling this function.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGImage.h`

CGImageCreateWithJPEGDataProvider

Creates a bitmap image using JPEG-encoded data supplied by a data provider.

```
CGImageRef CGImageCreateWithJPEGDataProvider (
   CGDataProviderRef source,
   const CGFloat decode[],
   bool shouldInterpolate,
   CGColorRenderingIntent intent
);
```

Parameters

source
A data provider supplying JPEG-encoded data.

decode
The decode array for the image. Typically a decode array is unnecessary, and you should pass `NULL`.

shouldInterpolate
A Boolean value that specifies whether interpolation should occur. The interpolation setting specifies whether Quartz should apply a pixel-smoothing algorithm to the image.

intent
A `CGColorRenderingIntent` constant that specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context.

Return Value
A new Quartz bitmap image. You are responsible for releasing this object by calling `CGImageRelease` (page 218).

Availability
Available in Mac OS X version 10.1 and later.

Declared In
`CGImage.h`

CGImageCreateWithMask

Creates a bitmap image from an existing image and an image mask.

```
CGImageRef CGImageCreateWithMask (
   CGImageRef image,
   CGImageRef mask
);
```

Parameters

image

The image to apply the `mask` parameter to. This image must not be an image mask and may not have an image mask or masking color associated with it.

mask

A mask. If the mask is an image, it must be in the DeviceGray color space, must not have an alpha component, and may not itself be masked by an image mask or a masking color. If the mask is not the same size as the image specified by the `image` parameter, then Quartz scales the mask to fit the image.

Return Value
An image created by masking `image` with `mask`. You are responsible for releasing this object by calling `CGImageRelease` (page 218).

Discussion
The resulting image depends on whether the `mask` parameter is an image mask or an image. If the `mask` parameter is an image mask, then the source samples of the image mask act as an inverse alpha value. That is, if the value of a source sample in the image mask is S, then the corresponding region in `image` is blended with the destination using an alpha value of (1-S). For example, if S is 1, then the region is not painted, while if S is 0, the region is fully painted.

If the `mask` parameter is an image, then it serves as an alpha mask for blending the image onto the destination. The source samples of `mask`' act as an alpha value. If the value of the source sample in mask is S, then the corresponding region in image is blended with the destination with an alpha of S. For example, if S is 0, then the region is not painted, while if S is 1, the region is fully painted.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGImage.h`

CGImageCreateWithMaskingColors

Creates a bitmap image by masking an existing bitmap image with the provided color values.

```
CGImageRef CGImageCreateWithMaskingColors (
   CGImageRef image,
   const CGFloat components[]
);
```

Parameters

image

The image to mask. This parameter may not be an image mask, may not already have an image mask or masking color associated with it, and cannot have an alpha component.

components

An array of color components that specify a color or range of colors to mask the image with. The array must contain 2N values { min[1], max[1], ... min[N], max[N] } where N is the number of components in color space of `image`. Each value in `components` must be a valid image sample value. If `image` has integer pixel components, then each value must be in the range [0 .. 2**`bitsPerComponent` - 1] (where `bitsPerComponent` is the number of bits/component of `image`). If `image` has floating-point pixel components, then each value may be any floating-point number which is a valid color component.

Return Value

An image created by masking `image` with the colors specified in the `components` array. You are responsible for releasing this object by calling `CGImageRelease` (page 218).

Discussion

Any image sample with color value {c[1], ... c[N]} where min[i] <= c[i] <= max[i] for 1 <= i <= N is masked out (that is, not painted). This means that anything underneath the unpainted samples, such as the current fill color, shows through.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGImage.h`

CGImageCreateWithPNGDataProvider

Creates a Quartz bitmap image using PNG-encoded data supplied by a data provider.

```
CGImageRef CGImageCreateWithPNGDataProvider (
   CGDataProviderRef source,
   const CGFloat decode[],
   bool shouldInterpolate,
   CGColorRenderingIntent intent
);
```

Parameters

source

A data provider supplying PNG-encoded data.

decode

The decode array for the image. Typically a decode array is unnecessary, and you should pass `NULL`.

shouldInterpolate

A Boolean value that specifies whether interpolation should occur. The interpolation setting specifies whether Quartz should apply a pixel-smoothing algorithm to the image.

intent
A `CGColorRenderingIntent` constant that specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context.

Return Value
A new Quartz bitmap image. You are responsible for releasing this object by calling `CGImageRelease` (page 218).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGImage.h`

CGImageGetAlphaInfo

Returns the alpha channel information for a bitmap image.

```
CGImageAlphaInfo CGImageGetAlphaInfo (
   CGImageRef image
);
```

Parameters
image
The image to examine.

Return Value
A `CGImageAlphaInfo` constant that specifies (1) whether the bitmap contains an alpha channel, (2) where the alpha bits are located in the image data, and (3) whether the alpha value is premultiplied. For possible values, see "Constants" (page 220). The function returns `kCGImageAlphaNone` if the `image` parameter refers to an image mask.

Discussion
The alpha value is what determines the opacity of a pixel when it is drawn.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

CGImageGetBitmapInfo

Returns the bitmap information for a bitmap image.

```
CGBitmapInfo CGImageGetBitmapInfo (
   CGImageRef image
);
```

Parameters
image
An image.

Return Value
The bitmap information associated with an image.

Discussion
This function returns a constant that specifies:

- The type of bitmap data—floating point or integer. You use the constant `kCGBitmapFloatComponents` to extract this information.
- Whether an alpha channel is in the data, and if so, how the alpha data is stored. You use the constant `kCGBitmapAlphaInfoMask` to extract the alpha information. Alpha information is specified as one of the constants listed in "Alpha Information for Images" (page 220).

You can extract the alpha information

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGImage.h`

CGImageGetBitsPerComponent

Returns the number of bits allocated for a single color component of a bitmap image.

```
size_t CGImageGetBitsPerComponent (
   CGImageRef image
);
```

Parameters

image
The image to examine.

Return Value
The number of bits used in memory for each color component of the specified bitmap image (or image mask). Possible values are 1, 2, 4, or 8. For example, for a 16-bit RGB(A) colorspace, the function would return a value of 4 bits per color component.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

CGImageGetBitsPerPixel

Returns the number of bits allocated for a single pixel in a bitmap image.

```
size_t CGImageGetBitsPerPixel (
   CGImageRef image
);
```

Parameters

image
The image to examine.

Return Value
The number of bits used in memory for each pixel of the specified bitmap image (or image mask).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

CGImageGetBytesPerRow

Returns the number of bytes allocated for a single row of a bitmap image.

```
size_t CGImageGetBytesPerRow (
   CGImageRef image
);
```

Parameters

image
The image to examine.

Return Value
The number of bytes used in memory for each row of the specified bitmap image (or image mask).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

CGImageGetColorSpace

Return the color space for a bitmap image.

```
CGColorSpaceRef CGImageGetColorSpace (
   CGImageRef image
);
```

Parameters

image
The image to examine.

Return Value
The source color space for the specified bitmap image, or `NULL` if the image is an image mask. You are responsible for retaining and releasing the color space as necessary.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

CGImageGetDataProvider

Returns the data provider for a bitmap image.

```
CGDataProviderRef CGImageGetDataProvider (
   CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

The data provider for the specified bitmap image (or image mask). You are responsible for retaining and releasing the data provider as necessary.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGImage.h`

CGImageGetDecode

Returns the decode array for a bitmap image.

```
const CGFloat * CGImageGetDecode (
   CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

The decode array for a bitmap image (or image mask). See the discussion for a description of possible return values.

Discussion

For a bitmap image or image mask, for each color component in the source color space, the decode array contains a pair of values denoting the upper and lower limits of a range. When the image is rendered, Quartz uses a linear transform to map the original component value into a relative number, within the designated range, that is appropriate for the destination color space. If remapping of the image's color values is not allowed, the returned value will be `NULL`.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGImage.h`

CGImageGetHeight

Returns the height of a bitmap image.

```
size_t CGImageGetHeight (
   CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value
The height in pixels of the bitmap image (or image mask).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonCocoa_PictureCursor

Declared In
`CGImage.h`

CGImageGetRenderingIntent

Returns the rendering intent setting for a bitmap image.

```
CGColorRenderingIntent CGImageGetRenderingIntent (
   CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value
Returns the `CGColorRenderingIntent` constant that specifies how Quartz should handle colors that are not located within the gamut of the destination color space of a graphics context in which the image is drawn. If the image is an image mask, this function returns `kCGRenderingIntentDefault`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

CGImageGetShouldInterpolate

Returns the interpolation setting for a bitmap image.

```
bool CGImageGetShouldInterpolate (
   CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value
Returns `1` if interpolation is enabled for the specified bitmap image (or image mask), otherwise, returns `0`.

Discussion
The interpolation setting specifies whether Quartz should apply an edge-smoothing algorithm to the associated image.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

CGImageGetTypeID

Returns the type identifier for Quartz bitmap images.

```
CFTypeID CGImageGetTypeID (
   void
);
```

Return Value
The identifier for the opaque type `CGImageRef` (page 219).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGImage.h`

CGImageGetWidth

Returns the width of a bitmap image.

```
size_t CGImageGetWidth (
   CGImageRef image
);
```

Parameters
image
The image to examine.

Return Value
The width, in pixels, of the specified bitmap image (or image mask).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonCocoa_PictureCursor

Declared In
`CGImage.h`

CGImageIsMask

Returns whether a bitmap image is an image mask.

```
bool CGImageIsMask (
   CGImageRef image
);
```

Parameters

image

The image to examine.

Return Value

A Boolean value that indicates whether the image passed in the `image` parameter is an image mask (`true` indicates that the image is an image mask).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGImage.h`

CGImageMaskCreate

Creates a bitmap image mask from data supplied by a data provider.

```
CGImageRef CGImageMaskCreate (
   size_t width,
   size_t height,
   size_t bitsPerComponent,
   size_t bitsPerPixel,
   size_t bytesPerRow,
   CGDataProviderRef provider,
   const CGFloat decode[],
   bool shouldInterpolate
);
```

Parameters

width

The width, in pixels, of the required image mask.

height

The height, in pixels, of the required image mask.

bitsPerComponent

The number of significant masking bits in a source pixel. For example, if the source image is an 8-bit mask, you specify 8 bits per component. Image masks must be 1, 2, 4, or 8 bits per component.

bitsPerPixel

The total number of bits in a source pixel.

bytesPerRow

The number of bytes to use for each horizontal row of the image mask.

provider

The data source for the image mask.

decode

Typically a decode array is unnecessary, and you should pass `NULL`.

shouldInterpolate

A Boolean value that specifies whether interpolation should occur. The interpolation setting specifies whether Quartz should apply an edge-smoothing algorithm to the image mask.

Return Value
A Quartz bitmap image mask. You are responsible for releasing this object by calling `CGImageRelease` (page 218).

Discussion
A Quartz bitmap image mask is used the same way an artist uses a silkscreen, or a sign painter uses a stencil. The bitmap represents a mask through which a color is transferred. The bitmap itself does not have a color. It gets its color from the fill color currently set in the graphics state.

When you draw into a context with a bitmap image mask, Quartz uses the mask to determine where and how the current fill color is applied to the image rectangle. Each sample value in the mask specifies how much of the current fill color is masked out at a specific location. Effectively, the sample value specifies the opacity of the mask. Larger values represent greater opacity and hence less color applied to the page.

Image masks must be 1, 2, 4, or 8 bits per component. For a 1-bit mask, a sample value of `1` specifies sections of the mask that are masked out; these sections block the current fill color. A sample value of `0` specifies sections of the mask that are not masked out; these sections show the current fill color of the graphics state when the mask is painted. You can think of the sample values as an inverse alpha. That is, a value of `1` is transparent and `0` is opaque.

For image masks that are 2, 4, or 8 bits per component, each component is mapped to a range of 0 to 1 by scaling using this formula:

```
1/(2^bits per component - 1)
```

For example, a 4-bit mask has values that range from 0 to 15. These values are scaled by 1/15 so that each component ranges from 0 to 1. Component values that rescale to 0 or 1 behave the same way as they behave for 1-bit image masks. Values that scale to between 0 and 1 act as an inverse alpha. That is, the fill color is painted as if it has an alpha value of (`1 - MaskSampleValue`). For example, if the sample value of an 8-bit mask scales to `0.8`, the current fill color is painted as if it has an alpha value of `0.2`, that is (`1-0.8`).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

CGImageRelease

Decrements the retain count of a bitmap image.

```
void CGImageRelease (
   CGImageRef image
);
```

Parameters

image

The image to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the `image` parameter is `NULL`.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
WhackedTV

Declared In
`CGImage.h`

CGImageRetain

Increments the retain count of a bitmap image.

```
CGImageRef CGImageRetain (
   CGImageRef image
);
```

Parameters

image
The image to retain.

Return Value
The same image you passed in as the `image` parameter.

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `image` parameter is `NULL`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGImage.h`

Data Types

CGImageRef

An opaque type that encapsulates bitmap image information.

```
typedef struct CGImage *CGImageRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGImage.h`

Constants

Alpha Information for Images

Storage options for alpha component data.

```
enum CGImageAlphaInfo {
    kCGImageAlphaNone,
    kCGImageAlphaPremultipliedLast,
    kCGImageAlphaPremultipliedFirst,
    kCGImageAlphaLast,
    kCGImageAlphaFirst,
    kCGImageAlphaNoneSkipLast,
    kCGImageAlphaNoneSkipFirst
};
typedef enum CGImageAlphaInfo CGImageAlphaInfo;
```

Constants

`kCGImageAlphaFirst`

The alpha component is stored in the most significant bits of each pixel. For example, non-premultiplied ARGB.

Available in Mac OS X v10.0 and later.

Declared in `CGImage.h`.

`kCGImageAlphaLast`

The alpha component is stored in the least significant bits of each pixel. For example, non-premultiplied RGBA.

Available in Mac OS X v10.0 and later.

Declared in `CGImage.h`.

`kCGImageAlphaNone`

There is no alpha channel. If the total size of the pixel is greater than the space required for the number of color components in the color space, the least significant bits are ignored. This value is equivalent to `kCGImageAlphaNoneSkipLast`.

Available in Mac OS X v10.0 and later.

Declared in `CGImage.h`.

`kCGImageAlphaNoneSkipFirst`

There is no alpha channel. If the total size of the pixel is greater than the space required for the number of color components in the color space, the most significant bits are ignored.

Available in Mac OS X v10.0 and later.

Declared in `CGImage.h`.

`kCGImageAlphaOnly`

There is no color data, only an alpha channel.

Available in Mac OS X v10.3 and later.

Declared in `CGImage.h`.

`kCGImageAlphaNoneSkipLast`
: There is no alpha channel. If the total size of the pixel is greater than the space required for the number of color components in the color space, the least significant bits are ignored. This value is equivalent to `kCGImageAlphaNone`.

  Available in Mac OS X v10.0 and later.

  Declared in `CGImage.h`.

`kCGImageAlphaPremultipliedFirst`
: The alpha component is stored in the most significant bits of each pixel and the color components have already been multiplied by this alpha value. For example, premultiplied ARGB.

  Available in Mac OS X v10.0 and later.

  Declared in `CGImage.h`.

`kCGImageAlphaPremultipliedLast`
: The alpha component is stored in the least significant bits of each pixel and the color components have already been multiplied by this alpha value. For example, premultiplied RGBA.

  Available in Mac OS X v10.0 and later.

  Declared in `CGImage.h`.

Discussion
A `CGImageAlphaInfo` constant specifies (1) whether a bitmap contains an alpha channel, (2) where the alpha bits are located in the image data, and (3) whether the alpha value is premultiplied. You can obtain a `CGImageAlphaInfo` constant for an image by calling the function `CGImageGetAlphaInfo` (page 211). (You provide a `CGBitmapInfo` constant to the function `CGImageCreate` (page 205), part of which is a `CGImageAlphaInfo` constant.)

Quartz accomplishes alpha blending by combining the color components of the source image with the color components of the destination image using the linear interpolation formula, where "source" is one color component of one pixel of the new paint and "destination" is one color component of the background image.

Quartz supports premultiplied alpha only for images. You should not premultiply any other color values specified in Quartz.

Declared In
`CGImage.h`

Image Bitmap Information

Component information for a bitmap image.

```
enum {
    kCGBitmapAlphaInfoMask = 0x1F,
    kCGBitmapFloatComponents = (1 << 8),

    kCGBitmapByteOrderMask = 0x7000,
    kCGBitmapByteOrderDefault = (0 << 12),
    kCGBitmapByteOrder16Little = (1 << 12),
    kCGBitmapByteOrder32Little = (2 << 12),
    kCGBitmapByteOrder16Big = (3 << 12),
    kCGBitmapByteOrder32Big = (4 << 12)
};
typedef uint32_t CGBitmapInfo;
```

```
#ifdef __BIG_ENDIAN__
    kCGBitmapByteOrder16Host kCGBitmapByteOrder16Big
    kCGBitmapByteOrder32Host kCGBitmapByteOrder32Big
#else
    kCGBitmapByteOrder16Host kCGBitmapByteOrder16Little
    kCGBitmapByteOrder32Host kCGBitmapByteOrder32Little
#endif
```

Constants

`kCGBitmapAlphaInfoMask`
The alpha information mask. Use this to extract alpha information that specifies whether a bitmap contains an alpha channel and how the alpha channel is generated.

Available in Mac OS X v10.4 and later.

Declared in `CGImage.h`.

`kCGBitmapFloatComponents`
The components of a bitmap are floating-point values.

Available in Mac OS X v10.4 and later.

Declared in `CGImage.h`.

`kCGBitmapByteOrderMask`
The byte ordering of pixel formats.

Available in Mac OS X v10.4 and later.

Declared in `CGImage.h`.

`kCGBitmapByteOrderDefault`
The default byte order.

Available in Mac OS X v10.4 and later.

Declared in `CGImage.h`.

`kCGBitmapByteOrder16Little`
16-bit, little endian format.

Available in Mac OS X v10.4 and later.

Declared in `CGImage.h`.

`kCGBitmapByteOrder32Little`
32-bit, little endian format.

Available in Mac OS X v10.4 and later.

Declared in `CGImage.h`.

`kCGBitmapByteOrder16Big`
16-bit, big endian format.

Available in Mac OS X v10.4 and later.

Declared in `CGImage.h`.

`kCGBitmapByteOrder32Big`
32-bit, big endian format.

Available in Mac OS X v10.4 and later.

Declared in `CGImage.h`.

`kCGBitmapByteOrder16Host`
16-bit, host endian format.

`kCGBitmapByteOrder32Host`
32-bit, host endian format.

Discussion
Applications that store pixel data in memory using ARGB format must take care in how they read data. If the code is not written correctly, it's possible to misread the data which leads to colors or alpha that appear wrong. The Quartz byte order constants specify the byte ordering of pixel formats. To specify byte ordering to Quartz use a bitwise OR operator to combine the appropriate constant with the `bitmapInfo` parameter.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGImage.h`

CGImageDestination Reference

Derived From:	CFType
Framework:	ApplicationServices/ImageIO
Declared in	CGImageDestination.h
Companion guide	Quartz 2D Programming Guide

Overview

CGImageDestination objects, available in Mac OS X v10.4 or later, abstract the data-writing task. An image destination can represent a single image or multiple images. It can contain thumbnail images as well as properties for each image.

The functions described in this reference can write data to three kinds of destinations: a URL, a `CFData` object, and a data consumer. After creating a CGImageDestination object for the appropriate destination, you can add image data and set image properties. When you are finished adding data, call the function `CGImageDestinationFinalize` to write the image data and properties to the URL, `CFData` object, or data consumer.

Functions by Task

Creating Image Destinations

`CGImageDestinationCreateWithDataConsumer` (page 228)
Creates an image destination that writes to the specified data consumer.

`CGImageDestinationCreateWithData` (page 228)
Creates an image destination that writes to a Core Foundation mutable data object.

`CGImageDestinationCreateWithURL` (page 229)
Creates an image destination that writes to a location specified by a URL.

Adding Images

`CGImageDestinationAddImage` (page 226)
Adds an image to an image destination.

`CGImageDestinationAddImageFromSource` (page 227)
Adds an image from an image source to an image destination.

Getting Type Identifiers

`CGImageDestinationCopyTypeIdentifiers` (page 227)
Returns an array of the uniform type identifiers (UTIs) that are supported for image destinations.

`CGImageDestinationGetTypeID` (page 230)
Returns the unique type identifier of an image destination opaque type.

Setting Properties

`CGImageDestinationSetProperties` (page 230)
Applies one or more properties to all images in an image destination.

Finalizing an Image Destination

`CGImageDestinationFinalize` (page 229)
Writes image data and properties to the data, URL, or data consumer associated with the image destination.

Functions

CGImageDestinationAddImage

Adds an image to an image destination.

```
void CGImageDestinationAddImage (
    CGImageDestinationRef idst,
    CGImageRef image,
    CFDictionaryRef properties
);
```

Parameters

idst
An image destination

image
The image to add.

properties
An optional dictionary that specifies the properties of the added image. The dictionary can contain any of the properties described in "Destination Properties" (page 231) or the image properties described in *CGImageProperties Reference*.

Discussion
The function logs an error if you add more images than what you specified when you created the image destination.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGImageDestination.h

CGImageDestinationAddImageFromSource

Adds an image from an image source to an image destination.

```
void CGImageDestinationAddImageFromSource (
    CGImageDestinationRef idst,
    CGImageSourceRef isrc,
    size_t index,
    CFDictionaryRef properties
);
```

Parameters

idst
An image destination.

isrc
An image source.

index
An index that specifies the location of the image in the image source. The index is zero-based.

properties
A dictionary that specifies properties to overwrite or add to the source image properties. If a key in `properties` has the value `kCFNull`, the corresponding property in the image destination is removed. The dictionary can contain any of the properties described in "Destination Properties" (page 231) or the image properties described in *CGImageProperties Reference*.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGImageDestination.h

CGImageDestinationCopyTypeIdentifiers

Returns an array of the uniform type identifiers (UTIs) that are supported for image destinations.

```
CFArrayRef CGImageDestinationCopyTypeIdentifiers (
    void
);
```

Return Value
Returns an array of the UTIs that are supported for image destinations. See Uniform Type Identifiers Overview for a list of system-declared and third-party UTIs that can be returned.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGImageDestination.h

CGImageDestinationCreateWithData

Creates an image destination that writes to a Core Foundation mutable data object.

```
CGImageDestinationRef CGImageDestinationCreateWithData (
    CFMutableDataRef data,
    CFStringRef type,
    size_t count,
    CFDictionaryRef options
);
```

Parameters

data

The data object to write to. For more information on data objects, see *CFData Reference* and Data Objects.

type

The uniform type identifier (UTI) of the resulting image file. See *Uniform Type Identifiers Overview* for a list of system-declared and third-party UTIs.

count

The number of images (not including thumbnail images) that the image file will contain.

options

Reserved for future use. Pass `NULL`.

Return Value

An image destination. You are responsible for releasing this object using `CFRelease`.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGImageDestination.h`

CGImageDestinationCreateWithDataConsumer

Creates an image destination that writes to the specified data consumer.

```
CGImageDestinationRef CGImageDestinationCreateWithDataConsumer (
    CGDataConsumerRef consumer,
    CFStringRef type,
    size_t count,
    CFDictionaryRef options
);
```

Parameters

consumer

The data consumer to write to. For information on data consumers see *CGDataConsumer Reference* and *Quartz 2D Programming Guide*.

type

The uniform type identifier (UTI) of the resulting image file. See *Uniform Type Identifiers Overview* for a list of system-declared and third-party UTIs.

count

The number of images (not including thumbnail images) that the image file will contain.

options
Reserved for future use. Pass `NULL`.

Return Value
An image destination. You are responsible for releasing this object using `CFRelease`.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGImageDestination.h`

CGImageDestinationCreateWithURL

Creates an image destination that writes to a location specified by a URL.

```
CGImageDestinationRef CGImageDestinationCreateWithURL (
    CFURLRef url,
    CFStringRef type,
    size_t count,
    CFDictionaryRef options
);
```

Parameters

url
The URL to write to. If the URL already exists, the data at this location is overwritten.

type
The UTI (uniform type identifier) of the resulting image file. See *Uniform Type Identifiers Overview* for a list of system-declared and third-party UTIs.

count
The number of images (not including thumbnail images) that the image file will contain.

options
Reserved for future use. Pass `NULL`.

Return Value
An image destination. You are responsible for releasing this object using `CFRelease`.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGImageDestination.h`

CGImageDestinationFinalize

Writes image data and properties to the data, URL, or data consumer associated with the image destination.

```
bool CGImageDestinationFinalize (
   CGImageDestinationRef idst
);
```

Parameters

idst

An image destination.

Return Value
Returns `true` if the image is successfully written; `false` otherwise.

Discussion
You must call this function or the output of the image destination will not be valid. After calling this function, no additional data can be added to the image destination.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGImageDestination.h`

CGImageDestinationGetTypeID

Returns the unique type identifier of an image destination opaque type.

```
CFTypeID CGImageDestinationGetTypeID (
    void
);
```

Return Value
Returns the Core Foundation type ID for an image destination.

Discussion
A type identifier is an integer that identifies the opaque type to which a Core Foundation object belongs. You use type IDs in various contexts, such as when you are operating on heterogeneous collections.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGImageDestination.h`

CGImageDestinationSetProperties

Applies one or more properties to all images in an image destination.

```
void CGImageDestinationSetProperties (
    CGImageDestinationRef idst,
    CFDictionaryRef properties
);
```

Parameters

idst

An image destination.

properties
A dictionary that contains the properties to apply. You can set any of the properties described in "Destination Properties" (page 231) or the image properties described in *CGImageProperties Reference*.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGImageDestination.h

Data Types

CGImageDestinationRef

An opaque type that represents an image destination.

```
typedef struct CGImageDestination *CGImageDestinationRef;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGImageDestination.h

Constants

Destination Properties

Properties for a single image in an image destination.

```
const CFStringRef kCGImageDestinationLossyCompressionQuality
const CFStringRef kCGImageDestinationBackgroundColor
```

Constants
kCGImageDestinationLossyCompressionQuality
The desired compression quality to use when writing to an image destination. If present, the value associated with this key must be a CFNumberRef data type in the range 0.0 to 1.0. A value of 1.0 specifies to use lossless compression if destination format supports it. A value of 0.0 implies to use maximum compression.

Available in Mac OS X v10.4 and later.

Declared in CGImageDestination.h.

`kCGImageDestinationBackgroundColor`

The desired background color to composite against when writing an image that has an alpha component to a destination format that does not support alpha. If present, the value associated with this key must be a `CGColorRef` (page 35) data type without an alpha component of its own. If not present, and if a background color is needed, a white color is used.

Available in Mac OS X v10.4 and later.

Declared in `CGImageDestination.h`.

Declared In
`CGImageDestination.h`

CGImageSource Reference

Derived From:	CFType
Framework:	ApplicationServices/ImageIO
Declared in	CGImageSource.h
Companion guides	Quartz 2D Programming Guide CGImage Reference

Overview

CGImageSource objects, available in Mac OS X v10.4 or later, abstract the data-reading task. An image source can read image data from a URL, a `CFData` object, or a data consumer.

After creating a CGImageSource object for the appropriate source, you can obtain images, thumbnails, image properties, and other image information using CGImageSource functions.

Functions by Task

Creating an Image Source

`CGImageSourceCreateWithDataProvider` (page 238)
Creates an image source that reads data from the specified data provider.

`CGImageSourceCreateWithData` (page 238)
Creates an image source that reads from a Core Foundation data object.

`CGImageSourceCreateWithURL` (page 239)
Creates an image source that reads from a location specified by a URL.

Creating Images From an Image Source

`CGImageSourceCreateImageAtIndex` (page 236)
Creates a CGImage object for the image data associated with the specified index in an image source.

`CGImageSourceCreateThumbnailAtIndex` (page 237)
Creates a thumbnail image of the image located at a specified location in an image source.

`CGImageSourceCreateIncremental` (page 236)
Create an incremental image source.

Updating an Image Source

`CGImageSourceUpdateData` (page 242)
Updates an incremental image source with new data.

`CGImageSourceUpdateDataProvider` (page 242)
Updates an incremental image source with a new data provider.

Getting Information From an Image Source

`CGImageSourceGetTypeID` (page 241)
Returns the unique type identifier of an image source opaque type.

`CGImageSourceGetType` (page 241)
Returns the uniform type identifier of the source container.

`CGImageSourceCopyTypeIdentifiers` (page 235)
Returns an array of uniform type identifiers (UTIs) that are supported for image sources.

`CGImageSourceGetCount` (page 239)
Returns the number of images (not including thumbnails) in the image source.

`CGImageSourceCopyProperties` (page 234)
Returns the properties of the image source.

`CGImageSourceCopyPropertiesAtIndex` (page 235)
Returns the properties of the image at a specified location in an image source.

`CGImageSourceGetStatus` (page 240)
Return the status of an image source.

`CGImageSourceGetStatusAtIndex` (page 240)
Returns the current status of an image that is at a specified location in an image source.

Functions

CGImageSourceCopyProperties

Returns the properties of the image source.

```
CFDictionaryRef CGImageSourceCopyProperties (
   CGImageSourceRef isrc,
   CFDictionaryRef options
);
```

Parameters

isrc
An image source.

options
A dictionary you can use to request additional options. See "Image Source Option Dictionary Keys" (page 244) for the keys you can supply.

Return Value
A dictionary that contains the properties associated with the image source container. See *CGImageProperties Reference* for a list of properties that can be in the dictionary.

Discussion
These properties apply to the container in general but not necessarily to any individual image contained in the image source.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGImageSource.h`

CGImageSourceCopyPropertiesAtIndex

Returns the properties of the image at a specified location in an image source.

```
CFDictionaryRef CGImageSourceCopyPropertiesAtIndex (
   CGImageSourceRef isrc,
   size_t index,
   CFDictionaryRef options
);
```

Parameters

isrc
An image source.

index
The index of the image whose properties you want to obtain. The index is zero-based.

options
A dictionary you can use to request additional options. See "Image Source Option Dictionary Keys" (page 244) for the keys you can supply.

Return Value
A dictionary that contains the properties associated with the image. See *CGImageProperties Reference* for a list of properties that can be in the dictionary.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGImageSource.h`

CGImageSourceCopyTypeIdentifiers

Returns an array of uniform type identifiers (UTIs) that are supported for image sources.

```
CFArrayRef CGImageSourceCopyTypeIdentifiers (
   void
);
```

Return Value
Returns an array of the UTIs that are supported for image sources.

Discussion
See Uniform Type Identifiers Overview for a list of system-declared and third-party UTIs.

Availability
Available in Mac OS X version 10.4 and later.

Related Sample Code
CarbonCocoa_PictureCursor

Declared In
`CGImageSource.h`

CGImageSourceCreateImageAtIndex

Creates a CGImage object for the image data associated with the specified index in an image source.

```
CGImageRef CGImageSourceCreateImageAtIndex (
   CGImageSourceRef isrc,
   size_t index,
   CFDictionaryRef options
);
```

Parameters

isrc

An image source.

index

The index that specifies the location of the image. The index is zero-based.

options

A dictionary that specifies additional creation options. See "`Image Source Option Dictionary Keys`" (page 244) for the keys you can supply.

Return Value
Returns a CGImage object. You are responsible for releasing this object using `CGImageRelease` (page 218).

Availability
Available in Mac OS X version 10.4 and later.

Related Sample Code
CarbonCocoa_PictureCursor

Declared In
`CGImageSource.h`

CGImageSourceCreateIncremental

Create an incremental image source.

```
CGImageSourceRef CGImageSourceCreateIncremental (
   CFDictionaryRef options
);
```

Parameters

options

A dictionary that specifies additional creation options. See "Image Source Option Dictionary Keys" (page 244) for the keys you can supply.

Return Value

Returns an image source object. You are responsible for releasing this object using `CFRelease`.

Discussion

The function `CGImageSourceCreateIncremental` creates an empty image source container to which you can add data later by calling the functions `CGImageSourceUpdateDataProvider` or `CGImageSourceUpdateData`. You don't provide data when you call this function.

An incremental image is an image that is created in chunks, similar to the way large images viewed over the web are loaded piece by piece.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGImageSource.h`

CGImageSourceCreateThumbnailAtIndex

Creates a thumbnail image of the image located at a specified location in an image source.

```
CGImageRef CGImageSourceCreateThumbnailAtIndex (
   CGImageSourceRef isrc,
   size_t index,
   CFDictionaryRef options
);
```

Parameters

isrc

An image source.

index

The index that specifies the location of the image. The index is zero-based.

options

A dictionary that specifies additional creation options. See "Image Source Option Dictionary Keys" (page 244) for the keys you can supply.

Return Value

A CGImage object. You are responsible for releasing this object using `CGImageRelease` (page 218).

Discussion

If the image source is a PDF, this function creates a 72 dpi image of the PDF page specified by the index that you pass. You must, however, pass an options dictionary that contains either the `kCGImageSourceCreateThumbnailFromImageIfAbsent` or `kCGImageSourceCreateThumbnailFromImageAlways` keys, with the value of the key set to `TRUE`.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGImageSource.h`

CGImageSourceCreateWithData

Creates an image source that reads from a Core Foundation data object.

```
CGImageSourceRef CGImageSourceCreateWithData (
   CFDataRef data,
   CFDictionaryRef options
);
```

Parameters

data

The data object to read from. For more information on data objects, see *CFData Reference* and Data Objects.

options

A dictionary that specifies additional creation options. See "Image Source Option Dictionary Keys" (page 244) for the keys you can supply.

Return Value
An image source. You are responsible for releasing this object using `CFRelease`.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGImageSource.h`

CGImageSourceCreateWithDataProvider

Creates an image source that reads data from the specified data provider.

```
CGImageSourceRef CGImageSourceCreateWithDataProvider (
   CGDataProviderRef provider,
   CFDictionaryRef options
);
```

Parameters

provider

The data provider to read from. For more information on data providers, see *CGDataProvider Reference* and *Quartz 2D Programming Guide*.

options

A dictionary that specifies additional creation options. See "Image Source Option Dictionary Keys" (page 244) for the keys you can supply.

Return Value
An image source. You are responsible for releasing this object using `CFRelease`.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGImageSource.h`

CGImageSourceCreateWithURL

Creates an image source that reads from a location specified by a URL.

```
CGImageSourceRef CGImageSourceCreateWithURL (
   CFURLRef url,
   CFDictionaryRef options
);
```

Parameters

url

The URL to read from.

options

A dictionary that specifies additional creation options. See "`Image Source Option Dictionary Keys`" (page 244) for the keys you can supply.

Return Value
An image source. You are responsible for releasing this object using `CFRelease`.

Availability
Available in Mac OS X version 10.4 and later.

Related Sample Code
CarbonCocoa_PictureCursor

Declared In
`CGImageSource.h`

CGImageSourceGetCount

Returns the number of images (not including thumbnails) in the image source.

```
size_t CGImageSourceGetCount (
   CGImageSourceRef isrc
);
```

Parameters

isrc

An image source.

Return Value
The number of images. If the image source is a multilayered PSD file, the function returns `1`.

Discussion
This function does not extract the layers of a PSD file.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGImageSource.h

CGImageSourceGetStatus

Return the status of an image source.

```
CGImageSourceStatus CGImageSourceGetStatus (
   CGImageSourceRef isrc
);
```

Parameters

isrc
An image source.

Return Value
Returns the current status of the image source. See "Image Source Status" (page 243) for a list of possible values.

Discussion
The status is particularly informative for incremental image sources, but may also be used by clients that provide non-incremental data.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGImageSource.h

CGImageSourceGetStatusAtIndex

Returns the current status of an image that is at a specified location in an image source.

```
CGImageSourceStatus CGImageSourceGetStatusAtIndex (
   CGImageSourceRef isrc,
   size_t index
);
```

Parameters

isrc
An image source.

index
The index of the image whose status you want to obtain. The index is zero-based.

Return Value
Returns the current status of the image. See "Image Source Status" (page 243) for a list of possible values.

Discussion
The status is particularly informative for incremental image sources, but may also be used by clients that provide non-incremental data.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGImageSource.h

CGImageSourceGetType

Returns the uniform type identifier of the source container.

```
CFStringRef CGImageSourceGetType (
   CGImageSourceRef isrc
);
```

Parameters

isrc
An image source.

Return Value
The uniform type identifier of the image.

Discussion
The uniform type identifier (UTI) of the source container can be different from the type of the images in the container. For example, the `.icns` format supports embedded `JPEG2000`. The type of the source container is `"com.apple.icns"` but type of the images is `JPEG2000`.

See Uniform Type Identifier Concepts for a list of system-declared and third-party UTIs.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGImageSource.h

CGImageSourceGetTypeID

Returns the unique type identifier of an image source opaque type.

```
CFTypeID CGImageSourceGetTypeID (
   void
);
```

Return Value
Returns the Core Foundation type ID for an image source.

Discussion
A type identifier is an integer that identifies the opaque type to which a Core Foundation object belongs. You use type IDs in various contexts, such as when you are operating on heterogeneous collections. Note that a CFType ID is different from a uniform type identifier (UTI).

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGImageSource.h

CGImageSourceUpdateData

Updates an incremental image source with new data.

```
void CGImageSourceUpdateData (
   CGImageSourceRef isrc,
   CFDataRef data,
   bool final
);
```

Parameters

isrc

An image source.

data

The data to add to the image source. Each time you call the function `CGImageSourceUpdateData`, the `data` parameter must contain all of the image file data accumulated so far.

final

A value that specifies whether the data is the final set. Pass `true` if it is, `false` otherwise.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGImageSource.h`

CGImageSourceUpdateDataProvider

Updates an incremental image source with a new data provider.

```
void CGImageSourceUpdateDataProvider (
   CGImageSourceRef isrc,
   CGDataProviderRef provider,
   bool final
);
```

Parameters

isrc

An image source.

provider

The new data provider. The new data provider must provide all the previous data supplied to the image source plus any additional new data.

final

A value that specifies whether the data is the final set. Pass `true` if it is, `false` otherwise.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGImageSource.h`

Data Types

CGImageSourceRef

An opaque type that represents an image source.

```
typedef struct CGImageSource *CGImageSourceRef;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGImageSource.h`

Constants

Image Source Status

Status states for images and image sources.

```
enum CGImageSourceStatus {
    kCGImageStatusUnexpectedEOF = -5,
    kCGImageStatusInvalidData = -4,
    kCGImageStatusUnknownType = -3,
    kCGImageStatusReadingHeader = -2,
    kCGImageStatusIncomplete = -1,
    kCGImageStatusComplete = 0
};
typedef enum CGImageSourceStatus CGImageSourceStatus;
```

Constants

`kCGImageStatusUnexpectedEOF`
The end of the file was encountered unexpectedly.
Available in Mac OS X v10.4 and later.
Declared in `CGImageSource.h`.

`kCGImageStatusInvalidData`
The data is not valid.
Available in Mac OS X v10.4 and later.
Declared in `CGImageSource.h`.

`kCGImageStatusUnknownType`
The image is an unknown type.
Available in Mac OS X v10.4 and later.
Declared in `CGImageSource.h`.

`kCGImageStatusReadingHeader`
In the process of reading the header.

Available in Mac OS X v10.4 and later.

Declared in `CGImageSource.h`.

`kCGImageStatusIncomplete`
The operation is not complete

Available in Mac OS X v10.4 and later.

Declared in `CGImageSource.h`.

`kCGImageStatusComplete`
The operation is complete.

Available in Mac OS X v10.4 and later.

Declared in `CGImageSource.h`.

Discussion
These status values are returned by the functions `CGImageSourceGetStatus` (page 240) and `CGImageSourceGetStatusAtIndex` (page 240).

Declared In
`CGImageSource.h`

Image Source Option Dictionary Keys

Keys that you can include in the options dictionary to create an image source.

```
CFStringRef kCGImageSourceTypeIdentifierHint;
CFStringRef kCGImageSourceShouldAllowFloat;
CFStringRef kCGImageSourceShouldCache;
CFStringRef kCGImageSourceCreateThumbnailFromImageIfAbsent;
CFStringRef kCGImageSourceCreateThumbnailFromImageAlways;
CFStringRef kCGImageSourceThumbnailMaxPixelSize;
CFStringRef kCGImageSourceCreateThumbnailWithTransform
```

Constants
`kCGImageSourceTypeIdentifierHint`
The best guess of the uniform type identifier (UTI) for the format of the image source file. If specified, the value of this key must be a CFString object. This key can be provided in the options dictionary when you create a CGImageSource object.

Available in Mac OS X v10.4 and later.

Declared in `CGImageSource.h`.

`kCGImageSourceShouldAllowFloat`
Whether the image should be returned as a CGImage object that uses floating-point values, if supported by the file format. CGImage objects that use extended-range floating-point values may require additional processing to render in a pleasing manner. The value of this key must be a CFBoolean value. The default value is `kCFBooleanFalse`.

Available in Mac OS X v10.4 and later.

Declared in `CGImageSource.h`.

`kCGImageSourceShouldCache`
: Whether the image should be cached in a decoded form. The value of this key must be a CFBoolean value. The default value is `kCFBooleanTrue`. This key can be provided in the options dictionary that you can pass to the functions `CGImageSourceCopyPropertiesAtIndex` (page 235) and `CGImageSourceCreateImageAtIndex` (page 236).

  Available in Mac OS X v10.4 and later.

  Declared in `CGImageSource.h`.

`kCGImageSourceCreateThumbnailFromImageIfAbsent`
: Whether a thumbnail should be automatically created for an image if a thumbnail isn't present in the image source file. The thumbnail is created from the full image, subject to the limit specified by `kCGImageSourceThumbnailMaxPixelSize`. If a maximum pixel size isn't specified, then the thumbnail is the size of the full image, which in most cases is not desirable. This key must be a CFBoolean value. The default value is `kCFBooleanFalse`. This key can be provided in the options dictionary that you pass to the function `CGImageSourceCreateThumbnailAtIndex` (page 237).

  Available in Mac OS X v10.4 and later.

  Declared in `CGImageSource.h`.

`kCGImageSourceCreateThumbnailFromImageAlways`
: Whether a thumbnail should be created from the full image even if a thumbnail is present in the image source file. The thumbnail is created from the full image, subject to the limit specified by `kCGImageSourceThumbnailMaxPixelSize`. If a maximum pixel size isn't specified, then the thumbnail is the size of the full image, which probably isn't what you want. This key must be a CFBoolean value. The default value is `kCFBooleanFalse`. This key can be provided in the options dictionary that you can pass to the function `CGImageSourceCreateThumbnailAtIndex` (page 237).

  Available in Mac OS X v10.4 and later.

  Declared in `CGImageSource.h`.

`kCGImageSourceThumbnailMaxPixelSize`
: The maximum width and height in pixels of a thumbnail. If this key is not specified, the width and height of a thumbnail is not limited and thumbnails may be as big as the image itself. If present, this key must be a CFNumber value. This key can be provided in the options dictionary that you pass to the function `CGImageSourceCreateThumbnailAtIndex` (page 237).

  Available in Mac OS X v10.4 and later.

  Declared in `CGImageSource.h`.

`kCGImageSourceCreateThumbnailWithTransform`
: Whether the thumbnail should be rotated and scaled according to the orientation and pixel aspect ratio of the full image. The value of this key must be a CFBoolean value. The default value is `kCFBooleanFalse`.

  Available in Mac OS X v10.4 and later.

  Declared in `CGImageSource.h`.

Discussion

Except for `kCGImageSourceTypeIdentifierHint`, which you use when creating an image source, these constants specify options that you can set when creating an image from image source. Each constant is a key; you must supply the appropriate value when you add this option to the options dictionary.

Declared In

`CGImageSource.h`

CGLayer Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGLayer.h

Overview

CGLayer objects are useful for offscreen drawing and can be used in much the same way that a bitmap context can be used. In fact, a CGLayer object is a much better representation than a bitmap context.

Using CGLayer objects can improve performance, particularly when you need to capture a piece of drawing that you stamp repeatedly (using the same scale factor and orientation). Quartz can cache CGLayer objects to the video card, making drawing a CGLayer to a destination much faster than rendering the equivalent image constructed from a bitmap context.

A CGLayer object is created relative to a graphics context. Although layer uses this graphics context as a reference for initialization, you are not restricted to drawing the layer to this graphics context. You can draw the layer to other graphics contexts, although any limitations of the original context are imposed. For example, if you create a CGLayer object using a bitmap context, the layer is rendered as a bitmap when drawn to any other graphics context.

You can use a CGLayer when you want to apply a shadow to a group of objects (such as a group of circles) rather than to individual objects.

Use these layers in your code whenever you can, especially when:

- You need to reuse a filled or stroked shape.
- You are building a scene and at least some of it can be reused. Put the reusable drawing in its own CGLayer.

Any CG object that you draw repeatedly—including CGPath, CGShading, and CGPDFPage—benefit from improved performance if you draw it to a CGLayer object.

Functions by Task

Creating Layer Objects

`CGLayerCreateWithContext` (page 249)
Creates a CGLayer object that is associated with a graphics context.

Drawing Layer Content

`CGContextDrawLayerInRect` (page 249)
Draws the contents of a CGLayer object into the specified rectangle.

`CGContextDrawLayerAtPoint` (page 248)
Draws the contents of a CGLayer object at the specified point.

Retaining and Releasing Layers

`CGLayerRelease` (page 251)
Decrements the retain count of a CGLayer object.

`CGLayerRetain` (page 252)
Increments the retain count of a CGLayer object.

Getting the CFType ID for a Layer

`CGLayerGetTypeID` (page 251)
Returns the unique type identifier used for CGLayer objects.

Getting Layer Information

`CGLayerGetSize` (page 251)
Returns the width and height of a CGLayer object.

`CGLayerGetContext` (page 250)
Returns the graphics context associated with a CGLayer object.

Functions

CGContextDrawLayerAtPoint

Draws the contents of a CGLayer object at the specified point.

```
void CGContextDrawLayerAtPoint (
   CGContextRef context,
   CGPoint point,
   CGLayerRef layer
);
```

Parameters

context
The graphics context associated with the layer.

point
The location, in current user space coordinates, to use as the origin for the drawing.

layer
The layer whose contents you want to draw.

Discussion
Calling the function `CGContextDrawLayerAtPoint` is equivalent to calling the function `CGContextDrawLayerInRect` with a rectangle that has its origin at `point` and its size equal to the size of the layer.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGLayer.h`

CGContextDrawLayerInRect

Draws the contents of a CGLayer object into the specified rectangle.

```
void CGContextDrawLayerInRect (
   CGContextRef context,
   CGRect rect,
   CGLayerRef layer
);
```

Parameters
context
The graphics context associated with the layer.

rect
The rectangle, in current user space coordinates, to draw to.

layer
The layer whose contents you want to draw.

Discussion
The contents are scaled, if necessary, to fit into the rectangle.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGLayer.h`

CGLayerCreateWithContext

Creates a CGLayer object that is associated with a graphics context.

```
CGLayerRef CGLayerCreateWithContext (
    CGContextRef context,
    CGSize size,
    CFDictionaryRef auxiliaryInfo
);
```

Parameters

context

The graphics context you want to create the layer relative to. The layer uses this graphics context as a reference for initialization.

size

The size, in default user space units, of the layer relative to the graphics context.

auxiliaryInfo

Reserved for future use. Pass `NULL`.

Return Value

A CGLayer object. You are responsible for releasing this object using the function `CGLayerRelease` (page 251) when you no longer need the layer.

Discussion

After you create a CGLayer object, you should reuse it whenever you can to facilitate the Quartz caching strategy. Quartz caches any objects that are reused, including CGLayer objects. Objects that are reused frequently remain in the cache. In contrast, objects that are used once in a while may be moved in and out of the cache according to their frequency of use. If you don't reuse CGLayer objects, Quartz won't cache them. This means that you lose an opportunity to improve the performance of your application.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGLayer.h`

CGLayerGetContext

Returns the graphics context associated with a CGLayer object.

```
CGContextRef CGLayerGetContext (
    CGLayerRef layer
);
```

Parameters

layer

The layer whose graphics context you want to obtain.

Return Value

The graphics context associated with the layer.

Discussion

The context that's returned is the context for the layer itself, not the context that you specified when you created the layer.

Availability

Available in Mac OS X version 10.4 and later.

Declared In
CGLayer.h

CGLayerGetSize

Returns the width and height of a CGLayer object.

```
CGSize CGLayerGetSize (
    CGLayerRef layer
);
```

Parameters

layer
The layer whose width and height you want to obtain.

Return Value
The width and height of the layer, in default user space coordinates.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGLayer.h

CGLayerGetTypeID

Returns the unique type identifier used for CGLayer objects.

```
CFTypeID CGLayerGetTypeID (
    void
);
```

Return Value
The type identifier for CGLayer objects.

Discussion
A type identifier is an integer that identifies the opaque type to which a Core Foundation object belongs. You use type IDs in various contexts, such as when you are operating on heterogeneous collections.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGLayer.h

CGLayerRelease

Decrements the retain count of a CGLayer object.

```
void CGLayerRelease (
    CGLayerRef layer
);
```

Parameters

layer

The layer to release.

Discussion

This function is equivalent to calling `CFRelease (layer)` except that it does not crash (as `CFRetain` does) if the `layer` parameter is `null`.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGLayer.h`

CGLayerRetain

Increments the retain count of a CGLayer object.

```
CGLayerRef CGLayerRetain (
    CGLayerRef layer
);
```

Parameters

layer

The layer to retain.

Return Value

The same layer you passed in as the `layer` parameter.

Discussion

This function is equivalent to calling `CFRetain (layer)` except that it does not crash (as `CFRetain` does) if the `layer` parameter is `null`.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGLayer.h`

Data Types

CGLayerRef

An opaque type used for offscreen drawing.

```
typedef struct CGLayer *CGLayerRef;
```

Availability

Available in Mac OS X v10.4 and later.

Declared In
`CGLayer.h`

CGPath Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPath.h
Companion guide	Quartz 2D Programming Guide

Overview

A **graphics path** is a description of a 2D geometric scene using sequences of lines and Bézier curves. `CGPathRef` defines an opaque type that represents an immutable graphics path. `CGMutablePathRef` defines an opaque type that represents a mutable graphics path. To draw using a Quartz path, you need to add the path to a graphics context—see `CGContextAddPath` (page 67).

Each figure in a scene may be described by a **subpath**. A subpath has an ordered set of **path elements**, that represent single steps in the construction of a subpath. (For example, `MoveToPoint (bottom left)` and `AddLineToPoint (bottom right)` are path elements.) A subpath also maintains state information, including a **starting point** and a **current point**. When drawing a path, Quartz traverses each subpath using its path elements and its state.

The lines and curves in a subpath are always connected, but they do not necessarily form a closed figure. Furthermore, subpaths do not need to be connected to each other. For example, you could use a graphics path to draw the outlines of a sequence of text characters.

Functions by Task

Creating and Managing Paths

`CGPathCreateMutable` (page 267)
Creates a mutable graphics path.

`CGPathCreateMutableCopy` (page 267)
Creates a mutable copy of an existing graphics path.

`CGPathCreateCopy` (page 266)
Creates an immutable copy of a graphics path.

`CGPathRelease` (page 271)
Decrements the retain count of a graphics path.

`CGPathRetain` (page 271)
Increments the retain count of a graphics path.

Modifying Quartz Paths

`CGPathAddArc` (page 257)
Appends an arc to a mutable graphics path, possibly preceded by a straight line segment.

`CGPathAddArcToPoint` (page 258)
Appends an arc to a mutable graphics path, possibly preceded by a straight line segment.

`CGPathAddCurveToPoint` (page 259)
Appends a Bézier curve to a mutable graphics path.

`CGPathAddLines` (page 261)
Appends an array of new line segments to a mutable graphics path.

`CGPathAddLineToPoint` (page 261)
Appends a line segment to a mutable graphics path.

`CGPathAddPath` (page 262)
Appends a path to a mutable graphics path.

`CGPathAddQuadCurveToPoint` (page 262)
Appends a quadratic curve to a mutable graphics path.

`CGPathAddRect` (page 263)
Appends a rectangle to a mutable graphics path.

`CGPathAddRects` (page 264)
Appends an array of rectangles to a mutable graphics path.

`CGPathApply` (page 265)
For each element in a graphics path, calls a custom applier function.

`CGPathMoveToPoint` (page 270)
Starts a new subpath at a specified location in a mutable graphics path.

`CGPathCloseSubpath` (page 265)
Closes and completes a subpath in a mutable graphics path.

`CGPathAddEllipseInRect` (page 260)
Adds to a path an ellipse that fits inside a rectangle.

Getting Information about Quartz Paths

`CGPathEqualToPath` (page 268)
Indicates whether two graphics paths are equivalent.

`CGPathGetBoundingBox` (page 268)
Returns the bounding box of a graphics path.

`CGPathGetCurrentPoint` (page 268)
Returns the current point in a graphics path.

`CGPathGetTypeID` (page 269)
Returns the Core Foundation type identifier for Quartz graphics paths.

`CGPathIsEmpty` (page 269)
Indicates whether or not a graphics path is empty.

`CGPathIsRect` (page 270)
Indicates whether or not a graphics path represents a rectangle.

`CGPathContainsPoint` (page 266)
Checks whether a point is contained in a graphics path.

Functions

CGPathAddArc

Appends an arc to a mutable graphics path, possibly preceded by a straight line segment.

```
void CGPathAddArc (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   CGFloat x,
   CGFloat y,
   CGFloat radius,
   CGFloat startAngle,
   CGFloat endAngle,
   bool clockwise
);
```

Parameters

path
The mutable graphics path to change.

m
A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the arc before it is added to the path.

x
The x-coordinate of the center point of the arc.

y
The y-coordinate of the center point of the arc.

r
The radius of the arc.

startAngle
The angle (in radians) from horizontal that determines the starting point of the arc.

endAngle
The angle (in radians) from horizontal that determines the ending point of the arc.

clockwise
A Boolean value that specifies whether or not to draw the arc in the clockwise direction; `true` specifies clockwise.

Discussion
An arc is a segment of a circle with radius r centered at a point `(x,y)`. When you call this function, you provide the center point, radius, and two angles in radians. Quartz uses this information to determine the end points of the arc, and then approximates the new arc using a sequence of cubic Bézier curves. The `clockwise` parameter determines the direction in which the arc is drawn.

A transformation may be applied to the Bézier curves before they are added to the path. If no transform is needed, the second argument should be `NULL`.

If the specified path already contains a subpath, Quartz implicitly adds a line connecting the current point to the beginning of the arc. If the path is empty, Quartz creates a new subpath for the arc and does not add the initial straight line segment.

The ending point of the arc becomes the new current point of the path.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPath.h`

CGPathAddArcToPoint

Appends an arc to a mutable graphics path, possibly preceded by a straight line segment.

```
void CGPathAddArcToPoint (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   CGFloat x1,
   CGFloat y1,
   CGFloat x2,
   CGFloat y2,
   CGFloat radius
);
```

Parameters

path

The mutable path to change. The path must not be empty.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the arc before it is added to the path.

x1

The x-coordinate of the user space for the end point of the first tangent line. The first tangent line is drawn from the current point to `(x1,y1)`.

y1

The y-coordinate of the user space for the end point of the first tangent line. The first tangent line is drawn from the current point to `(x1,y1)`.

x2

The x-coordinate of the user space for the end point of the second tangent line. The second tangent line is drawn from `(x1,y1)` to `(x2,y2)`.

y2

The y-coordinate of the user space for the end point of the second tangent line. The second tangent line is drawn from `(x1,y1)` to `(x2,y2)`.

radius

The radius of the arc, in user space coordinates.

Discussion

This function uses a sequence of cubic Bézier curves to draw an arc that is tangent to the line from the current point to (x1,y1) and to the line from (x1,y1) to (x2,y2). The start and end points of the arc are located on the first and second tangent lines, respectively. The start and end points of the arc are also the "tangent points" of the lines.

If the current point and the first tangent point of the arc (the starting point) are not equal, Quartz appends a straight line segment from the current point to the first tangent point. After adding the arc, the current point is reset to the end point of the arc (the second tangent point).

For another way to draw an arc in a path, see `CGPathAddArc` (page 257).

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGPath.h`

CGPathAddCurveToPoint

Appends a Bézier curve to a mutable graphics path.

```
void CGPathAddCurveToPoint (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   CGFloat cp1x,
   CGFloat cp1y,
   CGFloat cp2x,
   CGFloat cp2y,
   CGFloat x,
   CGFloat y
);
```

Parameters

path

The mutable path to change. The path must not be empty.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the curve before it is added to the path.

cx1

The x-coordinate of the first control point.

cy1

The y-coordinate of the first control point.

cx2

The x-coordinate of the second control point.

cy2
The y-coordinate of the second control point.

x
The x-coordinate of the end point of the curve.

y
The y-coordinate of the end point of the curve.

Discussion
Appends a cubic Bézier curve from the current point in a path to the specified location using two control points, after an optional transformation. Before returning, this function updates the current point to the specified location `(x,y)`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPath.h`

CGPathAddEllipseInRect

Adds to a path an ellipse that fits inside a rectangle.

```
void CGPathAddEllipseInRect (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   CGRect rect
);
```

Parameters

path
The path to modify.

m
An affine transform to apply to the ellipse, or `NULL` if you don't want to transform the ellipse.

rect
A rectangle to enclose the ellipse.

Discussion
The ellipse is approximated by a sequence of Bézier curves. Its center is the midpoint of the rectangle defined by the `rect` parameter. If the rectangle is square, then the ellipse is circular with a radius equal to one-half the width (or height) of the rectangle. If the `rect` parameter specifies a rectangular shape, then the major and minor axes of the ellipse are defined by the width and height of the rectangle.

The ellipse forms a complete subpath of the path—that is, the ellipse drawing starts with a move-to operation and ends with a close-subpath operation, with all moves oriented in the clockwise direction. If you supply an affine transform, then the constructed Bézier curves that define the ellipse are transformed before they are added to the path.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPath.h`

CGPathAddLines

Appends an array of new line segments to a mutable graphics path.

```
void CGPathAddLines (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   const CGPoint points[],
   size_t count
);
```

Parameters

path

The mutable path to change.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the lines before adding them to the path.

points

An array of points that specifies the line segments to add.

count

The number of elements in the array.

Discussion

This is a convenience function that adds a sequence of connected line segments to a path, using the following operation:

```
CGPathMoveToPoint (path, m, points[0].x, points[0].y);
for (k = 1; k < count; k++) {
    CGPathAddLineToPoint (path, m, points[k].x, points[k].y);
}
```

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGPath.h`

CGPathAddLineToPoint

Appends a line segment to a mutable graphics path.

```
void CGPathAddLineToPoint (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   CGFloat x,
   CGFloat y
);
```

Parameters

path

The mutable path to change. The path must not be empty.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the line before it is added to the path.

x
The x-coordinate of the end point of the line.

y
The y-coordinate of the end point of the line.

Discussion
Before returning, this function updates the current point to the specified location `(x,y)`.

Availability
Available in Mac OS X version 10.2 and later.

Related Sample Code
CALayerEssentials

Declared In
`CGPath.h`

CGPathAddPath

Appends a path to a mutable graphics path.

```
void CGPathAddPath (
   CGMutablePathRef path1,
   const CGAffineTransform *m,
   CGPathRef path2
);
```

Parameters

path1
The mutable path to change.

m
A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to `path2` before it is added to `path1`.

path2
The path to add.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPath.h`

CGPathAddQuadCurveToPoint

Appends a quadratic curve to a mutable graphics path.

```
void CGPathAddQuadCurveToPoint (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   CGFloat cpx,
   CGFloat cpy,
   CGFloat x,
   CGFloat y
);
```

Parameters

path

The mutable path to change. The path must not be empty.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the curve before adding it to the path.

cx

The x-coordinate of the control point.

cy

The y-coordinate of the control point.

x

The x-coordinate of the end point of the curve.

y

The y-coordinate of the end point of the curve.

Discussion

Before returning, this function updates the current point to the specified location `(x, y)`.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGPath.h`

CGPathAddRect

Appends a rectangle to a mutable graphics path.

```
void CGPathAddRect (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   CGRect rect
);
```

Parameters

path

The mutable path to change.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the rectangle before adding it to the path.

rect

The rectangle to add.

Discussion
This is a convenience function that adds a rectangle to a path, using the following sequence of operations:

```
// start at origin
CGPathMoveToPoint (path, m, CGRectGetMinX(rect), CGRectGetMinY(rect));

// add bottom edge
CGPathAddLineToPoint (path, m, CGRectGetMaxX(rect), CGRectGetMinY(rect));

// add right edge
CGPathAddLineToPoint (path, m, CGRectGetMaxX(rect), CGRectGetMaxY(rect);

// add top edge
CGPathAddLineToPoint (path, m, CGRectGetMinX(rect), CGRectGetMaxY(rect));

// add left edge and close
CGPathCloseSubpath (path);
```

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPath.h`

CGPathAddRects

Appends an array of rectangles to a mutable graphics path.

```
void CGPathAddRects (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   const CGRect rects[],
   size_t count
);
```

Parameters

path
The mutable path to change.

m
An affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the rectangles before adding them to the path.

rects
The array of new rectangles to add.

count
The number of elements in the array.

Discussion
This is a convenience function that adds an array of rectangles to a path, using the following operation:

```
for (k = 0; k < count; k++) {
    CGPathAddRect (path, m, rects[k]);
}
```

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CGPath.h

CGPathApply

For each element in a graphics path, calls a custom applier function.

```
void CGPathApply (
   CGPathRef path,
   void *info,
   CGPathApplierFunction function
);
```

Parameters

path
The path to which the function will be applied.

info
A pointer to the user data that Quartz will pass to the function being applied, or NULL.

function
A pointer to the function to apply. See CGPathApplierFunction (page 272) for more information.

Discussion
For each element in the specified path, Quartz calls the applier function, which can examine (but not modify) the element.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CGPath.h

CGPathCloseSubpath

Closes and completes a subpath in a mutable graphics path.

```
void CGPathCloseSubpath (
   CGMutablePathRef path
);
```

Parameters

path
The path to change.

Discussion
Appends a line from the current point in a path to the starting point of the current subpath and ends the subpath. On return, the current point is now the previous starting point.

Availability
Available in Mac OS X version 10.2 and later.

Related Sample Code
CALayerEssentials

Declared In
CGPath.h

CGPathContainsPoint

Checks whether a point is contained in a graphics path.

```
bool CGPathContainsPoint (
   CGPathRef path,
   const CGAffineTransform *m,
   CGPoint point,
   bool eoFill
);
```

Parameters

path

The path to evaluate the point against.

m

An affine transform. If `m` is not `NULL` then the point is transformed by this affine transform prior to determining whether the path contains the point.

point

The point to check.

eoFill

A Boolean value that, if `true`, specifies to use the even-odd fill rule to evaluate the painted region of the path. If `false`, the winding fill rule is used.

Return Value
Returns `true` if the point is contained in the path; `false` otherwise.

Discussion
A point is contained in a path if it is inside the painted region when the path is filled and the path is a closed path. You can call the function `CGPathCloseSubpath` to ensure that a path is closed.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGPath.h

CGPathCreateCopy

Creates an immutable copy of a graphics path.

```
CGPathRef CGPathCreateCopy (
   CGPathRef path
);
```

Parameters

path

The path to copy.

Return Value
A new, immutable copy of the specified path. You are responsible for releasing this object.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CGPath.h

CGPathCreateMutable

Creates a mutable graphics path.

```
CGMutablePathRef CGPathCreateMutable (
   void
);
```

Return Value
A new mutable path. You are responsible for releasing this object.

Availability
Available in Mac OS X version 10.2 and later.

Related Sample Code
CALayerEssentials

Declared In
CGPath.h

CGPathCreateMutableCopy

Creates a mutable copy of an existing graphics path.

```
CGMutablePathRef CGPathCreateMutableCopy (
   CGPathRef path
);
```

Parameters

path
The path to copy.

Return Value
A new, mutable, copy of the specified path. You are responsible for releasing this object.

Discussion
You can modify a mutable graphics path by calling the various CGPath geometry functions, such as CGPathAddArc (page 257), CGPathAddLineToPoint (page 261), and CGPathMoveToPoint (page 270).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CGPath.h

CGPathEqualToPath

Indicates whether two graphics paths are equivalent.

```
bool CGPathEqualToPath (
   CGPathRef path1,
   CGPathRef path2
);
```

Parameters

path1

The first path being compared.

path2

The second path being compared.

Return Value

A Boolean value that indicates whether or not the two specified paths contain the same sequence of path elements. If the paths are not the same, returns `false`.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGPath.h`

CGPathGetBoundingBox

Returns the bounding box of a graphics path.

```
CGRect CGPathGetBoundingBox (
   CGPathRef path
);
```

Parameters

path

The graphics path to evaluate.

Return Value

A rectangle that represents the bounding box of the specified path.

Discussion

The bounding box is the smallest rectangle completely enclosing all points in the path, including control points for Bézier and quadratic curves.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGPath.h`

CGPathGetCurrentPoint

Returns the current point in a graphics path.

```
CGPoint CGPathGetCurrentPoint (
   CGPathRef path
);
```

Parameters

path

The path to evaluate.

Return Value
The current point in the specified path.

Discussion
If the path is empty—that is, if it has no elements—this function returns `CGPointZero` (see `CGGeometry`). To determine whether a path is empty, use `CGPathIsEmpty` (page 269).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPath.h`

CGPathGetTypeID

Returns the Core Foundation type identifier for Quartz graphics paths.

```
CFTypeID CGPathGetTypeID (
   void
);
```

Return Value
The Core Foundation identifier for the opaque type `CGPathRef` (page 272).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPath.h`

CGPathIsEmpty

Indicates whether or not a graphics path is empty.

```
bool CGPathIsEmpty (
   CGPathRef path
);
```

Parameters

path

The path to evaluate.

Return Value
A Boolean value that indicates whether the specified path is empty.

Discussion
An empty path contains no elements.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPath.h`

CGPathIsRect

Indicates whether or not a graphics path represents a rectangle.

```
bool CGPathIsRect (
   CGPathRef path,
   CGRect *rect
);
```

Parameters

path

The path to evaluate.

rect

On input, a pointer to an uninitialized rectangle. If the specified path represents a rectangle, on return contains a copy of the rectangle.

Return Value
A Boolean value that indicates whether the specified path represents a rectangle. If the path represents a rectangle, returns `true`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPath.h`

CGPathMoveToPoint

Starts a new subpath at a specified location in a mutable graphics path.

```
void CGPathMoveToPoint (
   CGMutablePathRef path,
   const CGAffineTransform *m,
   CGFloat x,
   CGFloat y
);
```

Parameters

path

The mutable path to change.

m

A pointer to an affine transformation matrix, or `NULL` if no transformation is needed. If specified, Quartz applies the transformation to the point before changing the path.

x

The x-coordinate of the new location.

y
The y-coordinate of the new location.

Discussion
This function initializes the starting point and the current point to the specified location (x,y) after an optional transformation.

Availability
Available in Mac OS X version 10.2 and later.

Related Sample Code
CALayerEssentials

Declared In
`CGPath.h`

CGPathRelease

Decrements the retain count of a graphics path.

```
void CGPathRelease (
   CGPathRef path
);
```

Parameters
path
The graphics path to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the `path` parameter is `NULL`.

Availability
Available in Mac OS X version 10.2 and later.

Related Sample Code
CALayerEssentials

Declared In
`CGPath.h`

CGPathRetain

Increments the retain count of a graphics path.

```
CGPathRef CGPathRetain (
   CGPathRef path
);
```

Parameters
path
The graphics path to retain.

Return Value
The same path you passed in as the `path` parameter.

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `path` parameter is `NULL`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPath.h`

Callbacks

CGPathApplierFunction

Defines a callback function that can view an element in a graphics path.

```
typedef void (*CGPathApplierFunction) (
    void *info,
    const CGPathElement *element
);
```

If you name your function `MyCGPathApplierFunc`, you would declare it like this:

```
void MyCGPathApplierFunc (
   void *info,
  const CGPathElement *element
);
```

Discussion
See also `CGPathApply` (page 265).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGPath.h`

Data Types

CGPathRef

An opaque type that represents an immutable graphics path.

```
typedef const struct CGPath *CGPathRef;
```

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGPath.h

CGMutablePathRef

An opaque type that represents a mutable graphics path.

```
typedef struct CGPath *CGMutablePathRef;
```

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGPath.h

CGPathElement

A data structure that provides information about a path element.

```
struct CGPathElement {
    CGPathElementType type;
    CGPoint * points;
};
typedef struct CGPathElement CGPathElement;
```

Fields
`type`
An element type (or operation).
`points`
An array of one or more points that serve as arguments.

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGPath.h

Constants

Path Drawing Modes

Options for rendering a path.

```
enum CGPathDrawingMode {
    kCGPathFill,
    kCGPathEOFill,
    kCGPathStroke,
    kCGPathFillStroke,
    kCGPathEOFillStroke
};
typedef enum CGPathDrawingMode CGPathDrawingMode;
```

Constants

`kCGPathFill`
Render the area contained within the path using the non-zero winding number rule.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGPathEOFill`
Render the area within the path using the even-odd rule.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGPathStroke`
Render a line along the path.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGPathFillStroke`
First fill and then stroke the path, using the nonzero winding number rule.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

`kCGPathEOFillStroke`
First fill and then stroke the path, using the even-odd rule.
Available in Mac OS X v10.0 and later.
Declared in `CGContext.h`.

Discussion
You can pass a path drawing mode constant to the function `CGContextDrawPath` (page 81) to specify how Quartz should paint a graphics context's current path.

Path Element Types

The type of element found in a path.

```
enum CGPathElementType {
    kCGPathElementMoveToPoint,
    kCGPathElementAddLineToPoint,
    kCGPathElementAddQuadCurveToPoint,
    kCGPathElementAddCurveToPoint,
    kCGPathElementCloseSubpath
};
typedef enum CGPathElementType CGPathElementType;
```

Constants

`kCGPathElementMoveToPoint`
The path element that starts a new subpath. See the function `CGPathMoveToPoint` (page 270).

Available in Mac OS X v10.2 and later.

Declared in `CGPath.h`.

`kCGPathElementAddLineToPoint`
The path element that adds a line from the current point to the specified point. See the function `CGPathAddLineToPoint` (page 261).

Available in Mac OS X v10.2 and later.

Declared in `CGPath.h`.

`kCGPathElementAddQuadCurveToPoint`
The path element that adds a quadratic curve from the current point to the specified point. See the function `CGPathAddQuadCurveToPoint` (page 262).

Available in Mac OS X v10.2 and later.

Declared in `CGPath.h`.

`kCGPathElementAddCurveToPoint`
The path element that adds a cubic curve from the current point to the specified point. See the function `CGPathAddCurveToPoint` (page 259).

Available in Mac OS X v10.2 and later.

Declared in `CGPath.h`.

`kCGPathElementCloseSubpath`
The path element that closes and completes a subpath. See the function `CGPathCloseSubpath` (page 265).

Available in Mac OS X v10.2 and later.

Declared in `CGPath.h`.

Discussion

For more information about paths, see `CGPathRef` (page 272).

CGPattern Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPattern.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPatternRef` opaque type represents a pattern that you can use to stroke along or fill in a graphics path. Quartz tiles the pattern cell for you, based on parameters you specify when you call `CGPatternCreate` (page 278).

To create a dashed line, see `CGContextSetLineDash` (page 111) in *CGContext Reference*.

Functions by Task

Creating a Pattern

`CGPatternCreate` (page 278)
Creates a pattern object.

Getting the CFType ID

`CGPatternGetTypeID` (page 279)
Returns the type identifier for Quartz patterns.

Retaining and Releasing a Pattern

`CGPatternRetain` (page 280)
Increments the retain count of a Quartz pattern.

`CGPatternRelease` (page 279)
Decrements the retain count of a Quartz pattern.

Functions

CGPatternCreate

Creates a pattern object.

```
CGPatternRef CGPatternCreate (
   void *info,
   CGRect bounds,
   CGAffineTransform matrix,
   CGFloat xStep,
   CGFloat yStep,
   CGPatternTiling tiling,
   bool isColored,
   const CGPatternCallbacks *callbacks
);
```

Parameters

info

A pointer to private storage used by your pattern drawing function, or `NULL`. For more information, see the discussion below.

bounds

The bounding box of the pattern cell, specified in pattern space. (Pattern space is an abstract space that maps to the default user space by the transformation matrix you specify with the `matrix` parameter.) The drawing done in your pattern drawing function is clipped to this rectangle.

matrix

A matrix that represents a transform from pattern space to the default user space of the context in which the pattern is used. If no transform is needed, pass the identity matrix.

xStep

The horizontal displacement between cells, specified in pattern space. For no additional horizontal space between cells (so that each pattern cells abuts the previous pattern cell in the horizontal direction), pass the width of the pattern cell.

yStep

The vertical displacement between cells, specified in pattern space. For no additional vertical space between cells (so that each pattern cells abuts the previous pattern cell in the vertical direction), pass the height of the pattern cell.

tiling

A `CGPatternTiling` constant that specifies the desired tiling method. For more information about tiling methods, see "Tiling Patterns" (page 283).

isColored

If you want to draw your pattern using its own intrinsic color, pass `true`. If you want to draw an uncolored (or masking) pattern that uses the fill or stroke color in the graphics state, pass `false`.

callbacks

A pointer to a pattern callback function table—your pattern drawing function is an entry in this table. See `CGPatternCallbacks` (page 282) for more information about callback function tables for patterns.

Return Value

A new Quartz pattern. You are responsible for releasing this object using `CGPatternRelease` (page 279).

Discussion
Quartz calls your drawing function at the appropriate time to draw the pattern cell. A pattern cell must be invariant—that is, the pattern cell should be drawn exactly the same way each time the drawing function is called.

The appearance of a pattern cell is unaffected by changes in the graphics state of the context in which the pattern is used.

See `CGPatternDrawPatternCallback` (page 280) for more information about pattern drawing functions.

Availability
Available in Mac OS X version 10.1 and later.

Declared In
`CGPattern.h`

CGPatternGetTypeID

Returns the type identifier for Quartz patterns.

```
CFTypeID CGPatternGetTypeID (
    void
);
```

Return Value
The identifier for the opaque type `CGPatternRef` (page 282).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPattern.h`

CGPatternRelease

Decrements the retain count of a Quartz pattern.

```
void CGPatternRelease (
   CGPatternRef pattern
);
```

Parameters
pattern
The pattern to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the *pattern* parameter is `NULL`.

Availability
Available in Mac OS X version 10.1 and later.

Declared In
`CGPattern.h`

CGPatternRetain

Increments the retain count of a Quartz pattern.

```
CGPatternRef CGPatternRetain (
   CGPatternRef pattern
);
```

Parameters

pattern

The pattern to retain.

Return Value

The same pattern you passed in as the `pattern` parameter.

Discussion

This function is equivalent to `CFRetain`, except that it does not cause an error if the `pattern` parameter is `NULL`.

Availability

Available in Mac OS X version 10.1 and later.

Declared In

`CGPattern.h`

Callbacks

CGPatternDrawPatternCallback

Draws a pattern cell.

```
typedef void (*CGPatternDrawPatternCallback) (
   void * info,
    CGContextRef context
);
```

If you name your function `MyDrawPattern`, you would declare it like this:

```
void MyDrawPattern (
    void * info,
    CGContextRef context
);
```

Parameters

info

A generic pointer to private data associated with the pattern. This is the same pointer you supplied to `CGPatternCreate` (page 278).

context

The graphics context for drawing the pattern cell.

Discussion
When a pattern is used to stroke or fill a graphics path, Quartz calls your custom drawing function at the appropriate time to draw the pattern cell. The cell should be drawn exactly the same way each time the drawing function is called.

In a drawing function associated with an uncolored pattern, you should not attempt to set a stroke or fill color or color space—if you do so, the result is undefined.

To learn how to associate your drawing function with a Quartz pattern, see `CGPatternCreate` (page 278) and `CGPatternCallbacks` (page 282).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGPattern.h`

CGPatternReleaseInfoCallback

Release private data or resources associated with the pattern.

```
typedef void (*CGPatternReleaseInfoCallback) (
    void * info
);
```

If you name your function `MyCGPatternReleaseInfo`, you would declare it like this:

```
void MyCGPatternReleaseInfo (
    void * info
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGPatternCreate` (page 278).

Discussion
Quartz calls your release function when it frees your pattern object.

To learn how to associate your release function with a Quartz pattern, see `CGPatternCreate` (page 278) and `CGPatternCallbacks` (page 282).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGPattern.h`

Data Types

CGPatternRef

An opaque type that represents a pattern.

```
typedef struct CGPattern * CGPatternRef;
```

Availability
Available in Mac OS X v10.1 and later.

Declared In
`CGPattern.h`

CGPatternCallbacks

A structure that holds a version and two callback functions for drawing a custom pattern.

```
struct CGPatternCallbacks {
    unsigned int version;
    CGPatternDrawPatternCallback drawPattern;
    CGPatternReleaseInfoCallback releaseInfo;
};
typedef struct CGPatternCallbacks CGPatternCallbacks;
```

Fields

`version`
: The version of the structure passed in as a parameter to the `CGPatternCreate` (page 278). For this version of the structure, you should set this value to zero.

`drawPattern`
: A pointer to a custom function that draws the pattern. For information about this callback function, see `CGPatternDrawPatternCallback` (page 280).

`releaseInfo`
: An optional pointer to a custom function that's invoked when the pattern is released. `CGPatternReleaseInfoCallback` (page 281).

Discussion
You supply a `CGPatternCallbacks` structure to the function `CGPatternCreate` (page 278) to create a data provider for direct access. The functions specified by the `CGPatternCallbacks` structure are responsible for drawing the pattern and for handling the pattern's memory management.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`CGPattern.h`

Constants

Tiling Patterns

Different methods for rendering a tiled pattern.

```
enum CGPatternTiling {
    kCGPatternTilingNoDistortion,
    kCGPatternTilingConstantSpacingMinimalDistortion,
    kCGPatternTilingConstantSpacing
};
typedef enum CGPatternTiling CGPatternTiling;
```

Constants

`kCGPatternTilingNoDistortion`

The pattern cell is not distorted when painted. The spacing between pattern cells may vary by as much as 1 device pixel.

Available in Mac OS X v10.1 and later.

Declared in `CGPattern.h`.

`kCGPatternTilingConstantSpacingMinimalDistortion`

Pattern cells are spaced consistently. The pattern cell may be distorted by as much as 1 device pixel when the pattern is painted.

Available in Mac OS X v10.1 and later.

Declared in `CGPattern.h`.

`kCGPatternTilingConstantSpacing`

Pattern cells are spaced consistently, as with `kCGPatternTilingConstantSpacingMinimalDistortion`. The pattern cell may be distorted additionally to permit a more efficient implementation.

Available in Mac OS X v10.1 and later.

Declared in `CGPattern.h`.

Declared In

`CGPattern.h`

CGPDFArray Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFArray.h
Companion guide	Quartz 2D Programming Guide

Overview

The CGPDFArray header file defines an opaque type that encapsulates a PDF array. A PDF array represents an array structure in a PDF document. PDF arrays may be heterogeneous—that is, they may contain any other PDF objects, including PDF strings, PDF dictionaries, and other PDF arrays.

Many CGPDFArray functions to retrieve values from a PDF array take the form:

```
bool CGPDFArrayGet<DataType> (
 CGPDFArrayRef array,
 size_t index,
 <DataType>Ref *value
);
```

These functions test the data type of the object at the specified index. If the object is not of the expected type, the function returns `false`. If the object is of the expected type, the function returns `true`, and the object is passed back in the `value` parameter.

This opaque type is not derived from CFType and therefore there are no functions for retaining and releasing it. CGPDFArray objects exist only as constituent parts of a CGPDFDocument object, and they are managed by their container.

Functions

CGPDFArrayGetArray

Returns whether an object at a given index in a PDF array is another PDF array and, if so, retrieves that array.

```
bool CGPDFArrayGetArray (
   CGPDFArrayRef array,
   size_t index,
   CGPDFArrayRef *value
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (`0` to `N-1`, where `N` is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF array. If the value at the specified index is a PDF array, then on return that array, otherwise the value is unspecified.

Return Value
Returns `true` if there is a PDF array at the specified index, otherwise `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFArray.h`

CGPDFArrayGetBoolean

Returns whether an object at a given index in a PDF array is a PDF Boolean and, if so, retrieves that Boolean.

```
bool CGPDFArrayGetBoolean (
   CGPDFArrayRef array,
   size_t index,
   CGPDFBoolean *value
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of `array` (`0` to `N-1`, where `N` is the count of `array`), the behavior is undefined.

value

On input, a pointer to a PDF Boolean. If the value at the specified index is a PDF Boolean, then on return that Boolean, otherwise the value is undefined.

Return Value
Returns `true` if there is a PDF Boolean at the specified index, otherwise `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFArray.h`

CGPDFArrayGetCount

Returns the number of items in a PDF array.

```
size_t CGPDFArrayGetCount (
   CGPDFArrayRef array
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

Return Value

Returns the number of items in the array.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFArray.h`

CGPDFArrayGetDictionary

Returns whether an object at a given index in a PDF array is a PDF dictionary and, if so, retrieves that dictionary.

```
bool CGPDFArrayGetDictionary (
   CGPDFArrayRef array,
   size_t index,
   CGPDFDictionaryRef *value
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (`0` to `N-1`, where `N` is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF dictionary. If the value at the specified index is a PDF dictionary, then on return that dictionary, otherwise the value is undefined.

Return Value

Returns `true` if there is a PDF dictionary at the specified index, otherwise `false`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFArray.h`

CGPDFArrayGetInteger

Returns whether an object at a given index in a PDF array is a PDF integer and, if so, retrieves that object.

```
bool CGPDFArrayGetInteger (
   CGPDFArrayRef array,
   size_t index,
   CGPDFInteger *value
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (`0` to `N-1`, where `N` is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF integer. If the value at the specified index is a PDF integer value, then on return contains that value, otherwise the value is undefined.

Return Value
Returns `true` if there is a PDF integer at the specified index, otherwise `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFArray.h`

CGPDFArrayGetName

Returns whether an object at a given index in a PDF array is a PDF name reference (represented as a constant C string) and, if so, retrieves that name.

```
bool CGPDFArrayGetName (
   CGPDFArrayRef array,
   size_t index,
   const char **value
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (`0` to `N-1`, where `N` is the count of the array), the behavior is undefined.

value

An uninitialized pointer to a constant C string. If the value at the specified index is a reference to a PDF name (represented by a constant C string) then upon return, contains that value; otherwise the value is undefined.

Return Value
Returns `true` if there is an array of characters at the specified index, otherwise `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
CGPDFArray.h

CGPDFArrayGetNull

Returns whether an object at a given index in a Quartz PDF array is a PDF null.

```
bool CGPDFArrayGetNull (
   CGPDFArrayRef array,
   size_t index
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (`0` to `N-1`, where `N` is the count of the array), the behavior is undefined.

Return Value
Returns `true` if there is a PDF null at the specified index, otherwise `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
CGPDFArray.h

CGPDFArrayGetNumber

Returns whether an object at a given index in a PDF array is a PDF number and, if so, retrieves that object.

```
bool CGPDFArrayGetNumber (
   CGPDFArrayRef array,
   size_t index,
   CGPDFReal *value
);
```

Parameters

array

A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index

The index of the value to retrieve. If the index is outside the index space of the array (`0` to `N-1`, where `N` is the count of the array), the behavior is undefined.

value

On input, a pointer to a PDF number. If the value at the specified index is a PDF number, then on return contains that value, otherwise the value is undefined.

Return Value
Returns `true` if there is a PDF number at the specified index, otherwise `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
CGPDFArray.h

CGPDFArrayGetObject

Returns whether an object at a given index in a PDF array is a PDF object and, if so, retrieves that object.

```
bool CGPDFArrayGetObject (
   CGPDFArrayRef array,
   size_t index,
   CGPDFObjectRef *value
);
```

Parameters

array
A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index
The index of the value to retrieve. If the index is outside the index space of the array (`0` to `N-1`, where `N` is the count of the array), the behavior is undefined.

value
On input, a pointer to a PDF object. If the value at the specified index is a PDF object, then on return contains that object, otherwise the value is undefined.

Return Value
Returns `true` if there is a PDF object at the specified index, otherwise `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
CGPDFArray.h

CGPDFArrayGetStream

Returns whether an object at a given index in a PDF array is a PDF stream and, if so, retrieves that stream.

```
bool CGPDFArrayGetStream (
   CGPDFArrayRef array,
   size_t index,
   CGPDFStreamRef *value
);
```

Parameters

array
A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index
The index of the value to retrieve. If the index is outside the index space of the array (`0` to `N-1`, where `N` is the count of the array), the behavior is undefined.

value
On input, a pointer to a PDF stream. If the value at the specified index is a PDF stream, then on return that stream, otherwise the value is undefined.

Return Value
Returns `true` if there is a PDF stream at the specified index, otherwise `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFArray.h`

CGPDFArrayGetString

Returns whether an object at a given index in a PDF array is a PDF string and, if so, retrieves that string.

```
bool CGPDFArrayGetString (
   CGPDFArrayRef array,
   size_t index,
   CGPDFStringRef *value
);
```

Parameters

array
A PDF array. If this parameter is not a valid PDF array, the behavior is undefined.

index
The index of the value to retrieve. If the index is outside the index space of the array (`0` to `N-1`, where `N` is the count of the array), the behavior is undefined.

value
On input, a pointer to a PDF string. If the value at the specified index is a PDF string, then on return that string, otherwise the value is undefined.

Return Value
Returns `true` if there is a PDF stream at the specified index, otherwise `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFArray.h`

Data Types

CGPDFArrayRef

An opaque type that encapsulates a PDF array.

```
typedef struct CGPDFArray *CGPDFArrayRef;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGPDFArray.h`

CGPDFContentStream Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFContentStream.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFContentStreamRef` opaque type provides access to the data that describes the appearance of a PDF page. A CGPDFContentStream object represents one or more PDF content streams for a page and their associated resource dictionaries. A PDF content stream is a sequential set of instructions that specifies how to paint items on a PDF page. A resource dictionary contains information needed by the content stream in order to decode the sequential instructions of the content stream.

CGPDFContentStream functions can retrieve both the content streams and the resource dictionaries associated with a PDF page.

This opaque type is not derived from CFType and therefore there are no functions for retaining and releasing it.

Functions by Task

Creating a PDF Content Stream Object

`CGPDFContentStreamCreateWithPage` (page 294)
Creates a content stream object from a PDF page object.

`CGPDFContentStreamCreateWithStream` (page 294)
Creates a PDF content stream object from an existing PDF content stream object.

Getting Data from a PDF Content Stream Object

`CGPDFContentStreamGetStreams` (page 295)
Gets the array of PDF content streams contained in a PDF content stream object.

`CGPDFContentStreamGetResource` (page 295)
Gets the specified resource from a PDF content stream object.

Retaining and Releasing a PDF Content Stream Object

`CGPDFContentStreamRetain` (page 296)
Increments the retain count of a PDF content stream object.

`CGPDFContentStreamRelease` (page 296)
Decrements the retain count of a PDF content stream object.

Functions

CGPDFContentStreamCreateWithPage

Creates a content stream object from a PDF page object.

```
CGPDFContentStreamRef CGPDFContentStreamCreateWithPage (
   CGPDFPageRef page
);
```

Parameters

page
A PDF page object.

Return Value
A new CGPDFContentStream object. You are responsible for releasing this object by calling the function `CGPDFContentStreamRelease`.

Discussion
A CGPDFContentStream object can contain more than one PDF content stream. To retrieve an array of the PDF content streams in the object, call the function `CGPDFContentStreamGetStreams` (page 295). To obtain the resources associated with a CGPDFContentStream object, call the function `CGPDFContentStreamGetResource` (page 295).

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFContentStream.h`

CGPDFContentStreamCreateWithStream

Creates a PDF content stream object from an existing PDF content stream object.

```
CGPDFContentStreamRef CGPDFContentStreamCreateWithStream (
   CGPDFStreamRef stream,
   CGPDFDictionaryRef streamResources,
   CGPDFContentStreamRef parent
);
```

Parameters

stream
The PDF stream you want to create a content stream from.

streamResources
A PDF dictionary that contains the resources associated with the stream you want to retrieve.

parent
The content stream of the page on which *stream* appears. Supply the `parent` parameter when you create a content stream that's used within a page.

Return Value
A CGPDFContentStream object created from the `stream` parameter. You are responsible for releasing this object by calling the function `CGPDFContentStreamRelease` (page 296).

Discussion
You can use this function to get access to the contents of a form, pattern, Type3 font, or any PDF stream.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFContentStream.h`

CGPDFContentStreamGetResource

Gets the specified resource from a PDF content stream object.

```
CGPDFObjectRef CGPDFContentStreamGetResource (
   CGPDFContentStreamRef cs,
   const char *category,
   const char *name
);
```

Parameters

cs
A CGPDFContentStream object.

category
A string that specifies the category of the resource you want to obtain.

name
A string that specifies the name of the resource you want to obtain.

Return Value
The resource dictionary.

Discussion
You can use this function to obtain resources used by the content stream, such as forms, patterns, color spaces, and fonts.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFContentStream.h`

CGPDFContentStreamGetStreams

Gets the array of PDF content streams contained in a PDF content stream object.

```
CFArrayRef CGPDFContentStreamGetStreams (
   CGPDFContentStreamRef cs
);
```

Parameters

cs

A CGPDFContentStream object.

Return Value
The array of PDF content streams that make up the content stream object represented by the `cs` parameter.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFContentStream.h`

CGPDFContentStreamRelease

Decrements the retain count of a PDF content stream object.

```
void CGPDFContentStreamRelease (
   CGPDFContentStreamRef cs
);
```

Parameters

cs

A PDF content stream.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFContentStream.h`

CGPDFContentStreamRetain

Increments the retain count of a PDF content stream object.

```
CGPDFContentStreamRef CGPDFContentStreamRetain (
   CGPDFContentStreamRef cs
);
```

Parameters

cs

A PDF content stream.

Return Value
The same PDF content stream you passed in as the `cs` parameter.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFContentStream.h`

Data Types

CGPDFContentStreamRef

An opaque type that provides access to the data that describes the appearance of a PDF page.

```
typedef struct CGPDFContentStream *CGPDFContentStreamRef;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPDFContentStream.h`

CGPDFContext Reference

Derived From:	`CGContextRef` (page 131)
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFContext.h
Companion guide	Quartz 2D Programming Guide

Overview

The CGPDFContext header file defines functions that create and get information about a Quartz PDF context. A CGPDFContext object is a type of `CGContextRef` (page 131) that is used for drawing PDF content. The functions in this reference operate only on Quartz PDF graphics contexts created using the functions CGPDFContextCreate (page 301) or CGPDFContextCreateWithURL (page 302).

When you draw to the PDF context using CGContext functions the drawing operations are recorded in PDF format. The PDF commands that represent the drawing are written to the destination specified when you create the PDF graphics context.

Functions by Task

Creating a Context

`CGPDFContextCreate` (page 301)
Creates a PDF graphics context.

`CGPDFContextCreateWithURL` (page 302)
Creates a URL-based PDF graphics context.

Beginning and Ending Pages

`CGPDFContextBeginPage` (page 300)
Begins a new page in a PDF graphics context.

`CGPDFContextEndPage` (page 303)
Ends the current page in the PDF graphics context.

Working with Destinations

`CGPDFContextAddDestinationAtPoint` (page 300)
Sets a destination to jump to when a point in the current page of a PDF graphics context is clicked.

`CGPDFContextSetDestinationForRect` (page 303)
Sets a destination to jump to when a rectangle in the current PDF page is clicked.

`CGPDFContextSetURLForRect` (page 304)
Sets the URL associated with a rectangle in a PDF graphics context.

Closing a PDF Context

`CGPDFContextClose` (page 301)
Closes a PDF document.

Functions

CGPDFContextAddDestinationAtPoint

Sets a destination to jump to when a point in the current page of a PDF graphics context is clicked.

```
void CGPDFContextAddDestinationAtPoint (
    CGContextRef context,
    CFStringRef name,
    CGPoint point
);
```

Parameters

context
A PDF graphics context.

name
A destination name.

point
A location in the current page of the PDF graphics context.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPDFContext.h`

CGPDFContextBeginPage

Begins a new page in a PDF graphics context.

```
void CGPDFContextBeginPage (
    CGContextRef context,
    CFDictionaryRef pageInfo
);
```

Parameters

context

A PDF graphics context.

pageInfo

A dictionary that contains key-value pairs that define the page properties.

Discussion

You must call the function `CGPDFContextEndPage` (page 303) to signal the end of the page.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGPDFContext.h`

CGPDFContextClose

Closes a PDF document.

```
void CGPDFContextClose(
    CGContextRef context
);
```

Parameters

context

A PDF graphics context.

Discussion

After closing the context, all pending data is written to the context destination, and the PDF file is completed. No additional data can be written to the destination context after the PDF document is closed.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CGPDFContext.h`

CGPDFContextCreate

Creates a PDF graphics context.

```
CGContextRef CGPDFContextCreate (
   CGDataConsumerRef consumer,
   const CGRect *mediaBox,
   CFDictionaryRef auxiliaryInfo
);
```

Parameters

consumer

The data consumer to receive the PDF output data.

mediaBox

A pointer to a rectangle that defines the size and location of the PDF page, or `NULL`. The origin of the rectangle should typically be `(0,0)`. Quartz uses this rectangle as the default bounds of the page's media box. If you pass `NULL`, Quartz uses a default page size of 8.5 by 11 inches (612 by 792 points).

auxiliaryInfo

A dictionary that specifies any additional information to be used by the PDF context when generating the PDF file, or `NULL`. The dictionary is retained by the new context, so on return you may safely release it. See "Auxiliary Dictionary Keys" (page 304) for keys you can include in the dictionary.

Return Value

A new PDF context, or `NULL` if the context cannot be created. You are responsible for releasing this object using `CGContextRelease` (page 96).

Discussion

This function creates a PDF drawing environment to your specifications. When you draw into the new context, Quartz renders your drawing as a sequence of PDF drawing commands that are passed to the data consumer object.

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

Declared In

`CGPDFContext.h`

CGPDFContextCreateWithURL

Creates a URL-based PDF graphics context.

```
CGContextRef CGPDFContextCreateWithURL (
   CFURLRef url,
   const CGRect *mediaBox,
   CFDictionaryRef auxiliaryInfo
);
```

Parameters

url

A Core Foundation URL that specifies where you want to place the resulting PDF file.

mediaBox

A rectangle that specifies the bounds of the PDF. The origin of the rectangle should typically be `(0,0)`. The `CGPDFContextCreateWithURL` function uses this rectangle as the default page media bounding box. If you pass `NULL`, `CGPDFContextCreateWithURL` uses a default page size of 8.5 by 11 inches (612 by 792 points).

auxiliaryInfo

A dictionary that specifies any additional information to be used by the PDF context when generating the PDF file, or `NULL`. The dictionary is retained by the new context, so on return you may safely release it.

Return Value
A new PDF context, or `NULL` if a context could not be created. You are responsible for releasing this object using `CGContextRelease` (page 96).

Discussion
When you call this function, Quartz creates a PDF drawing environment—that is, a graphics context—to your specifications. When you draw into the resulting context, Quartz renders your drawing as a series of PDF drawing commands stored in the specified location.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGPDFContext.h`

CGPDFContextEndPage

Ends the current page in the PDF graphics context.

```
void CGPDFContextEndPage (
    CGContextRef context
);
```

Parameters

context

A PDF graphics context.

Discussion
You can call `CGPDFContextEndPage` only after you call the function `CGPDFContextBeginPage` (page 300).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPDFContext.h`

CGPDFContextSetDestinationForRect

Sets a destination to jump to when a rectangle in the current PDF page is clicked.

```
void CGPDFContextSetDestinationForRect (
    CGContextRef context,
    CFStringRef name,
    CGRect rect
);
```

Parameters

context

A PDF graphics context.

name

A destination name.

rect

A rectangle that specifies an area of the current page of a PDF graphics context. The rectangle is specified in default user space (not device space).

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGPDFContext.h

CGPDFContextSetURLForRect

Sets the URL associated with a rectangle in a PDF graphics context.

```
void CGPDFContextSetURLForRect (
   CGContextRef context,
   CFURLRef url,
   CGRect rect
);
```

Parameters

context

A PDF graphics context.

url

A CFURL object that specifies the destination of the contents associated with the rectangle.

rect

A rectangle specified in default user space (not device space).

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGPDFContext.h

Constants

Auxiliary Dictionary Keys

Keys that used to set up a PDF context.

```
CFStringRef kCGPDFContextAuthor;
CFStringRef kCGPDFContextCreator;
CFStringRef kCGPDFContextTitle;
CFStringRef kCGPDFContextOwnerPassword;
CFStringRef kCGPDFContextUserPassword;
CFStringRef kCGPDFContextAllowsPrinting;
CFStringRef kCGPDFContextAllowsCopying;
CFStringRef kCGPDFContextOutputIntent;
CFStringRef kCGPDFContextOutputIntents;
CFStringRef kCGPDFContextSubject;
CFStringRef kCGPDFContextKeywords;
CFStringRef kCGPDFContextEncryptionKeyLength;
```

Constants

`kCGPDFContextAuthor`

The corresponding value is a string that represents the name of the person who created the document. This key is optional.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextCreator`

The corresponding value is a string that represents the name of the application used to produce the document. This key is optional.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextTitle`

The corresponding value is a string that represents the title of the document. This key is optional.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextOwnerPassword`

The owner password of the PDF document. If this key is specified, the document is encrypted using the value as the owner password; otherwise, the document will not be encrypted. The value of this key must be a CFString object that can be represented in ASCII encoding. Only the first 32 bytes are used for the password. There is no default value for this key. If the value of this key cannot be represented in ASCII, the document is not created and the creation function returns `NULL`.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextUserPassword`

The user password of the PDF document. If the document is encrypted, then the value of this key will be the user password for the document. If not specified, the user password is the empty string. The value of this key must be a CFString object that can be represented in ASCII encoding; only the first 32 bytes will be used for the password. If the value of this key cannot be represented in ASCII, the document is not created and the creation function returns `NULL`.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextAllowsPrinting`

Whether the document allows printing when unlocked with the user password. The value of this key must be a CFBoolean value. The default value of this key is `kCFBooleanTrue`.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextAllowsCopying`
Whether the document allows copying when unlocked with the user password. The value of this key must be a CFBoolean object. The default value of this key is `kCFBooleanTrue`.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextOutputIntent`
The output intent `PDF/X`. This key is optional. If present, the value of this key must be a CFDictionary object. The dictionary is added to the `/OutputIntents` entry in the PDF file document catalog. The keys and values contained in the dictionary must match those specified in section 9.10.4 of the PDF 1.4 specification, ISO/DIS 15930-3 document published by ISO/TC 130, and Adobe Technical Note #5413.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextOutputIntents`
Output intent dictionaries. This key is optional. If present, the value must be an array of one or more `kCGPDFContextOutputIntent` dictionaries. The array is added to the PDF document in the `/OutputIntents` entry in the PDF file's document catalog. Each dictionary in the array must be of form specified for the `kCGPDFContextOutputIntent` key, except that only the first dictionary in the array is required to contain the "S" key with a value of `GTS_PDFX`. If both the `kCGPDFContextOutputIntent` and `kCGPDFContextOutputIntents` keys are specified, the former is ignored.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextSubject`
The subject of a document. Optional; if present, the value of this key must be a `CFString` object.

Declared in `CGPDFContext.h`.

Available in Mac OS X v10.5 and later.

`kCGPDFContextKeywords`
The keywords for this document. This key is optional. If the value of this key is a `CFString` object, the `/Keywords` entry will be the specified string. If the value of this key is a `CFArray` object, then it must be an array of `CFString` objects. The `/Keywords` entry will, in this case, be the concatenation of the specified strings separated by commas (" , "). In addition, an entry with the key `"/AAPL:Keywords"` is stored in the document information dictionary; its value is an array consisting of each of the specified strings. The value of this key must be in one of the above forms; otherwise, this key is ignored.

Declared in `CGPDFContext.h`.

Available in Mac OS X v10.5 and later.

`kCGPDFContextEncryptionKeyLength`
The encryption key length in bits; see Table 3.18 "Entries common to all encryption dictionaries", PDF Reference: Adobe PDF version 1.5 (4th ed.) for more information. Optional; if present, the value of this key must be a `CFNumber` object with value which is a multiple of 8 between 40 and 128, inclusive. If this key is absent or invalid, the encryption key length defaults to 40 bits.

Declared in `CGPDFContext.h`.

Available in Mac OS X v10.5 and later.

Discussion
For more information about using these keys in a PDF context, see `CGPDFContextCreate` (page 301) and `CGPDFContextCreateWithURL` (page 302).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPDFContext.h`

Box Dictionary Keys

Keys that specify various PDF boxes.

```
CFStringRef kCGPDFContextMediaBox
CFStringRef kCGPDFContextCropBox
CFStringRef kCGPDFContextBleedBox
CFStringRef kCGPDFContextTrimBox
CFStringRef kCGPDFContextArtBox
```

Constants

`kCGPDFContextMediaBox`
The media box for the document or for a given page. This key is optional. If present, the value of this key must be a CFData object that contains a `CGRect` (stored by value, not by reference).

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextCropBox`
The crop box for the document or for a given page. This key is optional. If present, the value of this key must be a CFData object that contains a `CGRect` (stored by value, not by reference).

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextBleedBox`
The bleed box for the document or for a given page. This key is optional. If present, the value of this key must be a CFData object that contains a `CGRect` (stored by value, not by reference).

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextTrimBox`
The trim box for the document or for a given page. This key is optional. If present, the value of this key must be a CFData object that contains a `CGRect` (stored by value, not by reference).

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFContextArtBox`
The art box for the document or for a given page. This key is optional. If present, the value of this key must be a CFData object that contains a `CGRect` (stored by value, not by reference).

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

Discussion
For more information about using these keys in a PDF context, see `CGPDFContextCreate` (page 301) and `CGPDFContextCreateWithURL` (page 302).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPDFContext.h`

Output Intent Dictionary Keys

Keys to specify output intent options.

```
CFStringRef kCGPDFXOutputIntentSubtype;
CFStringRef kCGPDFXOutputConditionIdentifier;
CFStringRef kCGPDFXOutputCondition;
CFStringRef kCGPDFXRegistryName;
CFStringRef kCGPDFXInfo;
CFStringRef kCGPDFXDestinationOutputProfile;
```

Constants

`kCGPDFXOutputIntentSubtype`
The output intent subtype. This key is required. The value of this key must be a CFString object equal to `"GTS_PDFX"`; otherwise, the dictionary is ignored.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFXOutputConditionIdentifier`
A string identifying the intended output device or production condition in a human- or machine-readable form. This key is required. The value of this key must be a CFString object. For best results, the string should be restricted to characters in the ASCII character set.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFXOutputCondition`
A text string identifying the intended output device or production condition in a human- readable form. This key is optional. If present, the value of this key must be a CFString object.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFXRegistryName`
A string identifying the registry in which the condition designated by `kCGPDFXOutputConditionIdentifier` is defined. This key is optional. If present, the value of this key must be a CFString object. For best results, the string should be lossless in ASCII encoding.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFXInfo`
A human-readable text string containing additional information or comments about the intended target device or production condition. This key is required if the value of `kCGPDFXOutputConditionIdentifier` does not specify a standard production condition. It is optional otherwise. If present, the value of this key must be a CFString object.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

`kCGPDFXDestinationOutputProfile`

An ICC profile stream defining the transformation from the PDF document's source colors to output device colorants. This key is required if the value of `kCGPDFXOutputConditionIdentifier` does not specify a standard production condition. It is optional otherwise. If present, the value of this key must be an ICC-based color space specified as a `CGColorSpace` object.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFContext.h`.

Discussion
For more information about using these keys in a PDF context, see `CGPDFContextCreate` (page 301) and `CGPDFContextCreateWithURL` (page 302).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPDFContext.h`

CGPDFDictionary Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFDictionary.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFDictionaryRef` opaque type encapsulates a PDF dictionary whose key-value pairs can specify any kind of PDF object, including another dictionary. Dictionary objects are the main building blocks of a PDF document. A key-value pair within a dictionary is called an entry. In a PDF dictionary, the key must be an array of characters. Within a given dictionary, the keys are unique—that is, no two keys in a single dictionary are equal (as determined by `strcmp`). The value associated with a key can be any kind of PDF object, including another dictionary. Dictionary objects are the main building blocks of a PDF document.

Many functions that retrieve values from a PDF dictionary take the form:

```
bool CGPDFDictionaryGet<DataType> (
 CGPDFDictionaryRef dictionary,
 const char *key,
 <DataType>Ref *value
);
```

These functions test whether there is an object associated with the specified key. If there is an object associated with the specified key, they test its data type. If there is no associated object, or if there is but it is not of the expected type, the function returns `false`. If there is an object associated with the specified key and it is of the expected type, the function returns `true` and the object is passed back in the `value` parameter.

This opaque type is not derived from CFType and therefore there are no functions for retaining and releasing it. CGPDFDictionary objects exist only as constituent parts of a CGPDFDocument object, and they are managed by their container.

Functions by Task

Applying a Function to All Entries

`CGPDFDictionaryApplyFunction` (page 312)
Applies a function to each entry in a dictionary.

Getting Data from a Dictionary

`CGPDFDictionaryGetArray` (page 313)
Returns whether there is a PDF array associated with a specified key in a PDF dictionary and, if so, retrieves that array.

`CGPDFDictionaryGetBoolean` (page 314)
Returns whether there is a PDF Boolean value associated with a specified key in a PDF dictionary and, if so, retrieves the Boolean value.

`CGPDFDictionaryGetCount` (page 314)
Returns the number of entries in a PDF dictionary.

`CGPDFDictionaryGetDictionary` (page 314)
Returns whether there is another PDF dictionary associated with a specified key in a PDF dictionary and, if so, retrieves that dictionary.

`CGPDFDictionaryGetInteger` (page 315)
Returns whether there is a PDF integer associated with a specified key in a PDF dictionary and, if so, retrieves that integer.

`CGPDFDictionaryGetName` (page 316)
Returns whether an object with a specified key in a PDF dictionary is a PDF name reference (represented as a constant C string) and, if so, retrieves that name.

`CGPDFDictionaryGetNumber` (page 316)
Returns whether there is a PDF number associated with a specified key in a PDF dictionary and, if so, retrieves that number.

`CGPDFDictionaryGetObject` (page 317)
Returns whether there is a PDF object associated with a specified key in a PDF dictionary and, if so, retrieves that object.

`CGPDFDictionaryGetStream` (page 317)
Returns whether there is a PDF stream associated with a specified key in a PDF dictionary and, if so, retrieves that stream.

`CGPDFDictionaryGetString` (page 318)
Returns whether there is a PDF string associated with a specified key in a PDF dictionary and, if so, retrieves that string.

Functions

CGPDFDictionaryApplyFunction

Applies a function to each entry in a dictionary.

```
void CGPDFDictionaryApplyFunction (
   CGPDFDictionaryRef dict,
   CGPDFDictionaryApplierFunction function,
   void *info
);
```

Parameters

dictionary
A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

function
The function to apply to each entry in the dictionary.

info
A pointer to contextual information to pass to the function.

Discussion
This function enumerates all of the entries in the dictionary, calling the function once for each. The current key, its associated value, and the contextual information are passed to the function (see also CGPDFDictionaryApplierFunction (page 318)).

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFDictionary.h`

CGPDFDictionaryGetArray

Returns whether there is a PDF array associated with a specified key in a PDF dictionary and, if so, retrieves that array.

```
bool CGPDFDictionaryGetArray (
   CGPDFDictionaryRef dict,
   const char *key,
   CGPDFArrayRef *value
);
```

Parameters

dictionary
A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key
The key for the value to retrieve.

value
On input, an uninitialized pointer to a PDF array. If the value associated with the specified key is a PDF array, then on return contains that array; otherwise the value is unspecified.

Return Value
Returns `true` if there is a PDF array associated with the specified key; otherwise, `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFDictionary.h`

CGPDFDictionaryGetBoolean

Returns whether there is a PDF Boolean value associated with a specified key in a PDF dictionary and, if so, retrieves the Boolean value.

```
bool CGPDFDictionaryGetBoolean (
   CGPDFDictionaryRef dict,
   const char *key,
   CGPDFBoolean *value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF Boolean value. If the value associated with the specified key is a PDF Boolean value, then on return contains that value; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF Boolean value associated with the specified key; otherwise, `false`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFDictionary.h`

CGPDFDictionaryGetCount

Returns the number of entries in a PDF dictionary.

```
size_t CGPDFDictionaryGetCount (
   CGPDFDictionaryRef dict
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

Return Value

Returns the number of entries in the dictionary.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFDictionary.h`

CGPDFDictionaryGetDictionary

Returns whether there is another PDF dictionary associated with a specified key in a PDF dictionary and, if so, retrieves that dictionary.

```
bool CGPDFDictionaryGetDictionary (
   CGPDFDictionaryRef dict,
   const char *key,
   CGPDFDictionaryRef *value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF dictionary. If the value associated with the specified key is a PDF dictionary, then on return contains that dictionary; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF dictionary associated with the specified key; otherwise, `false`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFDictionary.h`

CGPDFDictionaryGetInteger

Returns whether there is a PDF integer associated with a specified key in a PDF dictionary and, if so, retrieves that integer.

```
bool CGPDFDictionaryGetInteger (
   CGPDFDictionaryRef dict,
   const char *key,
   CGPDFInteger *value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF integer. If the value associated with the specified key is a PDF integer, then on return contains that value; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF integer associated with the specified key; otherwise, `false`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFDictionary.h`

CGPDFDictionaryGetName

Returns whether an object with a specified key in a PDF dictionary is a PDF name reference (represented as a constant C string) and, if so, retrieves that name.

```
bool CGPDFDictionaryGetName (
   CGPDFDictionaryRef dict,
   const char *key,
   const char **value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF name reference, represented as a constant C string. If the value associated with the specified key is a reference to a PDF name, then on return, the variable points to the name; otherwise, the value is undefined.

Return Value

Returns `true` if there is a character array associated with the specified key; otherwise, `false`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFDictionary.h`

CGPDFDictionaryGetNumber

Returns whether there is a PDF number associated with a specified key in a PDF dictionary and, if so, retrieves that number.

```
bool CGPDFDictionaryGetNumber (
   CGPDFDictionaryRef dict,
   const char *key,
   CGPDFReal *value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF number. If the value associated with the specified key is a PDF number (real or integer), then on return contains that value; otherwise the value is unspecified.

Return Value

Returns `true` if there is a PDF number associated with the specified key; otherwise, `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFDictionary.h`

CGPDFDictionaryGetObject

Returns whether there is a PDF object associated with a specified key in a PDF dictionary and, if so, retrieves that object.

```
bool CGPDFDictionaryGetObject (
   CGPDFDictionaryRef dict,
   const char *key,
   CGPDFObjectRef *value
);
```

Parameters

dictionary
A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key
The key for the value to retrieve.

value
On input, a pointer to a PDF object. If the value associated with the specified key is a PDF object, then on return contains that object; otherwise the value is unspecified.

Return Value
Returns `true` if there is a PDF object associated with the specified key; otherwise, `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFDictionary.h`

CGPDFDictionaryGetStream

Returns whether there is a PDF stream associated with a specified key in a PDF dictionary and, if so, retrieves that stream.

```
bool CGPDFDictionaryGetStream (
   CGPDFDictionaryRef dict,
   const char *key,
   CGPDFStreamRef *value
);
```

Parameters

dictionary
A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key
The key for the value to be retrieved.

value

On input, a pointer to a PDF stream. If the value associated with the specified key is a PDF stream, then on return contains that stream; otherwise, the value is unspecified.

Return Value
Returns `true` if there is a PDF stream associated with the specified key; otherwise, `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFDictionary.h`

CGPDFDictionaryGetString

Returns whether there is a PDF string associated with a specified key in a PDF dictionary and, if so, retrieves that string.

```
bool CGPDFDictionaryGetString (
   CGPDFDictionaryRef dict,
   const char *key,
   CGPDFStringRef *value
);
```

Parameters

dictionary

A PDF dictionary. If this parameter is not a valid PDF dictionary, the behavior is undefined.

key

The key for the value to retrieve.

value

On input, a pointer to a PDF string. If the value associated with the specified key is a PDF string, then on return contains that string; otherwise the value is unspecified.

Return Value
Returns `true` if there is a PDF string associated with the specified key; otherwise, `false`.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFDictionary.h`

Callbacks

CGPDFDictionaryApplierFunction

Performs custom processing on a key-value pair from a PDF dictionary, using optional contextual information.

```
typedef void (*CGPDFDictionaryApplierFunction) (
   const char *key,
   CGPDFObjectRef value,
   void *info,
);
```

If you name your function `MyFunction`, you would declare it like this:

```
void MyFunction (
   const char *key,
   CGPDFObjectRef object,
   void *info
);
```

Parameters

key

The current key in the dictionary.

object

The value in the dictionary associated with the key.

info

The contextual information that your provided in the `info` parameter in `CGPDFDictionaryApplyFunction` (page 312).

Discussion

`CGPDFDictionaryApplierFunction` defines the callback for `CGPDFDictionaryApplyFunction`, that enumerates all of the entries in the dictionary, calling your custom applier function once for each entry. The current key, its associated value, and the contextual information are passed to your applier function using the `key`, `value`, and `info` parameters respectively.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGPDFDictionary.h`

Data Types

CGPDFDictionaryRef

An opaque type that encapsulates a PDF dictionary.

```
typedef struct CGPDFDictionary *CGPDFDictionaryRef;
```

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGPDFDictionary.h`

CGPDFDocument Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFDocument.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFDocumentRef` opaque type encapsulates a document that contains PDF (Portable Document Format) drawing information. PDF provides an efficient format for cross-platform exchange of documents with rich content. PDF files can contain multiple pages of images and text. A PDF document object contains all the information relating to a PDF document, including its catalog and contents.

Note that PDF documents may be encrypted, and that some operations may be restricted until a valid password is supplied—see the functions listed in "Managing Encryption" (page 322). Quartz also supports decrypting encrypted documents.

Quartz can both display and generate files that are compliant with the PDF standard. When imaging PDF files, `CGPDFDocumentRef` is the basic type used to represent a PDF document.

Functions by Task

Creating PDF Document Objects

`CGPDFDocumentCreateWithProvider` (page 324)
Creates a Quartz PDF document using a data provider.

`CGPDFDocumentCreateWithURL` (page 324)
Creates a Quartz PDF document using data specified by a URL.

Retaining and Releasing PDF Documents

`CGPDFDocumentRelease` (page 333)
Decrements the retain count of a PDF document.

`CGPDFDocumentRetain` (page 333)
Increments the retain count of a Quartz PDF document.

Getting the CFType ID for a PDF Document Object

`CGPDFDocumentGetTypeID` (page 331)
Returns the type identifier for Quartz PDF documents.

Getting Information About Quartz PDF Documents

`CGPDFDocumentGetCatalog` (page 326)
Returns the document catalog of a Quartz PDF document.

`CGPDFDocumentGetNumberOfPages` (page 329)
Returns the number of pages in a PDF document.

`CGPDFDocumentGetPage` (page 329)
Returns a page from a Quartz PDF document.

`CGPDFDocumentGetVersion` (page 331)
Returns the major and minor version numbers of a Quartz PDF document.

`CGPDFDocumentGetInfo` (page 328)
Gets the information dictionary for a PDF document.

`CGPDFDocumentGetID` (page 327)
Gets the file identifier for a PDF document.

Managing Encryption

`CGPDFDocumentAllowsCopying` (page 323)
Returns whether the specified PDF document allows copying.

`CGPDFDocumentAllowsPrinting` (page 323)
Returns whether a PDF document allows printing.

`CGPDFDocumentIsEncrypted` (page 332)
Returns whether the specified PDF file is encrypted.

`CGPDFDocumentIsUnlocked` (page 332)
Returns whether the specified PDF document is currently unlocked.

`CGPDFDocumentUnlockWithPassword` (page 334)
Unlocks an encrypted PDF document, if a valid password is supplied.

Getting Page Information

`CGPDFDocumentGetArtBox` (page 325) Deprecated in Mac OS X version 10.3 and later
Returns the art box of a page in a PDF document.

`CGPDFDocumentGetBleedBox` (page 325) Deprecated in Mac OS X version 10.3 and later
Returns the bleed box of a page in a PDF document.

`CGPDFDocumentGetCropBox` (page 327) Deprecated in Mac OS X version 10.3 and later
Returns the crop box of a page in a PDF document.

`CGPDFDocumentGetMediaBox` (page 328) Deprecated in Mac OS X version 10.3 and later
Returns the media box of a page in a PDF document.

`CGPDFDocumentGetRotationAngle` (page 330) Deprecated in Mac OS X version 10.3 and later
Returns the rotation angle of a page in a PDF document.

`CGPDFDocumentGetTrimBox` (page 330) Deprecated in Mac OS X version 10.3 and later
Returns the trim box of a page in a PDF document.

Functions

CGPDFDocumentAllowsCopying

Returns whether the specified PDF document allows copying.

```
bool CGPDFDocumentAllowsCopying (
   CGPDFDocumentRef document
);
```

Parameters

document
A PDF document.

Return Value

A Boolean that, if `true`, indicates that the document allows copying. If the value is `false`, the document does not allow copying.

Discussion

This function returns `true` if the specified PDF document allows copying. It returns `false` if the document is encrypted and the current password doesn't grant permission to perform copying.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGPDFDocument.h`

CGPDFDocumentAllowsPrinting

Returns whether a PDF document allows printing.

```
bool CGPDFDocumentAllowsPrinting (
   CGPDFDocumentRef document
);
```

Parameters

document
A PDF document.

Return Value

A Boolean that, if `true`, indicates that the document allows printing. If the value is `false`, the document does not allow printing.

Discussion

This function returns `true` if the specified PDF document allows printing. It returns `false` if the document is encrypted and the current password doesn't grant permission to perform printing.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentCreateWithProvider

Creates a Quartz PDF document using a data provider.

```
CGPDFDocumentRef CGPDFDocumentCreateWithProvider (
   CGDataProviderRef provider
);
```

Parameters

provider

A data provider that supplies the PDF document data.

Return Value
A new Quartz PDF document, or `NULL` if a document can not be created. You are responsible for releasing the object using `CGPDFDocumentRelease` (page 333).

Discussion
Distributing individual pages of a PDF document to separate threads is not supported. If you want to use threads, consider creating a separate document for each thread and operating on a block of pages per thread.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextDrawPDFDocument` (page 82)

Related Sample Code
CarbonSketch

Declared In
`CGPDFDocument.h`

CGPDFDocumentCreateWithURL

Creates a Quartz PDF document using data specified by a URL.

```
CGPDFDocumentRef CGPDFDocumentCreateWithURL (
   CFURLRef url
);
```

Parameters

url

The URL address at which the PDF document data is located.

Return Value
A new Quartz PDF document, or `NULL` if a document could not be created. You are responsible for releasing the object using `CGPDFDocumentRelease` (page 333).

Discussion
Distributing individual pages of a PDF document to separate threads is not supported. If you want to use threads, consider creating a separate document for each thread and operating on a block of pages per thread.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGContextDrawPDFDocument` (page 82)

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetArtBox

Returns the art box of a page in a PDF document. (Deprecated in Mac OS X version 10.3 and later.)

```
CGRect CGPDFDocumentGetArtBox (
   CGPDFDocumentRef document,
   int page
);
```

Parameters

document
The PDF document to examine.

page
An integer that specifies the number of the page to examine.

Return Value
A rectangle that represents the art box for the specified page, expressed in default PDF user space units (points).

Discussion
The replacement function for this one is `CGPDFPageGetBoxRect`, which gets the rectangle associated with a type of box (art, media, crop, bleed trim) that represents a content region or page dimensions of a PDF page. For more information see *CGPDFPage Reference*.

The art box defines the extent of the page's meaningful content (including potential white space) as intended by the document creator. The default value is the page's crop box.

Availability
Available in Mac OS X version 10.0 and later.
Deprecated in Mac OS X version 10.3 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetBleedBox

Returns the bleed box of a page in a PDF document. (Deprecated in Mac OS X version 10.3 and later.)

```
CGRect CGPDFDocumentGetBleedBox (
   CGPDFDocumentRef document,
   int page
);
```

Parameters

document

The PDF document to examine.

page

An integer that specifies the number of the page to examine.

Return Value
A rectangle that represents the bleed box for the specified page, expressed in default PDF user space units (points).

Discussion
The replacement function for this one is `CGPDFPageGetBoxRect`, which gets the rectangle associated with a type of box (art, media, crop, bleed trim) that represents a content region or page dimensions of a PDF page. For more information see *CGPDFPage Reference*.

The bleed box defines the bounds to which the contents of the page should be clipped when output in a production environment. The default value is the page's crop box.

Availability
Available in Mac OS X version 10.0 and later.
Deprecated in Mac OS X version 10.3 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetCatalog

Returns the document catalog of a Quartz PDF document.

```
CGPDFDictionaryRef CGPDFDocumentGetCatalog (
   CGPDFDocumentRef document
);
```

Parameters

document

A PDF document.

Return Value
The document catalog of the specified document.

Discussion
The entries in a PDF document catalog recursively describe the contents of the PDF document. You can access the contents of a PDF document catalog by calling the function `CGPDFDocumentGetCatalog`. For information on accessing PDF metadata, see *Quartz 2D Programming Guide*.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetCropBox

Returns the crop box of a page in a PDF document. (Deprecated in Mac OS X version 10.3 and later.)

```
CGRect CGPDFDocumentGetCropBox (
   CGPDFDocumentRef document,
   int page
);
```

Parameters

document

The PDF document to examine.

page

An integer that specifies the number of the page to examine.

Return Value

A rectangle that represents the crop box for the specified page, expressed in default PDF user space units (points).

Discussion

The replacement function for this one is `CGPDFPageGetBoxRect`, which gets the rectangle associated with a type of box (art, media, crop, bleed trim) that represents a content region or page dimensions of a PDF page. For more information see *CGPDFPage Reference*.

The crop box defines the region to which the contents of the page are to be clipped (or cropped) when displayed or printed. Unlike the other boxes, the crop box has no defined meaning in terms of physical page geometry or intended use—it merely suggests where the page should be clipped.

Availability

Available in Mac OS X version 10.0 and later.

Deprecated in Mac OS X version 10.3 and later.

Declared In

`CGPDFDocument.h`

CGPDFDocumentGetID

Gets the file identifier for a PDF document.

```
CGPDFArrayRef CGPDFDocumentGetID (
   CGPDFDocumentRef document
);
```

Parameters

document

The document whose file identifier you want to obtain.

Return Value

Returns the file identifier for the document.

Discussion

A PDF file identifier is defined in the PDF specification as an array of two strings, the first of which is a permanent identifier that doesn't change even when the file is updated. The second string changes each time the file is updated. For more information, see *PDF Reference: Version 1.3 (Second Edition)*, Adobe Systems Incorporated.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetInfo

Gets the information dictionary for a PDF document.

```
CGPDFDictionaryRef CGPDFDocumentGetInfo (
   CGPDFDocumentRef document
);
```

Parameters

document
The document whose dictionary you want to obtain.

Return Value
The information dictionary for the document.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetMediaBox

Returns the media box of a page in a PDF document. (Deprecated in Mac OS X version 10.3 and later.)

```
CGRect CGPDFDocumentGetMediaBox (
   CGPDFDocumentRef document,
   int page
);
```

Parameters

document
The PDF document to examine.

page
An integer that specifies the number of the page to examine.

Return Value
A rectangle that represents the media box for the specified page, expressed in default PDF user space units (points).

Discussion
The replacement function for this one is `CGPDFPageGetBoxRect`, which gets the rectangle associated with a type of box (art, media, crop, bleed trim) that represents a content region or page dimensions of a PDF page. For more information see *CGPDFPage Reference*.

The media box defines the location and size of the physical medium on which the page is intended to be displayed or printed. For example, if the page size is 8.5 by 11 inches, this function returns the coordinate pairs `(0,0)` and `(612,792)`.

Availability
Available in Mac OS X version 10.0 and later.
Deprecated in Mac OS X version 10.3 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetNumberOfPages

Returns the number of pages in a PDF document.

```
size_t CGPDFDocumentGetNumberOfPages (
   CGPDFDocumentRef document
);
```

Parameters

document
The PDF document to examine.

Return Value
The total number of pages in the PDF document.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetPage

Returns a page from a Quartz PDF document.

```
CGPDFPageRef CGPDFDocumentGetPage (
   CGPDFDocumentRef document,
   size_t pageNumber
);
```

Parameters

document
A PDF document.

pageNumber
The number of the page requested.

Return Value
Return the PDF page corresponding to the specified page number, or `NULL` if no such page exists in the document. Pages are numbered starting at 1.

Availability
Available in Mac OS X version 10.3 and later.

Related Sample Code
CarbonSketch

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetRotationAngle

Returns the rotation angle of a page in a PDF document. (Deprecated in Mac OS X version 10.3 and later.)

```
int CGPDFDocumentGetRotationAngle (
   CGPDFDocumentRef document,
   int page
);
```

Parameters

document

The PDF document to examine.

page

An integer that specifies the number of the page to examine.

Return Value
The rotation angle of the page, expressed in degrees. If the specified page does not exist, returns 0.

Discussion
The replacement function for this one is `CGPDFPageGetRotationAngle`. For more information see *CGPDFPage Reference*.

Availability
Available in Mac OS X version 10.0 and later.
Deprecated in Mac OS X version 10.3 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetTrimBox

Returns the trim box of a page in a PDF document. (Deprecated in Mac OS X version 10.3 and later.)

```
CGRect CGPDFDocumentGetTrimBox (
   CGPDFDocumentRef document,
   int page
);
```

Parameters

document

The PDF document to examine.

page

A value specifying the number of the page to examine.

Return Value
Returns a rectangle that represents the trim box for the specified page, expressed in default PDF user space units (points).

Discussion
The replacement function for this one is `CGPDFPageGetBoxRect`, which gets the rectangle associated with a type of box (art, media, crop, bleed trim) that represents a content region or page dimensions of a PDF page. For more information see *CGPDFPage Reference*.

The trim box defines the intended dimensions of the finished page after trimming. It may be smaller than the media box, to allow for production-related content such as printing instructions, cut marks, or color bars. The default value is the page's crop box.

Availability
Available in Mac OS X version 10.0 and later.
Deprecated in Mac OS X version 10.3 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetTypeID

Returns the type identifier for Quartz PDF documents.

```
CFTypeID CGPDFDocumentGetTypeID (
   void
);
```

Return Value
The identifier for the opaque type `CGPDFDocumentRef` (page 334).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentGetVersion

Returns the major and minor version numbers of a Quartz PDF document.

```
void CGPDFDocumentGetVersion (
   CGPDFDocumentRef document,
   int *majorVersion,
   int *minorVersion
);
```

Parameters

document
A PDF document.

majorVersion
On return, contains the major version number of the document.

minorVersion
On return, contains the minor version number of the document.

Return Value
On return, the values of the `majorVersion` and `minorVersion` parameters are set to the major and minor version numbers of the document respectively.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentIsEncrypted

Returns whether the specified PDF file is encrypted.

```
bool CGPDFDocumentIsEncrypted (
   CGPDFDocumentRef document
);
```

Parameters

document
A PDF document.

Return Value
A Boolean that, if `true`, indicates that the document is encrypted. If the value is `false`, the document is not encrypted.

Discussion
If the document is encrypted, a password must be supplied before certain operations are enabled. For more information, see `CGPDFDocumentUnlockWithPassword` (page 334).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentIsUnlocked

Returns whether the specified PDF document is currently unlocked.

```
bool CGPDFDocumentIsUnlocked (
   CGPDFDocumentRef document
);
```

Parameters

document
A PDF document.

Return Value
A Boolean that, if `true`, indicates that the document is not locked. If the value is `false`, the document is locked.

Discussion
There are two possible reasons why a PDF document is unlocked:

- The document is not encrypted.
- The document is encrypted, and a valid password was previously specified using `CGPDFDocumentUnlockWithPassword` (page 334).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentRelease

Decrements the retain count of a PDF document.

```
void CGPDFDocumentRelease (
    CGPDFDocumentRef document
);
```

Parameters

document
The PDF document to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the `document` parameter is `NULL`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentRetain

Increments the retain count of a Quartz PDF document.

```
CGPDFDocumentRef CGPDFDocumentRetain (
    CGPDFDocumentRef document
);
```

Parameters

document
The PDF document to retain.

Return Value
The same document you passed in as the *document* parameter.

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `document` parameter is `NULL`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGPDFDocument.h`

CGPDFDocumentUnlockWithPassword

Unlocks an encrypted PDF document, if a valid password is supplied.

```
bool CGPDFDocumentUnlockWithPassword (
   CGPDFDocumentRef document,
   const char *password
);
```

Parameters

document

A PDF document.

password

A pointer to a string that contains the password.

Return Value
A Boolean that, if `true`, indicates that the document has been successfully unlocked. If the value is `false`, the document has not been unlocked.

Discussion
Given an encrypted PDF document and a password, this function does the following:

- Sets the lock state of the document, based on the validity of the password.
- Returns `true` if the document is unlocked.
- Returns `false` if the document cannot be unlocked with the specified password.

Unlocking a PDF document makes it possible to decrypt the document and perform other privileged operations. Different passwords enable different operations.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGPDFDocument.h`

Data Types

CGPDFDocumentRef

An opaque type that represents a PDF (Portable Document Format) document.

```
typedef struct CGPDFDocument * CGPDFDocumentRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGPDFDocument.h`

CGPDFObject Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFObject.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFObjectRef` opaque type represents PDF objects in a PDF document. PDF supports several basic types of object: Boolean values, integer and real numbers, strings, names, arrays, dictionaries, and streams. Most of these are represented in Quartz by corresponding specific types. A CGPDFObject can represent any of these types. You use CGPDFObject functions to determine the type of the object, and retrieve the object value if it is of an expected type.

This opaque type is not derived from CFType and therefore there are no functions for retaining and releasing it. CGPDFObject objects exist as constituent parts of a CGPDFDocument object, and are managed by their container.

Functions

CGPDFObjectGetType

Returns the PDF type identifier of an object.

```
CGPDFObjectType CGPDFObjectGetType (
   CGPDFObjectRef object
);
```

Parameters

object

A PDF object. If the value if not a PDF object, the behavior is unspecified.

Return Value

Returns the type of the `object` parameter. See "Data Types" (page 338).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFObject.h`

CGPDFObjectGetValue

Returns whether an object is of a given type and if it is, retrieves its value.

```
bool CGPDFObjectGetValue (
   CGPDFObjectRef object,
   CGPDFObjectType type,
   void *value
);
```

Parameters

object

A PDF object.

type

A PDF object type.

value

If the `object` parameter is a PDF object of the specified type, then on return contains that object, otherwise the value is unspecified.

Return Value

Returns `true` if the specified object is a PDF object of the specified type, otherwise `false`.

Discussion

The function gets the value of the `object` parameter. If the type of `object` is equal to the type specified, then:

- If the `value` parameter is not a null pointer, then the value of `object` is copied to `value`, and the function returns `true`.
- If the `value` parameter is a null pointer, then the function simply returns `true`. This allows you to test whether `object` is of the type specified.

If the type of `object` is `kCGPDFObjectTypeInteger` and `type` is equal to `kCGPDFObjectTypeReal`, then the value of `object` is converted to floating point, the result copied to `value`, and the function returns `true`. If none of the preceding conditions is met, returns `false`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFObject.h`

Data Types

CGPDFObjectRef

An opaque type that contains information about a PDF object.

```
typedef union CGPDFObject *CGPDFObjectRef;
```

Availability

Available in Mac OS X v10.3 and later.

Declared In
CGPDFObject.h

CGPDFBoolean

A PDF Boolean value.

```
typedef unsigned char CGPDFBoolean;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGPDFObject.h

CGPDFInteger

A PDF integer value.

```
typedef long int CGPDFInteger;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGPDFObject.h

CGPDFReal

A PDF real value.

```
typedef CGFloat CGPDFReal;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGPDFObject.h

Constants

PDF Object Types

Types of PDF object.

```
enum CGPDFObjectType {
    kCGPDFObjectTypeNull = 1,
    kCGPDFObjectTypeBoolean,
    kCGPDFObjectTypeInteger,
    kCGPDFObjectTypeReal,
    kCGPDFObjectTypeName,
    kCGPDFObjectTypeString,
    kCGPDFObjectTypeArray,
    kCGPDFObjectTypeDictionary,
    kCGPDFObjectTypeStream
};typedef enum CGPDFObjectType CGPDFObjectType;
```

Constants

`kCGPDFObjectTypeNull`
The type for a PDF null.
Available in Mac OS X v10.3 and later.
Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeBoolean`
The type for a PDF Boolean.
Available in Mac OS X v10.3 and later.
Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeInteger`
The type for a PDF integer.
Available in Mac OS X v10.3 and later.
Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeReal`
The type for a PDF real.
Available in Mac OS X v10.3 and later.
Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeName`
Type for a PDF name.
Available in Mac OS X v10.3 and later.
Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeString`
The type for a PDF string.
Available in Mac OS X v10.3 and later.
Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeArray`
Type for a PDF array.
Available in Mac OS X v10.3 and later.
Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeDictionary`
The type for a PDF dictionary.
Available in Mac OS X v10.3 and later.
Declared in `CGPDFObject.h`.

`kCGPDFObjectTypeStream`

The type for a PDF stream.

Available in Mac OS X v10.3 and later.

Declared in `CGPDFObject.h`.

Declared In

`CGPDFObject.h`

CGPDFOperatorTable Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFOperatorTable.h

Overview

A CGPDFOperatorTable object stores callback functions for PDF operators. You pass an operator table and a PDF content stream to a CGPDFScanner object. When the scanner parses a PDF operator, Quartz invokes your callback for that operator. See also *CGPDFScanner Reference* and *CGPDFContentStream Reference*.

Note: This opaque type is not derived from CFType and therefore you can't use the Core Foundation base functions on it, such as `CFRetain` and `CFRelease`. Memory management is handled by the specific functions `CGPDFOperatorTableRetain` (page 344) and `CGPDFOperatorTableRelease` (page 344).

For more about PDF operators, see the latest version of *PDF Reference*, Adobe Systems Incorporated.

Functions by Task

Creating a PDF Operator Table

`CGPDFOperatorTableCreate` (page 344)
Creates an empty PDF operator table.

Setting Callback Functions

`CGPDFOperatorTableSetCallback` (page 345)
Sets a callback function for a PDF operator.

Retaining and Releasing a PDF Operator Table

`CGPDFOperatorTableRetain` (page 344)
Increments the retain count of a CGPDFOperatorTable object.

`CGPDFOperatorTableRelease` (page 344)
Decrements the retain count of a CGPDFOperatorTable object.

Functions

CGPDFOperatorTableCreate

Creates an empty PDF operator table.

```
CGPDFOperatorTableRef CGPDFOperatorTableCreate (
   void
);
```

Return Value
An empty PDF operator table. You are responsible for releasing this object by calling `CGPDFOperatorTableRelease` (page 344).

Discussion
Call the function `CGPDFOperatorTableSetCallback` (page 345) to fill the operator table with callbacks.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFOperatorTable.h`

CGPDFOperatorTableRelease

Decrements the retain count of a CGPDFOperatorTable object.

```
void CGPDFOperatorTableRelease (
   CGPDFOperatorTableRef table
);
```

Parameters
table
A PDF operator table.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFOperatorTable.h`

CGPDFOperatorTableRetain

Increments the retain count of a CGPDFOperatorTable object.

```
CGPDFOperatorTableRef CGPDFOperatorTableRetain (
   CGPDFOperatorTableRef table
);
```

Parameters

table

A PDF operator table.

Return Value

The same PDF operator table you passed in as the `table` parameter.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGPDFOperatorTable.h`

CGPDFOperatorTableSetCallback

Sets a callback function for a PDF operator.

```
void CGPDFOperatorTableSetCallback (
   CGPDFOperatorTableRef table,
   const char *name,
   CGPDFOperatorCallback callback
);
```

Parameters

table

A PDF operator table.

name

The name of the PDF operator you want to set a callback for.

callback

The callback to invoke for the PDF operator specified by the `name` parameter.

Discussion

You call the function `CGPDFOperatorTableSetCallback` for each PDF operator for which you want to provide a callback. See Appendix A in the *PDF Reference, Second Edition*, version 1.3, Adobe Systems Incorporated for a summary of PDF operators.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGPDFOperatorTable.h`

Callbacks

CGPDFOperatorCallback

Performs custom processing for PDF operators.

```
typedef void (*CGPDFOperatorCallback)(
   CGPDFScannerRef scanner,
   void *info
);
```

If you name your function `MyCGPDFOperatorCallback`, you would declare it like this:

```
void MyCGPDFOperatorCallback (
   CGPDFScannerRef scanner,
   void *info
);
```

Parameters

scanner

A CGPDFScanner object. Quartz passes the scanner to your callback function. The scanner contains the PDF content stream that has the PDF operator that corresponds to this callback.

info

A pointer to data passed to the callback.

Discussion

Your callback function takes any action that's appropriate for your application. For example, if you want to count the number of inline images in a PDF but ignore the image data, you would set a callback for the `EI` operator. In your callback you would increment a counter for each call.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGPDFOperatorTable.h`

Data Types

CGPDFOperatorTableRef

An opaque type that stores callback functions for PDF operators.

```
typedef struct CGPDFOperatorTable *CGPDFOperatorTableRef;
```

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGPDFOperatorTable.h`

CGPDFPage Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFPage.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFPageRef` opaque type represents a page in a PDF document.

Functions by Task

Retaining and Releasing a PDF Page

`CGPDFPageRetain` (page 352)
Increments the retain count of a PDF page.

`CGPDFPageRelease` (page 351)
Decrements the retain count of a PDF page.

Getting the CFType ID

`CGPDFPageGetTypeID` (page 351)
Returns the CFType ID for PDF page objects.

Getting Page Information

`CGPDFPageGetBoxRect` (page 348)
Returns the rectangle that represents a type of box for a content region or page dimensions of a PDF page.

`CGPDFPageGetDictionary` (page 348)
Returns the dictionary of a PDF page.

`CGPDFPageGetDocument` (page 349)
Returns the document for a page.

`CGPDFPageGetDrawingTransform` (page 349)
Returns the affine transform that maps a box to a given rectangle on a PDF page.

`CGPDFPageGetPageNumber` (page 350)
Returns the page number of the specified PDF page.

`CGPDFPageGetRotationAngle` (page 351)
Returns the rotation angle of a PDF page.

Functions

CGPDFPageGetBoxRect

Returns the rectangle that represents a type of box for a content region or page dimensions of a PDF page.

```
CGRect CGPDFPageGetBoxRect (
   CGPDFPageRef page,
   CGPDFBox box
);
```

Parameters

page
A PDF page.

box
A `CGPDFBox` constant that specifies the type of box. For possible values, see "`PDF Boxes`" (page 353).

Return Value

Returns the rectangle associated with the type of box specified by the `box` parameter in the specified page.

Discussion

Returns the rectangle associated with the specified box in the specified page. This is the value of the corresponding entry (such as `/MediaBox`, `/ArtBox`, and so on) in the page's dictionary.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CarbonSketch

Declared In

`CGPDFPage.h`

CGPDFPageGetDictionary

Returns the dictionary of a PDF page.

```
CGPDFDictionaryRef CGPDFPageGetDictionary (
   CGPDFPageRef page
);
```

Parameters

page

A PDF page.

Return Value
Returns the PDF dictionary for the specified page.

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGPDFPage.h

CGPDFPageGetDocument

Returns the document for a page.

```
CGPDFDocumentRef CGPDFPageGetDocument (
   CGPDFPageRef page
);
```

Parameters

page

A PDF page.

Return Value
The PDF document with which the specified page is associated.

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGPDFPage.h

CGPDFPageGetDrawingTransform

Returns the affine transform that maps a box to a given rectangle on a PDF page.

```
CGAffineTransform CGPDFPageGetDrawingTransform (
   CGPDFPageRef page,
   CGPDFBox box,
   CGRect rect,
   int rotate,
   bool preserveAspectRatio
);
```

Parameters

page

A PDF page.

box

A `CGPDFBox` constant that specifies the type of box. For possible values, see "PDF Boxes" (page 353).

rect

A Quartz rectangle.

rotate

An integer, that must be a multiple of `90`, that specifies the angle by which the specified rectangle is rotated clockwise.

preserveAspectRatio

A Boolean value that specifies whether or not the aspect ratio should be preserved. A value of `true` specifies that the aspect ratio should be preserved.

Return Value
An affine transform that maps the box specified by the `box` parameter to the rectangle specified by the `rect` parameter.

Discussion
Quartz constructs the affine transform as follows:

- Computes the effective rectangle by intersecting the rectangle associated with `box` and the `/MediaBox` entry of the specified page.
- Rotates the effective rectangle according to the page's `/Rotate` entry.
- Centers the resulting rectangle on `rect`. If the value of the `rotate` parameter is non-zero, then the rectangle is rotated clockwise by rotate degrees. The value of `rotate` must be a multiple of 90.
- Scales the rectangle, if necessary, so that it coincides with the edges of `rect`. If the value of `preserveAspectRatio` parameter is `true`, then the final rectangle coincides with the edges of `rect` only in the more restrictive dimension.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGPDFPage.h`

CGPDFPageGetPageNumber

Returns the page number of the specified PDF page.

```
size_t CGPDFPageGetPageNumber (
   CGPDFPageRef page
);
```

Parameters

page

A PDF page.

Return Value
Returns the page number of the specified page.

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGPDFPage.h

CGPDFPageGetRotationAngle

Returns the rotation angle of a PDF page.

```
int CGPDFPageGetRotationAngle (
   CGPDFPageRef page
);
```

Parameters

page
A PDF page.

Return Value
The rotation angle (in degrees) of the specified page. This is the value of the `/Rotate` entry in the page's dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGPDFPage.h

CGPDFPageGetTypeID

Returns the CFType ID for PDF page objects.

```
CFTypeID CGPDFPageGetTypeID (
   void
);
```

Return Value
Returns the Core Foundation type for a PDF page.

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGPDFPage.h

CGPDFPageRelease

Decrements the retain count of a PDF page.

```
void CGPDFPageRelease (
   CGPDFPageRef page
);
```

Parameters

page
A PDF page.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the `page` parameter is `NULL`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGPDFPage.h`

CGPDFPageRetain

Increments the retain count of a PDF page.

```
CGPDFPageRef CGPDFPageRetain (
   CGPDFPageRef page
);
```

Parameters

page

A PDF page.

Return Value
The same page you passed in as the `page` parameter.

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `page` parameter is `NULL`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGPDFPage.h`

Data Types

CGPDFPageRef

An opaque type that represents a page in a PDF document.

```
typedef struct CGPDFPage *CGPDFPageRef;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGPDFPage.h`

Constants

PDF Boxes

Box types for a PDF page.

```
enum CGPDFBox {
    kCGPDFMediaBox = 0,
    kCGPDFCropBox = 1,
    kCGPDFBleedBox = 2,
    kCGPDFTrimBox = 3,
    kCGPDFArtBox = 4
};
typedef enum CGPDFBox CGPDFBox;
```

Constants

`kCGPDFMediaBox`

The page media box—a rectangle, expressed in default user space units, that defines the boundaries of the physical medium on which the page is intended to be displayed or printed

Available in Mac OS X v10.3 and later.

Declared in `CGPDFPage.h`.

`kCGPDFCropBox`

The page crop box—a rectangle, expressed in default user space units, that defines the visible region of default user space. When the page is displayed or printed, its contents are to be clipped to this rectangle.

Available in Mac OS X v10.3 and later.

Declared in `CGPDFPage.h`.

`kCGPDFBleedBox`

The page bleed box—a rectangle, expressed in default user space units, that defines the region to which the contents of the page should be clipped when output in a production environment

Available in Mac OS X v10.3 and later.

Declared in `CGPDFPage.h`.

`kCGPDFTrimBox`

The page trim box—a rectangle, expressed in default user space units, that defines the intended dimensions of the finished page after trimming.

Available in Mac OS X v10.3 and later.

Declared in `CGPDFPage.h`.

`kCGPDFArtBox`

The page art box—a rectangle, expressed in default user space units, defining the extent of the page's meaningful content (including potential white space) as intended by the page's creator.

Available in Mac OS X v10.3 and later.

Declared in `CGPDFPage.h`.

Declared In

`CGPDFPage.h`

CGPDFScanner Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFScanner.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFScannerRef` opaque type is used to parse a PDF content stream. You can set up the PDF scanner object to invoke callbacks when it encounters specific PDF operators in the stream.

This opaque type is not derived from CFType and therefore there are no functions for retaining and releasing it.

Functions by Task

Creating a PDF Scanner Object

`CGPDFScannerCreate` (page 356)
Creates a CGPDFScanner object.

Retaining and Releasing PDF Scanner Objects

`CGPDFScannerRetain` (page 362)
Increments the retain count of a scanner object.

`CGPDFScannerRelease` (page 361)
Decrements the retain count of a scanner object.

Parsing Content

`CGPDFScannerScan` (page 362)
Parses the content stream of a CGPDFScanner object.

`CGPDFScannerGetContentStream` (page 357)
Returns the content stream associated with a CGPDFScanner object.

Getting PDF Objects from the Scanner Stack

`CGPDFScannerPopObject` (page 360)
Retrieves an object from the scanner stack.

`CGPDFScannerPopBoolean` (page 358)
Retrieves a Boolean object from the scanner stack.

`CGPDFScannerPopInteger` (page 358)
Retrieves an integer object from the scanner stack.

`CGPDFScannerPopNumber` (page 359)
Retrieves a real value object from the scanner stack.

`CGPDFScannerPopName` (page 359)
Retrieves a character string from the scanner stack.

`CGPDFScannerPopString` (page 361)
Retrieves a string object from the scanner stack.

`CGPDFScannerPopArray` (page 357)
Retrieves an array object from the scanner stack.

`CGPDFScannerPopDictionary` (page 358)
Retrieves a PDF dictionary object from the scanner stack.

`CGPDFScannerPopStream` (page 360)
Retrieves a PDF stream object from the scanner stack.

Functions

CGPDFScannerCreate

Creates a CGPDFScanner object.

```
CGPDFScannerRef CGPDFScannerCreate (
   CGPDFContentStreamRef cs,
   CGPDFOperatorTableRef table,
   void *info
);
```

Parameters

cs
A CGPDFContentStream object. (See *CGPDFContentStream Reference*.)

table
A CGPDFOperatorTable object that contains callbacks for the PDF operators you want to handle.

info
A pointer to data you want passed to your CGPDFOperatorTable callback function. (See *CGPDFOperatorTable Reference*.)

Return Value
A CGPDFScanner object. You are responsible for releasing this object by calling the function `CGPDFScannerRelease`.

Discussion
When you want to parse the contents of the PDF stream, call the function `CGPDFScannerScan` (page 362).

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFScanner.h`

CGPDFScannerGetContentStream

Returns the content stream associated with a CGPDFScanner object.

```
CGPDFContentStreamRef CGPDFScannerGetContentStream (
   CGPDFScannerRef scanner
);
```

Parameters

scanner
The scanner object whose content stream you want to obtain.

Return Value
Return the content stream associated with `scanner`.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFScanner.h`

CGPDFScannerPopArray

Retrieves an array object from the scanner stack.

```
bool CGPDFScannerPopArray (
   CGPDFScannerRef scanner,
   CGPDFArrayRef *value
);
```

Parameters

scanner
A valid scanner object.

value
On output, points to the CGPDFArray object popped from the `scanner` stack.

Return Value
Returns `true` if `value` is retrieved successfully; `false` otherwise.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFScanner.h`

CGPDFScannerPopBoolean

Retrieves a Boolean object from the scanner stack.

```
bool CGPDFScannerPopBoolean (
   CGPDFScannerRef scanner,
   CGPDFBoolean *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the CGPDFBoolean object popped from the `scanner` stack.

Return Value

Returns `true` if `value` is retrieved successfully; `false` otherwise.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGPDFScanner.h`

CGPDFScannerPopDictionary

Retrieves a PDF dictionary object from the scanner stack.

```
bool CGPDFScannerPopDictionary (
   CGPDFScannerRef scanner,
   CGPDFDictionaryRef *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the CGPDFDictionary object popped from the `scanner` stack.

Return Value

Returns `true` if `value` is retrieved successfully; `false` otherwise.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGPDFScanner.h`

CGPDFScannerPopInteger

Retrieves an integer object from the scanner stack.

```
bool CGPDFScannerPopInteger (
   CGPDFScannerRef scanner,
   CGPDFInteger *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the CGPDFInteger object popped from the `scanner` stack.

Return Value

Returns `true` if `value` is retrieved successfully; `false` otherwise.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGPDFScanner.h`

CGPDFScannerPopName

Retrieves a character string from the scanner stack.

```
bool CGPDFScannerPopName (
   CGPDFScannerRef scanner,
   const char **value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the character string popped from the `scanner` stack.

Return Value

Returns `true` if `value` is retrieved successfully; `false` otherwise.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGPDFScanner.h`

CGPDFScannerPopNumber

Retrieves a real value object from the scanner stack.

```
bool CGPDFScannerPopNumber (
   CGPDFScannerRef scanner,
   CGPDFReal *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the CGPDFReal object popped from the `scanner` stack.

Return Value
Returns `true` if `value` is retrieved successfully; `false` otherwise.

Discussion
The number retrieved from the scanner can be a real value or an integer value. However, the result is always converted to a `CGPDFReal` data type.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFScanner.h`

CGPDFScannerPopObject

Retrieves an object from the scanner stack.

```
bool CGPDFScannerPopObject (
   CGPDFScannerRef scanner,
   CGPDFObjectRef *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the object popped from the `scanner` stack.

Return Value
Returns `true` if `value` is retrieved successfully; `false` otherwise.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFScanner.h`

CGPDFScannerPopStream

Retrieves a PDF stream object from the scanner stack.

```
bool CGPDFScannerPopStream (
   CGPDFScannerRef scanner,
   CGPDFStreamRef *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the CGPDFStream object popped from the `scanner` stack.

Return Value

Returns `true` if `value` is retrieved successfully; `false` otherwise.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGPDFScanner.h`

CGPDFScannerPopString

Retrieves a string object from the scanner stack.

```
bool CGPDFScannerPopString (
   CGPDFScannerRef scanner,
   CGPDFStringRef *value
);
```

Parameters

scanner

A valid scanner object.

value

On output, points to the CGPDFString object popped from the `scanner` stack.

Return Value

Returns `true` if `value` is retrieved successfully; `false` otherwise.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGPDFScanner.h`

CGPDFScannerRelease

Decrements the retain count of a scanner object.

```
void CGPDFScannerRelease (
   CGPDFScannerRef scanner
);
```

Parameters

scanner

The scanner object to release.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGPDFScanner.h

CGPDFScannerRetain

Increments the retain count of a scanner object.

```
CGPDFScannerRef CGPDFScannerRetain (
   CGPDFScannerRef scanner
);
```

Parameters

scanner

The scanner object to retain.

Return Value
The same scanner object passed to the function in the `scanner` parameter.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
CGPDFScanner.h

CGPDFScannerScan

Parses the content stream of a CGPDFScanner object.

```
bool CGPDFScannerScan (
   CGPDFScannerRef scanner
);
```

Parameters

scanner

The scanner object whose content stream you want to parse.

Return Value
Returns `true` if the entire stream is parsed successfully; `false` if parsing fails (for example, if the stream data is corrupted).

Discussion
The function `CGPDFScannerScan` parses the PDF content stream associated with the scanner. Each time Quartz parses a PDF operator for which you register a callback, Quartz invokes your callback.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CGPDFScanner.h`

Data Types

CGPDFScannerRef

An opaque type used to parse a PDF content stream.

```
typedef struct CGPDFScanner *CGPDFScannerRef;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGPDFScanner.h`

CGPDFStream Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFStream.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFStreamRef` opaque type represents a PDF stream. A PDF stream consists of a dictionary that describes a sequence of bytes. Streams typically represent objects with potentially large amounts of data, such as images and page descriptions.

This opaque type is not derived from CFType and therefore there are no functions for retaining and releasing it.

Functions

CGPDFStreamCopyData

Returns the data associated with a PDF stream.

```
CFDataRef CGPDFStreamCopyData (
   CGPDFStreamRef stream,
   CGPDFDataFormat *format
);
```

Parameters

stream

A PDF stream.

format

On return, contains a constant that specifies the format of the data returned—`CGPDFDataFormatRaw` (page 367), `CGPDFDataFormatJPEGEncoded` (page 367), or `CGPDFDataFormatJPEG2000` (page 367).

Return Value

A CFData object that contains a copy of the stream data. You are responsible for releasing this object.

Availability

Available in Mac OS X version 10.3 and later.

Declared In
CGPDFStream.h

CGPDFStreamGetDictionary

Returns the dictionary associated with a PDF stream.

```
CGPDFDictionaryRef CGPDFStreamGetDictionary (
   CGPDFStreamRef stream
);
```

Parameters

stream
A PDF stream.

Return Value
The PDF dictionary for the specified stream.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
CGPDFStream.h

Data Types

CGPDFStream

An opaque type that represents a PDF stream.

```
typedef struct CGPDFStream *CGPDFStreamRef;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGPDFStream.h

Constants

CGPDFDataFormat

The encoding format of PDF data.

```
enum CGPDFDataFormat {
 CGPDFDataFormatRaw,
 CGPDFDataFormatJPEGEncoded,
 CGPDFDataFormatJPEG2000
};
typedef enum CGPDFDataFormat CGPDFDataFormat;
```

Constants

`CGPDFDataFormatRaw`

The data stream is not encoded.

Available in Mac OS X v10.3 and later.

Declared in `CGPDFStream.h`.

`CGPDFDataFormatJPEGEncoded`

The data stream is encoded in JPEG format.

Available in Mac OS X v10.3 and later.

Declared in `CGPDFStream.h`.

`CGPDFDataFormatJPEG2000`

The data stream is encoded in JPEG-2000 format.

Available in Mac OS X v10.4 and later.

Declared in `CGPDFStream.h`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFStream.h`

CGPDFString Reference

Derived From:	None
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPDFString.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGPDFStringRef` opaque type represents a string in a PDF document A PDF string object of a series of bytes—unsigned integer values in the range 0 to 255. The string elements are not integer objects, but are stored in a more compact format. For more information on the representation of strings in PDF, see the latest version of *PDF Reference*, Adobe Systems Incorporated.

This opaque type is not derived from CFType and therefore there are no functions for retaining and releasing it. CGPDFString objects exist as constituent parts of a CGPDFDocument object, and are managed by their container.

Functions by Task

Converting PDF Strings

`CGPDFStringCopyTextString` (page 370)
Returns a CFString object that represents a PDF string as a text string.

`CGPDFStringCopyDate` (page 370)
Converts a string to a date.

Getting PDF String Data

`CGPDFStringGetBytePtr` (page 370)
Returns a pointer to the bytes of a PDF string.

`CGPDFStringGetLength` (page 371)
Returns the number of bytes in a PDF string.

Functions

CGPDFStringCopyDate

Converts a string to a date.

```
CFDateRef CGPDFStringCopyDate (
   CGPDFStringRef string
);
```

Parameters

string

The string to convert to a date.

Return Value

A CFDate object.

Discussion

The PDF specification defines a specific format for strings that represent dates. This function converts strings in that form to CFDate objects.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CGPDFString.h`

CGPDFStringCopyTextString

Returns a CFString object that represents a PDF string as a text string.

```
CFStringRef CGPDFStringCopyTextString (
   CGPDFStringRef string
);
```

Parameters

string

A PDF string. If this value is `NULL`, it will cause an error.

Return Value

Returns a CFString object that represents the specified PDF string as a text string. You are responsible for releasing this object.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFString.h`

CGPDFStringGetBytePtr

Returns a pointer to the bytes of a PDF string.

```
const unsigned char * CGPDFStringGetBytePtr (
   CGPDFStringRef string
);
```

Parameters

string

A PDF string.

Return Value

Returns a pointer to the bytes of the specified string. If the string is `NULL`, the function returns `NULL`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFString.h`

CGPDFStringGetLength

Returns the number of bytes in a PDF string.

```
size_t CGPDFStringGetLength (
   CGPDFStringRef string
);
```

Parameters

string

A PDF string.

Return Value

Returns the number of bytes referenced by the string, or `0` if the string is `NULL`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPDFString.h`

Data Types

CGPDFStringRef

An opaque data type that represents a string in a PDF document.

```
typedef struct CGPDFString *CGPDFStringRef;
```

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGPDFString.h`

CGPSConverter Reference

Derived From:	*CFType Reference*
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGPSConverter.h
Companion guide	Quartz 2D Programming Guide

Overview

`CGPSConverterRef` is an opaque type used to convert PostScript data to PDF data. The PostScript data is supplied by a data provider and written into a data consumer. When you create a PostScript converter object, you can supply callback functions for Quartz to invoke at various stages of the conversion process,

Functions

CGPSConverterAbort

Tells a PostScript converter to abort a conversion at the next available opportunity.

```
bool CGPSConverterAbort (
   CGPSConverterRef converter
);
```

Parameters

converter

A PostScript converter.

Return Value

A Boolean value that indicates whether the converter is currently converting data (`true` if it is).

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPSConverter.h`

CGPSConverterConvert

Uses a PostScript converter to convert PostScript data to PDF data.

```
bool CGPSConverterConvert (
   CGPSConverterRef converter,
   CGDataProviderRef provider,
   CGDataConsumerRef consumer,
   CFDictionaryRef options
);
```

Parameters

converter

A PostScript converter.

provider

A Quartz data provider that supplies PostScript data.

consumer

A Quartz data provider that will receive the resulting PDF data.

options

This parameter should be `NULL`; it is reserved for future expansion of the API.

Return Value

A Boolean value that indicates whether the PostScript conversion completed successfully (`true` if it did).

Discussion

The conversion is thread safe, however it is not possible to have more than one conversion job in process within a given address space or process. If a given thread is running a conversion and another thread starts a new conversion, the second conversion will block until the first conversion is complete.

Important: Although `CGPSConverterConvert` is thread safe (it uses locks to prevent more than one conversion at a time in the same process), it is not thread safe with respect to the Resource Manager. If your application uses the Resource Manager on a separate thread, you should either use locks to prevent `CGPSConverterConvert` from executing during your usage of the Resource Manager or you should perform your conversions using the Post Script converter in a separate process.

In general, you can avoid this issue by using nib files instead of Resource Manager resources.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CGPSConverter.h`

CGPSConverterCreate

Creates a new PostScript converter.

```
CGPSConverterRef CGPSConverterCreate (
   void *info,
   const CGPSConverterCallbacks *callbacks,
   CFDictionaryRef options
);
```

Parameters

info

A pointer to the data that will be passed to the callbacks.

callbacks
A pointer to a PostScript converter callbacks structure that specifies the callbacks to be used during a conversion process.

options
This parameter should be `NULL`; it is reserved for future expansion of the API.

Return Value
A new PostScript converter, or `NULL` if a converter could not be created. You are responsible for releasing this object.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPSConverter.h`

CGPSConverterGetTypeID

Returns the Core Foundation type identifier for PostScript converters.

```
CFTypeID CGPSConverterGetTypeID (
   void
);
```

Return Value
The Core Foundation identifier for the opaque type `CGPSConverterRef` (page 380).

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPSConverter.h`

CGPSConverterIsConverting

Checks whether the converter is currently converting data.

```
bool CGPSConverterIsConverting (
   CGPSConverterRef converter
);
```

Parameters
converter
A PostScript converter.

Return Value
Returns `true` that indicates if the conversion is in progress.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CGPSConverter.h`

Callbacks by Task

Performing Custom Tasks at the Document Level

`CGPSConverterBeginDocumentCallback` (page 376)
Performs custom tasks at the beginning of a PostScript conversion process.

`CGPSConverterEndDocumentCallback` (page 377)
Performs custom tasks at the end of a PostScript conversion process.

Performing Custom Tasks at the Page Level

`CGPSConverterBeginPageCallback` (page 377)
Performs custom tasks at the beginning of each page in a PostScript conversion process.

`CGPSConverterEndPageCallback` (page 378)
Performs custom tasks at the end of each page of a PostScript conversion process.

Reporting Progress and Messages

`CGPSConverterProgressCallback` (page 379)
Reports progress periodically during a PostScript conversion process.

`CGPSConverterMessageCallback` (page 378)
Passes messages generated during a PostScript conversion process.

Performing Custom Clean-up Tasks

`CGPSConverterReleaseInfoCallback` (page 380)
Performs custom tasks when a PostScript converter is released.

Callbacks

CGPSConverterBeginDocumentCallback

Performs custom tasks at the beginning of a PostScript conversion process.

```
typedef void (*CGPSConverterBeginDocumentCallback)(void
*info);
```

If you name your function `MyConverterBeginDocument`, you would declare it like this:

```
size_t MyConverterBeginDocument (
    void *info
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGPSConverterCreate` (page 374).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGPSConverter.h`

CGPSConverterBeginPageCallback

Performs custom tasks at the beginning of each page in a PostScript conversion process.

```
typedef void (*CGPSConverterBeginPageCallback)(void
*info, size_t pageNumber, CFDictionaryRef pageInfo);
```

If you name your function `MyConverterBeginDocument`, you would declare it like this:

```
void MyConverterBeginPage (
    void *info,
   size_t pageNumber,
   CFDictionaryRef pageInfo
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGPSConverterCreate` (page 374).

pageNumber

The current page number. Page numbers start at `1`.

pageInfo

A dictionary that contains contextual information about the page. This parameter is reserved for future API expansion, and is currently unused.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGPSConverter.h`

CGPSConverterEndDocumentCallback

Performs custom tasks at the end of a PostScript conversion process.

```
typedef void (*CGPSConverterEndDocumentCallback)(void
*info, bool success);
```

If you name your function `MyConverterEndDocument`, you would declare it like this:

```
void MyConverterEndDocument (
    void *info,
```

```
    bool success
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGPSConverterCreate` (page 374).

success

A Boolean value that indicates whether the PostScript conversion completed successfully (`true` if it did).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGPSConverter.h`

CGPSConverterEndPageCallback

Performs custom tasks at the end of each page of a PostScript conversion process.

```
typedef void (*CGPSConverterEndPageCallback)(void
*info, size_t pageNumber, CFDictionaryRef pageInfo);
```

If you name your function `MyConverterEndPage`, you would declare it like this:

```
void MyConverterEndPage (
   void *info,
   size_t *pageNumber,
   CFDictionaryRef pageInfo
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGPSConverterCreate` (page 374).

pageNumber

The current page number. Page numbers start at `1`.

pageInfo

A dictionary that contains contextual information about the page. This parameter is reserved for future API expansion, and is currently unused.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGPSConverter.h`

CGPSConverterMessageCallback

Passes messages generated during a PostScript conversion process.

```
typedef void (*CGPSConverterMessageCallback)(void
*info, CFStringRef message);
```

If you name your function `MyConverterMessage`, you would declare it like this:

```
void MyConverterMessage (
    void *info,
   CFStringRef message
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGPSConverterCreate` (page 374).

message

A string containing the message from the PostScript conversion process.

Discussion

There are several kinds of message that might be sent during a conversion process. The most likely are font substitution messages, and any messages that the PostScript code itself generates. Any PostScript messages written to `stdout` are routed through this callback—typically these are debugging or status messages and, although uncommon, can be useful in debugging. In addition, there may be error messages if the document is malformed.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGPSConverter.h`

CGPSConverterProgressCallback

Reports progress periodically during a PostScript conversion process.

```
typedef void (*CGPSConverterProgressCallback)(void
*info);
```

If you name your function `MyConverterProgress`, you would declare it like this:

```
void MyConverterProgress (
    void *info
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGPSConverterCreate` (page 374).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGPSConverter.h`

CGPSConverterReleaseInfoCallback

Performs custom tasks when a PostScript converter is released.

```
typedef void (*CGPSConverterReleaseInfoCallback)(void
*info);
```

If you name your function `MyConverterReleaseInfo`, you would declare it like this:

```
void MyConverterReleaseInfo (
  void *info
);
```

Parameters

info

A generic pointer to private data shared among your callback functions. This is the same pointer you supplied to `CGPSConverterCreate` (page 374).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGPSConverter.h`

Data Types

CGPSConverterRef

An opaque data type used to convert PostScript data to PDF data.

```
typedef struct CGPSConverter *CGPSConverterRef;
```

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGPSConverter.h`

CGPSConverterCallbacks

A structure for holding the callbacks provided when you create a PostScript converter object.

```
struct CGPSConverterCallbacks {
    unsigned int version;
    CGPSConverterBeginDocumentCallback beginDocument;
    CGPSConverterEndDocumentCallback endDocument;
    CGPSConverterBeginPageCallback beginPage;
    CGPSConverterEndPageCallback endPage;
    CGPSConverterProgressCallback noteProgress;
    CGPSConverterMessageCallback noteMessage;
    CGPSConverterReleaseInfoCallback releaseInfo;
};
typedef struct CGPSConverterCallbacks CGPSConverterCallbacks;
```

Fields

`version`
: The version number of the structure passed in as a parameter to the converter creation functions. The structure defined below is version `0`.

`beginDocument`
: The callback called at the beginning of the conversion of the PostScript document, or `NULL`.

`endDocument`
: The callback called at the end of conversion of the PostScript document, or `NULL`.

`beginPage`
: The callback called at the start of the conversion of each page in the PostScript document, or `NULL`.

`endPage`
: The callback called at the end of the conversion of each page in the PostScript document, or `NULL`.

`noteProgress`
: The callback called periodically during the conversion to indicate that conversion is proceeding, or `NULL`.

`noteMessage`
: The callback called to pass any messages that might result during the conversion, or `NULL`.

`releaseInfo`
: The callback called when the converter is deallocated, or `NULL`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGPSConverter.h`

CGShading Reference

Derived From:	CFType
Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGShading.h
Companion guide	Quartz 2D Programming Guide

Overview

`CGShadingRef` is an opaque type used to define linear (axial) and radial gradient fills whose color transitions are controlled by a function (`CGFunctionRef` (page 193)) that you provide. Shading means to fill using a smooth transition between colors across an area. To paint with a Quartz shading, you call `CGContextDrawShading` (page 84). This function fills the current clipping path using the specified color gradient, calling your parametric function repeatedly as it draws

An alternative to using a CGShading object is to use the `CGGradientRef` (page 201) opaque type. For applications that run in Mac OS X v10.5 and later, CGGradient objects are much simpler to use. (See *CGGradient Reference*.)

Functions by Task

Creating Shading Objects

`CGShadingCreateAxial` (page 384)
Creates a shading object to use for axial shading.

`CGShadingCreateRadial` (page 385)
Creates a shading object to use for radial shading.

Retaining and Releasing Shading Objects

`CGShadingRetain` (page 386)
Increments the retain count of a shading object.

`CGShadingRelease` (page 386)
Decrements the retain count of a shading object.

Getting the CFType ID

CGShadingGetTypeID (page 385)
Returns the Core Foundation type identifier for Quartz shading objects.

Functions

CGShadingCreateAxial

Creates a shading object to use for axial shading.

```
CGShadingRef CGShadingCreateAxial (
   CGColorSpaceRef colorspace,
   CGPoint start,
   CGPoint end,
   CGFunctionRef function,
   bool extendStart,
   bool extendEnd
);
```

Parameters

colorspace
The color space in which color values are expressed. Quartz retains this object; upon return, you may safely release it.

start
The starting point of the axis, in the shading's target coordinate space.

end
The ending point of the axis, in the shading's target coordinate space.

function
A CGFunction object created by the function CGFunctionCreate. This object refers to your function for creating an axial shading. Quartz retains this object; upon return, you may safely release it.

extendStart
A Boolean value that specifies whether to extend the shading beyond the starting point of the axis.

extendEnd
A Boolean value that specifies whether to extend the shading beyond the ending point of the axis.

Return Value
A new Quartz axial shading. You are responsible for releasing this object using CGShadingRelease (page 386).

Discussion
An axial shading is a color blend that varies along a linear axis between two endpoints and extends indefinitely perpendicular to that axis. When you are ready to draw the shading, call the function CGContextDrawShading (page 84).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CGShading.h

CGShadingCreateRadial

Creates a shading object to use for radial shading.

```
CGShadingRef CGShadingCreateRadial (
   CGColorSpaceRef colorspace,
   CGPoint start,
   CGFloat startRadius,
   CGPoint end,
   CGFloat endRadius,
   CGFunctionRef function,
   bool extendStart,
   bool extendEnd
);
```

Parameters

colorspace
The color space in which color values are expressed. Quartz retains this object; upon return, you may safely release it.

start
The center of the starting circle, in the shading's target coordinate space.

startRadius
The radius of the starting circle, in the shading's target coordinate space.

end
The center of the ending circle, in the shading's target coordinate space.

endRadius
The radius of the ending circle, in the shading's target coordinate space.

function
A CGFunction object created by the function `CGFunctionCreate`. This object refers to your function for creating a radial shading. Quartz retains this object; upon return, you may safely release it.

extendStart
A Boolean value that specifies whether to extend the shading beyond the starting circle.

extendEnd
A Boolean value that specifies whether to extend the shading beyond the ending circle.

Return Value

A new Quartz radial shading. You are responsible for releasing this object using `CGShadingRelease` (page 386).

Discussion

A radial shading is a color blend that varies between two circles. To draw the shading, call the function `CGContextDrawShading` (page 84).

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CGShading.h`

CGShadingGetTypeID

Returns the Core Foundation type identifier for Quartz shading objects.

```
CFTypeID CGShadingGetTypeID (
   void
);
```

Return Value
The Core Foundation identifier for the opaque type `CGShadingRef` (page 387).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGShading.h`

CGShadingRelease

Decrements the retain count of a shading object.

```
void CGShadingRelease (
   CGShadingRef shading
);
```

Parameters

shading
The shading object to release.

Discussion
This function is equivalent to `CFRelease`, except that it does not cause an error if the *shading* parameter is `NULL`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGShading.h`

CGShadingRetain

Increments the retain count of a shading object.

```
CGShadingRef CGShadingRetain (
   CGShadingRef shading
);
```

Parameters

shading
The shading object to retain.

Return Value
The same shading object you passed in as the `shading` parameter.

Discussion
This function is equivalent to `CFRetain`, except that it does not cause an error if the `shading` parameter is `NULL`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CGShading.h`

Data Types

CGShadingRef

An opaque type that represents a Quartz shading.

```
typedef struct CGShading *CGShadingRef;
```

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGShading.h`

Managers

Quartz Display Services Reference

Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGDirectDisplay.h CGDirectPalette.h CGDisplayConfiguration.h CGDisplayFade.h CGError.h CGRemoteOperation.h CGSession.h CGWindowLevel.h
Companion guide	Quartz Display Services Programming Topics

Overview

Note: This document was previously titled *Quartz Services Reference*. Some information related to low-level events has been moved from this document into *Quartz Event Services Reference*.

Quartz Display Services provides direct access to certain low-level features in the Mac OS X window server related to the configuration and control of display hardware. For example, you can use Quartz Display Services to:

- Examine and change display mode properties such as width, height, and pixel depth
- Configure a set of displays in a single operation
- Capture one or more displays for exclusive use
- Perform fade effects
- Activate display mirroring
- Configure gamma color correction tables and color palettes
- Receive notification of screen update operations

Functions by Task

Finding Displays

`CGMainDisplayID` (page 444)
Returns the display ID of the main display.

`CGGetOnlineDisplayList` (page 443)
Provides a list of displays that are online (active, mirrored, or sleeping).

`CGGetActiveDisplayList` (page 437)
Provides a list of displays that are active (or drawable).

`CGGetDisplaysWithOpenGLDisplayMask` (page 438)
Provides a list of displays that corresponds to the bits set in an OpenGL display mask.

`CGGetDisplaysWithPoint` (page 439)
Provides a list of online displays with bounds that include the specified point.

`CGGetDisplaysWithRect` (page 440)
Gets a list of online displays with bounds that intersect the specified rectangle.

`CGOpenGLDisplayMaskToDisplayID` (page 444)
Maps an OpenGL display mask to a display ID.

`CGDisplayIDToOpenGLDisplayMask` (page 420)
Maps a display ID to an OpenGL display mask.

Capturing and Releasing Displays

`CGDisplayCapture` (page 415)
Captures a display for exclusive use by an application.

`CGDisplayCaptureWithOptions` (page 415)
Captures a display for exclusive use by an application, using the specified options.

`CGDisplayRelease` (page 429)
Releases a captured display.

`CGDisplayIsCaptured` (page 423)
Returns a Boolean value indicating whether a display is captured.

`CGCaptureAllDisplays` (page 400)
Captures all attached displays.

`CGCaptureAllDisplaysWithOptions` (page 401)
Captures all attached displays, using the specified options.

`CGReleaseAllDisplays` (page 451)
Releases all captured displays.

`CGShieldingWindowID` (page 457)
Returns the window ID of the shield window for a captured display.

`CGShieldingWindowLevel` (page 458)
Returns the window level of the shield window for a captured display.

`CGDisplayAddressForPosition` (page 407)
Returns the address in frame buffer memory that corresponds to a position on an online display.

`CGDisplayBaseAddress` (page 409)
Returns the base address in frame buffer memory of an online display.

`CGDisplayGetDrawingContext` (page 419)
Returns a graphics context suitable for drawing to a captured display.

Configuring Displays

`CGBeginDisplayConfiguration` (page 399)
Begins a new set of display configuration changes.

`CGCancelDisplayConfiguration` (page 400)
Cancels a set of display configuration changes.

`CGCompleteDisplayConfiguration` (page 401)
Completes a set of display configuration changes.

`CGConfigureDisplayMirrorOfDisplay` (page 403)
Changes the configuration of a mirroring set.

`CGConfigureDisplayMode` (page 404)
Configures the display mode of a display.

`CGConfigureDisplayOrigin` (page 405)
Configures the origin of a display in global display (desktop) coordinates.

`CGRestorePermanentDisplayConfiguration` (page 453)
Restores the permanent display configuration settings for the current user.

`CGConfigureDisplayStereoOperation` (page 406)
Enables or disables stereo operation for a display, as part of a display configuration.

`CGDisplaySetStereoOperation` (page 433)
Immediately enables or disables stereo operation for a display.

Getting the Display Configuration

`CGDisplayCopyColorSpace` (page 416)
Returns the color space for a display.

`CGDisplayIOServicePort` (page 420)
Returns the I/O Kit service port of the specified display.

`CGDisplayIsActive` (page 421)
Returns a Boolean value indicating whether a display is active.

`CGDisplayIsAlwaysInMirrorSet` (page 421)
Returns a Boolean value indicating whether a display is always in a mirroring set.

`CGDisplayIsAsleep` (page 422)
Returns a Boolean value indicating whether a display is sleeping (and is therefore not drawable.)

`CGDisplayIsBuiltin` (page 422)
Returns a Boolean value indicating whether a display is built-in, such as the internal display in portable systems.

`CGDisplayIsInHWMirrorSet` (page 423)
Returns a Boolean value indicating whether a display is in a hardware mirroring set.

`CGDisplayIsInMirrorSet` (page 424)
Returns a Boolean value indicating whether a display is in a mirroring set.

`CGDisplayIsMain` (page 424)
Returns a Boolean value indicating whether a display is the main display.

`CGDisplayIsOnline` (page 425)
Returns a Boolean value indicating whether a display is connected or online.

`CGDisplayIsStereo` (page 425)
Returns a Boolean value indicating whether a display is running in a stereo graphics mode.

`CGDisplayMirrorsDisplay` (page 426)
For a secondary display in a mirroring set, returns the primary display.

`CGDisplayModelNumber` (page 426)
Returns the model number of a display monitor.

`CGDisplayPrimaryDisplay` (page 428)
Returns the primary display in a hardware mirroring set.

`CGDisplayRotation` (page 430)
Returns the rotation angle of a display in degrees.

`CGDisplayScreenSize` (page 431)
Returns the width and height of a display in millimeters.

`CGDisplaySerialNumber` (page 432)
Returns the serial number of a display monitor.

`CGDisplayUnitNumber` (page 435)
Returns the logical unit number of a display.

`CGDisplayUsesOpenGLAcceleration` (page 436)
Returns a Boolean value indicating whether Quartz is using OpenGL-based window acceleration (Quartz Extreme) to render in a display.

`CGDisplayVendorNumber` (page 436)
Returns the vendor number of the specified display's monitor.

Registering for Notification of Display Configuration Changes

These functions are used to register and unregister a callback function for notification of display configuration changes.

`CGDisplayRegisterReconfigurationCallback` (page 429)
Registers a callback function to be invoked whenever a local display is reconfigured.

`CGDisplayRemoveReconfigurationCallback` (page 430)
Removes the registration of a callback function that's invoked whenever a local display is reconfigured.

Retrieving Display Parameters

`CGDisplayBounds` (page 413)
Returns the bounds of a display in global display space.

`CGDisplayPixelsHigh` (page 427)
Returns the display height in pixel units.

`CGDisplayPixelsWide` (page 428)
Returns the display width in pixel units.

`CGDisplayBitsPerPixel` (page 413)
Returns the number of bits used to represent a pixel in the frame buffer.

`CGDisplayBitsPerSample` (page 413)
Returns the number of bits used to represent a pixel component in the frame buffer.

`CGDisplaySamplesPerPixel` (page 431)
Returns the number of color components used to represent a pixel.

`CGDisplayBytesPerRow` (page 414)
Returns the number of bytes per row in a display.

Using Display Modes

`CGDisplayAvailableModes` (page 408)
Returns information about the currently available display modes.

`CGDisplayBestModeForParameters` (page 410)
Returns information about the display mode closest to a specified depth and screen size.

`CGDisplayBestModeForParametersAndRefreshRate` (page 410)
Returns information about the display mode closest to a specified depth, screen size, and refresh rate.

`CGDisplayBestModeForParametersAndRefreshRateWithProperty` (page 412)
Returns information about the display mode closest to a specified depth, screen size, and refresh rate, with a required property.

`CGDisplayCurrentMode` (page 416)
Returns information about the current display mode.

`CGDisplaySwitchToMode` (page 434)
Switches a display to a different mode.

Adjusting the Display Gamma

`CGSetDisplayTransferByFormula` (page 455)
Sets the gamma function for a display, by specifying the coefficients of the gamma transfer formula.

`CGGetDisplayTransferByFormula` (page 440)
Gets the coefficients of the gamma transfer formula for a display.

`CGSetDisplayTransferByTable` (page 457)
Sets the color gamma function for a display, by specifying the values in the RGB gamma tables.

`CGGetDisplayTransferByTable` (page 442)
Gets the values in the RGB gamma tables for a display.

`CGSetDisplayTransferByByteTable` (page 454)
Sets the byte values in the 8-bit RGB gamma tables for a display.

`CGDisplayRestoreColorSyncSettings` (page 430)
Restores the gamma tables to the values in the user's ColorSync display profile.

`CGDisplayGammaTableCapacity` (page 419)
Returns the capacity, or number of entries, in the gamma table for a display.

Working With Color Palettes

`CGPaletteCreateDefaultColorPalette` (page 445)
Returns a new display palette representing the default 8-bit color palette.

`CGPaletteCreateFromPaletteBlendedWithColor` (page 446)
Returns a new tinted display palette. The new palette is derived from an existing palette blended with a solid color, at a specified level of intensity.

`CGPaletteCreateWithByteSamples` (page 446)
Returns a new display palette using 8-bit sample data.

`CGPaletteCreateWithCapacity` (page 447)
Returns a new display palette with a specified capacity. The new palette is initialized from the default color palette.

`CGPaletteCreateWithDisplay` (page 447)
Returns a copy of the current palette for a display.

`CGPaletteCreateWithSamples` (page 447)
Returns a new display palette using RGB sample data.

`CGPaletteCreateCopy` (page 445)
Returns a copy of a specified display palette.

`CGPaletteRelease` (page 450)
Decrements the retain count of a display palette.

`CGPaletteGetColorAtIndex` (page 448)
Returns the color value at the specified index.

`CGPaletteGetIndexForColor` (page 448)
Returns the index of the display palette entry that most closely matches a specified color value.

`CGPaletteGetNumberOfSamples` (page 449)
Returns the number of colors in a display palette.

`CGPaletteIsEqualToPalette` (page 449)
Returns a Boolean value indicating whether two display palettes are equal.

`CGPaletteSetColorAtIndex` (page 450)
Updates the color value at the specified index in a display palette.

`CGDisplayCanSetPalette` (page 414)
Returns a Boolean value indicating whether the current display mode supports palettes.

`CGDisplaySetPalette` (page 432)
Sets the palette for a display.

Display Fade Effects

`CGConfigureDisplayFadeEffect` (page 402)
Modifies the settings of the built-in fade effect that occurs during a display configuration.

`CGAcquireDisplayFadeReservation` (page 398)
Reserves the fade hardware for a specified time interval.

`CGDisplayFade` (page 417)
Performs a single fade operation.

`CGDisplayFadeOperationInProgress` (page 418)
Returns a Boolean value indicating whether a fade operation is currently in progress.

`CGReleaseDisplayFadeReservation` (page 452)
Releases a display fade reservation, and unfades the display if needed.

Beam Position

These functions are advisory in nature and depend on IO Kit and hardware-specific drivers to implement support. If you need extremely precise timing, or access to vertical blanking interrupts, you should consider writing a device driver to tie into hardware-specific capabilities.

`CGDisplayBeamPosition` (page 409)
Returns the current beam position on a display.

`CGDisplayWaitForBeamPositionOutsideLines` (page 437)
Waits until the beam position moves outside a region in a display screen. This function is not designed for VBL drawing synchronization.

Controlling the Mouse Cursor

`CGDisplayHideCursor` (page 419)
Hides the mouse cursor, and increments the hide cursor count.

`CGDisplayShowCursor` (page 434)
Decrements the hide cursor count, and shows the mouse cursor if the count is zero.

`CGDisplayMoveCursorToPoint` (page 427)
Moves the mouse cursor to a specified point relative to the display origin (the upper left corner of the display).

`CGCursorIsVisible` (page 407)
Returns a Boolean value indicating whether the mouse cursor is visible.

`CGCursorIsDrawnInFramebuffer` (page 406)
Returns a Boolean value indicating whether the mouse cursor is drawn in frame buffer memory.

`CGAssociateMouseAndMouseCursorPosition` (page 399)
Connects or disconnects the mouse and cursor while an application is in the foreground.

`CGWarpMouseCursorPosition` (page 461)
Moves the mouse cursor without generating events.

`CGGetLastMouseDelta` (page 442)
Reports the change in mouse position since the last mouse movement event received by the application.

Getting Window Server Information

`CGSessionCopyCurrentDictionary` (page 454)
Returns information about the caller's window server session.

`CGWindowServerCFMachPort` (page 462)
Returns a Core Foundation mach port (CFMachPort) that corresponds to the Mac OS X window server.

`CGWindowLevelForKey` (page 461)
Returns the window level that corresponds to one of the standard window types.

Getting Information About Refresh and Move Operations

You can use these functions to find out what areas on local displays are changing their appearance as the result of operations such as drawing, window movement or scrolling, and display reconfiguration.

`CGRegisterScreenRefreshCallback` (page 451)
Registers a callback function to be invoked when local displays are refreshed or modified.

`CGUnregisterScreenRefreshCallback` (page 458)
Removes a previously registered callback function invoked when local displays are refreshed or modified.

`CGWaitForScreenRefreshRects` (page 459)
Waits for screen refresh operations.

`CGScreenRegisterMoveCallback` (page 453)
Registers a callback function to be invoked when an area of the display is moved.

`CGScreenUnregisterMoveCallback` (page 454)
Removes a previously registered callback function invoked when an area of the display is moved.

`CGWaitForScreenUpdateRects` (page 460)
Waits for screen update operations.

`CGReleaseScreenRefreshRects` (page 452)
Deallocates a list of rectangles that represent changed areas on local displays.

Functions

CGAcquireDisplayFadeReservation

Reserves the fade hardware for a specified time interval.

```
CGError CGAcquireDisplayFadeReservation (
   CGDisplayReservationInterval seconds,
   CGDisplayFadeReservationToken *pNewToken
);
```

Parameters

seconds
The desired number of seconds to reserve the fade hardware. An application can specify any value in the interval `(0, kCGMaxDisplayReservationInterval]`.

pNewToken
A pointer to storage (provided by the caller) for a fade reservation token. On return, the storage contains a new token.

Return Value
Returns `kCGErrorNoneAvailable` if another fade reservation is in effect. Otherwise, returns `kCGErrorSuccess`.

Discussion
Before performing a fade operation, an application must reserve the fade hardware for a specified period of time. Quartz returns a token that represents a new fade reservation. The application uses this token as an argument in subsequent calls to other display fade functions.

During the fade reservation interval, the application has exclusive rights to use the fade hardware. At the end of the interval, the token becomes invalid and the hardware automatically returns to a normal state. Typically the application calls `CGReleaseDisplayFadeReservation` (page 452) to release the fade reservation before it expires.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGAssociateMouseAndMouseCursorPosition

Connects or disconnects the mouse and cursor while an application is in the foreground.

```
CGError CGAssociateMouseAndMouseCursorPosition (
   boolean_t connected
);
```

Parameters

connected
Pass `true` if the mouse and cursor should be connected; otherwise, pass `false`.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
When you call this function to disconnect the cursor and mouse, all events received by your application have a constant absolute location but contain mouse delta (change in X and Y) data. You may hide the cursor or change it into something appropriate for your application. You can reposition the cursor by using the function `CGDisplayMoveCursorToPoint` (page 427) or the function `CGWarpMouseCursorPosition` (page 461).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGRemoteOperation.h`

CGBeginDisplayConfiguration

Begins a new set of display configuration changes.

```
CGError CGBeginDisplayConfiguration (
   CGDisplayConfigRef *pConfigRef
);
```

Parameters

pConfigRef

A pointer to storage you provide for a display configuration. On return, your storage contains a new display configuration.

Return Value

A result code. If the object is successfully created, the result is `kCGErrorSuccess`. For other possible values, see "Quartz Display Services Result Codes" (page 486).

Discussion

This function creates a display configuration object that provides a context for a set of display configuration changes. After you specify the desired changes, you use `CGCompleteDisplayConfiguration` (page 401) to apply them in a single transaction.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGCancelDisplayConfiguration

Cancels a set of display configuration changes.

```
CGError CGCancelDisplayConfiguration (
   CGDisplayConfigRef configRef
);
```

Parameters

configRef

The display configuration to cancel. On return, the configuration is cancelled and is no longer valid.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

This function is used to abandon a display configuration. As a side effect, the display configuration object is released.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGCaptureAllDisplays

Captures all attached displays.

```
CGDisplayErr CGCaptureAllDisplays (
   void
);
```

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
This function captures all attached displays in a single operation. This operation provides an immersive environment for your application, and it prevents other applications from trying to adjust to display changes.

Availability
Available in Mac OS X v10.0 and later.

See Also
`CGDisplayCapture` (page 415)

Declared In
`CGDirectDisplay.h`

CGCaptureAllDisplaysWithOptions

Captures all attached displays, using the specified options.

```
CGDisplayErr CGCaptureAllDisplaysWithOptions (
   CGCaptureOptions options
);
```

Parameters

options
The options to use. See "Display Capture Options" (page 475).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
This function allows you to specify one or more options to use during capture of all attached displays.

Availability
Available in Mac OS X v10.3 and later.

See Also
`CGCaptureAllDisplays` (page 400)

Declared In
`CGDirectDisplay.h`

CGCompleteDisplayConfiguration

Completes a set of display configuration changes.

```
CGError CGCompleteDisplayConfiguration (
   CGDisplayConfigRef configRef,
   CGConfigureOption option
);
```

Parameters

configRef

The display configuration with the desired changes. On return, this configuration is no longer valid.

option

The scope of the display configuration changes. Pass one of the constants listed in "Display Configuration Scopes" (page 477).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
This function applies a set of display configuration changes as a single atomic transaction. The duration or scope of the changes depends on the value of the `option` parameter. The possible scopes are fully described in "Display Configuration Scopes" (page 477).

A configuration change may fail if an unsupported display mode is requested, or if another application is running in full-screen mode.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGConfigureDisplayFadeEffect

Modifies the settings of the built-in fade effect that occurs during a display configuration.

```
CGError CGConfigureDisplayFadeEffect (
   CGDisplayConfigRef configRef,
   CGDisplayFadeInterval fadeOutSeconds,
   CGDisplayFadeInterval fadeInSeconds,
   float fadeRed,
   float fadeGreen,
   float fadeBlue
);
```

Parameters

configRef

A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 399).

fadeOutSeconds

The time in seconds to fade from the normal display to the specified fade color. The fade out is completed before the display configuration is changed. If the interval is 0, Quartz applies the color immediately.

fadeInSeconds

Time in seconds to return from the specified fade color to the normal display. The fade-in is run asynchronously after the display configuration is changed.

fadeRed
An intensity value in the interval [0, 1] that represents the red component of the desired blend color.

fadeGreen
An intensity value in the interval [0, 1] that represents the green component of the desired blend color.

fadeBlue
An intensity value in the interval [0, 1] that represents the blue component of the desired blend color.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
This function provides a way to customize the built-in fade effect that Quartz performs when displays are reconfigured. The default time settings for this fade effect are 0.3 seconds to fade out, and 0.5 seconds to fade back in. The default fade color is French Blue for a normal desktop, and black for a captured display.

Before using this function, you need to call `CGBeginDisplayConfiguration` (page 399) to acquire the display configuration token for the desired display. No fade reservation is needed—when you call `CGCompleteDisplayConfiguration` (page 401), Quartz reserves the fade hardware (assuming it is available) and performs the fade.

Calling this function modifies the fade behavior for a single display configuration, and has no permanent effect.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGConfigureDisplayMirrorOfDisplay

Changes the configuration of a mirroring set.

```
CGError CGConfigureDisplayMirrorOfDisplay (
   CGDisplayConfigRef configRef,
   CGDirectDisplayID display,
   CGDirectDisplayID masterDisplay
);
```

Parameters

configRef
A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 399).

display
The display to add to a mirroring set.

masterDisplay
A display in a mirroring set, or `kCGNullDirectDisplay` to disable mirroring. To specify the main display, use `CGMainDisplayID` (page 444).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
Display mirroring and display matte generation are implemented either in hardware (preferred) or software, at the discretion of the device driver.

- Hardware mirroring

  With hardware mirroring enabled, all drawing is directed to the primary display—see `CGDisplayPrimaryDisplay` (page 428).

  If the device driver selects hardware matte generation, the display bounds and rowbytes values are adjusted to reflect the active drawable area.

- Software mirroring

  In this form of mirroring, identical content is drawn into each display in the mirroring set. Applications that use the window system need not be concerned about mirroring, as the window system takes care of all flushing of window content to the appropriate displays.

  Applications that draw directly to the display, as with display capture, must make sure to draw the same content to all mirrored displays in a software mirror set. When drawing to software mirrored displays using a full screen OpenGL context (not drawing through a window), you should create shared OpenGL contexts for each display and re-render for each display.

You can use the function `CGGetActiveDisplayList` (page 437) to determine which displays are active, or drawable. This automatically gives your application the correct view of the current displays.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGConfigureDisplayMode

Configures the display mode of a display.

```
CGError CGConfigureDisplayMode (
   CGDisplayConfigRef configRef,
   CGDirectDisplayID display,
   CFDictionaryRef mode
);
```

Parameters

configRef
A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 399).

display
The display being configured.

mode
A display mode dictionary (see the discussion below).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
A display mode is a set of properties such as width, height, pixel depth, and refresh rate, and options such as stretched LCD panel filling.

The display mode you provide must be one of the following:

- A dictionary returned by one of the `CGDisplayBestMode` functions, such as `CGDisplayBestModeForParameters` (page 410).
- A dictionary in the array returned by `CGDisplayAvailableModes` (page 408).

If you use this function to change the mode of a display in a mirroring set, Quartz may adjust the bounds, resolutions, and depth of the other displays in the set to a safe mode, with matching depth and the smallest enclosing size.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGConfigureDisplayOrigin

Configures the origin of a display in global display (desktop) coordinates.

```
CGError CGConfigureDisplayOrigin (
   CGDisplayConfigRef configRef,
   CGDirectDisplayID display,
   CGDisplayCoord x,
   CGDisplayCoord y
);
```

Parameters

configRef
A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 399).

display
The display being configured.

x
The desired x-coordinate for the upper left corner of the display.

y
The desired y-coordinate for the upper left corner of the display.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
In Quartz, the upper left corner of a display is called the origin. The origin of a display is always specified in global display (desktop) coordinates. The origin of the main or primary display is (0,0).

The new origin is placed as close as possible to the requested location, without overlapping or leaving a gap between displays.

If you use this function to change the origin of a mirrored display, the display may be removed from the mirroring set.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGConfigureDisplayStereoOperation

Enables or disables stereo operation for a display, as part of a display configuration.

```
CGError CGConfigureDisplayStereoOperation (
   CGDisplayConfigRef configRef,
   CGDirectDisplayID display,
   boolean_t stereo,
   boolean_t forceBlueLine
);
```

Parameters

configRef
A display configuration, acquired by calling `CGBeginDisplayConfiguration` (page 399).

display
The display being configured.

stereo
Pass `true` if you want to enable stereo operation. To disable it, pass `false`.

forceBlueLine
When in stereo operation, a display may need to generate a special stereo sync signal as part of the video output. The sync signal consists of a blue line which occupies the first 25% of the last scanline for the left eye view, and the first 75% of the last scanline for the right eye view. The remainder of the scanline is black. To force the display to generate this sync signal, pass `true`; otherwise, pass `false`.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
The system normally detects the presence of a stereo window and automatically switches a display containing a stereo window to stereo operation. This function provides a mechanism to force a display to stereo operation, and to set options (blue line sync signal) when in stereo operation.

On success, the display resolution, mirroring mode, and available display modes may change due to hardware-specific capabilities and limitations. You should check these settings to verify that they are appropriate for your application.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGDisplayConfiguration.h`

CGCursorIsDrawnInFramebuffer

Returns a Boolean value indicating whether the mouse cursor is drawn in frame buffer memory.

```
boolean_t CGCursorIsDrawnInFramebuffer (
   void
);
```

Return Value
If `true`, the cursor is drawn in frame buffer memory; otherwise, `false`.

Discussion
This function returns a Boolean value that indicates whether or not the cursor is drawn in the frame buffer. (The cursor could exist in an overlay plane or a similar mechanism that puts pixels on-screen without altering frame buffer content.) If the cursor is drawn in the frame buffer, it is read back along with window data.

The reported Boolean value is based on the union of the state of the cursor on all displays. If the cursor is drawn in the frame buffer on any display, the function returns `true`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGRemoteOperation.h`

CGCursorIsVisible

Returns a Boolean value indicating whether the mouse cursor is visible.

```
boolean_t CGCursorIsVisible (
   void
);
```

Return Value
If `true`, the cursor is visible on any display; otherwise, `false`.

Discussion
To hide or show the cursor, you can use the functions `CGDisplayHideCursor` (page 419) and `CGDisplayShowCursor` (page 434).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGRemoteOperation.h`

CGDisplayAddressForPosition

Returns the address in frame buffer memory that corresponds to a position on an online display.

```
void * CGDisplayAddressForPosition (
   CGDirectDisplayID display,
   CGDisplayCoord x,
   CGDisplayCoord y
);
```

Parameters

display

The display to access.

x

The x-coordinate of a position in global display space. The origin is the upper left corner of the main display.

y

The y-coordinate of a position in global display space. The origin is the upper left corner of the main display, and the y-axis is oriented down.

Return Value

The address in frame buffer memory that corresponds to the specified position. If the display ID is invalid or the point lies outside the bounds of the display, the return value is `NULL`.

Discussion

If the display has not been captured, the returned address may refer to read-only memory.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayAvailableModes

Returns information about the currently available display modes.

```
CFArrayRef CGDisplayAvailableModes (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

An array of dictionaries with display mode information, or `NULL` if the display is invalid. The array is owned by the system and you should not release it. Each dictionary in the array contains information about a mode that the display supports. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 479) and "Display Mode Optional Properties" (page 480). For general information about using dictionaries, see *CFDictionary Reference*.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QTCarbonShell

Declared In
CGDirectDisplay.h

CGDisplayBaseAddress

Returns the base address in frame buffer memory of an online display.

```
void * CGDisplayBaseAddress (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The base address in frame buffer memory of the specified display. If the display ID is invalid, the return value is NULL.

Discussion
If the display has not been captured, the returned address may refer to read-only memory.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDisplayBeamPosition

Returns the current beam position on a display.

```
CGBeamPosition CGDisplayBeamPosition (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The current beam position on the specified display. If the display does not implement conventional video vertical and horizontal sweep in painting, or the driver does not implement this functionality, 0 is returned.

Discussion
This function returns the number of the scan line on which the beam is currently positioned, expressed as a non-negative integer. The value increases as the beam moves lower on the display.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDisplayBestModeForParameters

Returns information about the display mode closest to a specified depth and screen size.

```
CFDictionaryRef CGDisplayBestModeForParameters (
   CGDirectDisplayID display,
   size_t bitsPerPixel,
   size_t width,
   size_t height,
   boolean_t *exactMatch
);
```

Parameters

display

The display to optimize.

bitsPerPixel

Optimal display depth in bits per pixel. Note that this value is not the same as pixel depth, which is the number of bits per channel or component.

width

Optimal display width in pixel units.

height

Optimal display height in pixel units.

exactMatch

A pointer to a Boolean variable. On return, its value is `true` if an exact match in display depth, width, and height is found; otherwise, `false`. If this information is not needed, pass `NULL`.

Return Value

A display mode dictionary, or `NULL` if the display is invalid. The dictionary is owned by the system and you should not release it. The dictionary contains information about the display mode closest to the specified depth and screen size. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 479) and "Display Mode Optional Properties" (page 480). For general information about using dictionaries, see *CFDictionary Reference*.

Discussion

This function tries to find an optimal display mode for the specified display. The function first tries to find a mode with the specified pixel depth and dimensions equal to or greater than the specified width and height. If no depth match is found, it tries to find a mode with greater depth and the same or greater dimensions. If a suitable display mode is not found, this function simply returns the current display mode.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayBestModeForParametersAndRefreshRate

Returns information about the display mode closest to a specified depth, screen size, and refresh rate.

```
CFDictionaryRef CGDisplayBestModeForParametersAndRefreshRate (
   CGDirectDisplayID display,
   size_t bitsPerPixel,
   size_t width,
   size_t height,
   CGRefreshRate refresh,
   boolean_t *exactMatch
);
```

Parameters

display

The display to access.

bitsPerPixel

Optimal display depth, in bits per pixel. Note that this value is not the same as pixel depth, which is the number of bits per channel or component.

width

Optimal display width, in pixel units.

height

Optimal display height, in pixel units.

refresh

Optimal display refresh rate, in frames per second.

exactMatch

A pointer to a Boolean variable. On return, its value is `true` if an exact match in display depth, width, height, and refresh rate is found; otherwise, `false`. If this information is not needed, pass `NULL`.

Return Value

A display mode dictionary, or `NULL` if the display is invalid. The dictionary is owned by the system and you should not release it. The dictionary contains information about the display mode closest to the specified depth, screen size, and refresh rate. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 479) and "Display Mode Optional Properties" (page 480). For general information about using dictionaries, see *CFDictionary Reference*.

Discussion

This function searches the list of available display modes for a mode that comes closest to satisfying these criteria:

- Has a pixel depth equal to or greater than the specified depth
- Has dimensions equal to or greater than the specified height and width
- Uses a refresh rate equal to or near the specified rate

If a suitable display mode is not found, this function simply returns the current display mode.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayBestModeForParametersAndRefreshRateWithProperty

Returns information about the display mode closest to a specified depth, screen size, and refresh rate, with a required property.

```
CFDictionaryRef CGDisplayBestModeForParametersAndRefreshRateWithProperty (
   CGDirectDisplayID display,
   size_t bitsPerPixel,
   size_t width,
   size_t height,
   CGRefreshRate refresh,
   CFStringRef property,
   boolean_t *exactMatch
);
```

Parameters

display

The display to access.

bitsPerPixel

Optimal display depth, in bits per pixel. Note that this value is not the same as pixel depth, which is the number of bits per channel or component.

width

Optimal display width, in pixels.

height

Optimal display height, in pixels.

refresh

Optimal display refresh rate, in refreshes per second.

property

A required display mode property. For a list of the properties you can specify, see "Display Mode Optional Properties" (page 480).

exactMatch

A pointer to a Boolean variable. On return, its value is `true` if an exact match in display depth, width, height, refresh rate, and property is found; otherwise, `false`. If this information is not needed, pass `NULL`.

Return Value

A display mode dictionary, or `NULL` if the display is invalid. The dictionary is owned by the system and you should not release it. The dictionary contains information about the display mode with the specified property that comes closest to the specified depth, screen size, and refresh rate. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 479) and "Display Mode Optional Properties" (page 480). For general information about using dictionaries, see *CFDictionary Reference*.

Discussion

This function searches the list of available display modes for a mode that includes the specified property and comes closest to satisfying these criteria:

- Has a pixel depth equal to or greater than the specified depth
- Has dimensions equal to or greater than the specified height and width
- Uses a refresh rate equal to or near the specified rate

If no matching display mode is found, this function simply returns the current display mode.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayBitsPerPixel

Returns the number of bits used to represent a pixel in the frame buffer.

```
size_t CGDisplayBitsPerPixel (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The number of bits used to represent a pixel in the frame buffer.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayBitsPerSample

Returns the number of bits used to represent a pixel component in the frame buffer.

```
size_t CGDisplayBitsPerSample (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The number of bits used to represent a pixel component such as a color value in the frame buffer.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayBounds

Returns the bounds of a display in global display space.

```
CGRect CGDisplayBounds (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

The bounds of the display, expressed as a rectangle in the global display coordinate space (relative to the upper left corner of the main display).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayBytesPerRow

Returns the number of bytes per row in a display.

```
size_t CGDisplayBytesPerRow (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

The number of bytes per row in the display. This number also represents the stride between pixels in the same column of the display.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayCanSetPalette

Returns a Boolean value indicating whether the current display mode supports palettes.

```
boolean_t CGDisplayCanSetPalette (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the current display mode supports palettes; otherwise, `false`.

Discussion
Palettes are supported in any display selected to run in a 256-color display mode.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayCapture

Captures a display for exclusive use by an application.

```
CGDisplayErr CGDisplayCapture (
   CGDirectDisplayID display
);
```

Parameters

display
The display to capture.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
When an application captures a display, Quartz does not allow other applications and system services to use the display or change its configuration.

If hardware or software mirroring is in effect, the easiest way to capture the primary display and all mirrored displays is to use the function `CGCaptureAllDisplays` (page 400). In case of software mirroring, applications that draw directly to the display must make sure to draw the same content to all displays in the mirror set.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayCaptureWithOptions

Captures a display for exclusive use by an application, using the specified options.

```
CGDisplayErr CGDisplayCaptureWithOptions (
   CGDirectDisplayID display,
   CGCaptureOptions options
);
```

Parameters

display
The display to capture.

options
The options to use. See "Display Capture Options" (page 475).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
This function allows you to specify one or more options to use during capture of a display.

Availability
Available in Mac OS X v10.3 and later.

See Also
`CGDisplayCapture` (page 415)

Declared In
`CGDirectDisplay.h`

CGDisplayCopyColorSpace

Returns the color space for a display.

```
CGColorSpaceRef CGDisplayCopyColorSpace (
    CGDirectDisplayID display
);
```

Parameters

display
The display whose color space you want to obtain.

Return Value
The current color space for the specified display. The caller is responsible for releasing the color space with the `CGColorSpaceRelease` (page 49) function.

Discussion
This function returns a display-dependent ICC-based color space. You can use this function when rendering content for a specific display in order to produce color-matched output for that display.

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

Declared In
`CGDisplayConfiguration.h`

CGDisplayCurrentMode

Returns information about the current display mode.

```
CFDictionaryRef CGDisplayCurrentMode (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A display mode dictionary, or `NULL` if the display is invalid. The dictionary is owned by the system and you should not release it. The dictionary contains information about the current display mode. For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 479) and "Display Mode Optional Properties" (page 480). For general information about using dictionaries, see *CFDictionary Reference*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayFade

Performs a single fade operation.

```
CGError CGDisplayFade (
   CGDisplayFadeReservationToken myToken,
   CGDisplayFadeInterval seconds,
   CGDisplayBlendFraction startBlend,
   CGDisplayBlendFraction endBlend,
   float redBlend,
   float greenBlend,
   float blueBlend,
   boolean_t synchronous
);
```

Parameters

myToken
A reservation token for the fade hardware, acquired by calling `CGAcquireDisplayFadeReservation` (page 398).

seconds
The desired number of seconds for the fade operation. You should use a value in the interval `[0, kCGMaxDisplayReservationInterval]`. If the value is 0, the ending blend color is applied immediately.

startBlend
An intensity value in the interval [0, 1] that specifies the alpha component of the desired blend color at the beginning of the fade operation. See "Display Fade Blend Fractions" (page 478).

endBlend
An intensity value in the interval [0, 1] that specifies the alpha component of the desired blend color at the end of the fade operation. See "Display Fade Blend Fractions" (page 478).

redBlend
An intensity value in the interval [0, 1] that specifies the red component of the desired blend color.

greenBlend
An intensity value in the interval [0, 1] that specifies the green component of the desired blend color.

blueBlend
An intensity value in the interval [0, 1] that specifies the blue component of the desired blend color.

synchronous

Pass `true` if you want the fade operation to be synchronous; otherwise, pass `false`. If a fade operation is synchronous, the function does not return until the operation is complete.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
Over the fade operation time interval, Quartz interpolates a blending coefficient between the starting and ending values given, applying a nonlinear (sine-based) bias term. Using this coefficient, the video output is blended with the specified color.

The following example shows how to perform a two-second synchronous fade-out to black:

```
CGDisplayFade (
    myToken,
    2.0,                        // 2 seconds
    kCGDisplayBlendNormal,      // starting state
    kCGDisplayBlendSolidColor,  // ending state
    0.0, 0.0, 0.0,              // black
    true                        // wait for completion
);
```

To perform a two-second asynchronous fade-in from black:

```
CGDisplayFade (
    myToken,
    2.0,                        // 2 seconds
    kCGDisplayBlendSolidColor,  // starting state
    kCGDisplayBlendNormal,      // ending state
    0.0, 0.0, 0.0,              // black
    false                       // don't wait for completion
);
```

If you specify an asynchronous fade operation, it's safe to call `CGReleaseDisplayFadeReservation` (page 452) immediately after this function returns.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayFadeOperationInProgress

Returns a Boolean value indicating whether a fade operation is currently in progress.

```
boolean_t CGDisplayFadeOperationInProgress (
   void
);
```

Return Value
If `true`, a fade operation is currently in progress; otherwise, `false`.

Discussion
You may call this function from any task running on the system. The calling task need not have a valid fade reservation.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayGammaTableCapacity

Returns the capacity, or number of entries, in the gamma table for a display.

```
CGTableCount CGDisplayGammaTableCapacity (
   CGDirectDisplayID display
);
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayGetDrawingContext

Returns a graphics context suitable for drawing to a captured display.

```
CGContextRef CGDisplayGetDrawingContext (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value
A Quartz graphics context suitable for drawing to a captured display, or `NULL` if the display has not been captured. The context is owned by the system and you should not release it.

Discussion
After capturing a display or changing the configuration of a captured display, you can use this function to obtain the current graphics context for the display. The graphics context remains valid while the display is captured and the display configuration is unchanged. Releasing the captured display or reconfiguring the display invalidates the context. To determine when the display configuration is changing, you can use the function `CGDisplayRegisterReconfigurationCallback` (page 429) to register a display reconfiguration callback.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayHideCursor

Hides the mouse cursor, and increments the hide cursor count.

```
CGDisplayErr CGDisplayHideCursor (
   CGDirectDisplayID display
);
```

Parameters

display

This parameter is not used. By default, you may pass `kCGDirectMainDisplay`.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

This function hides the cursor regardless of its current location; the `display` parameter is ignored. In most cases, the caller must be the foreground application to affect the cursor.

Availability

Available in Mac OS X v10.0 and later.

See Also

`CGDisplayShowCursor` (page 434)

Declared In

`CGDirectDisplay.h`

CGDisplayIDToOpenGLDisplayMask

Maps a display ID to an OpenGL display mask.

```
CGOpenGLDisplayMask CGDisplayIDToOpenGLDisplayMask (
   CGDirectDisplayID display
);
```

Parameters

display

The display ID to be converted.

Return Value

The OpenGL display mask that corresponds to the specified display.

Discussion

OpenGL sometimes identifies a display using a bitmask with one bit set. This function maps a display ID to the corresponding OpenGL display mask.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayIOServicePort

Returns the I/O Kit service port of the specified display.

```
io_service_t CGDisplayIOServicePort (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

The I/O Kit service port for the specified display.

Discussion

An I/O Kit service port can be passed to I/O Kit to obtain additional information about the display.

The port is owned by the graphics system, and should not be destroyed.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayIsActive

Returns a Boolean value indicating whether a display is active.

```
boolean_t CGDisplayIsActive (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the specified display is active; otherwise, `false`.

Discussion

An active display is connected, awake, and available for drawing. In a hardware mirroring set, only the primary display is active.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayIsAlwaysInMirrorSet

Returns a Boolean value indicating whether a display is always in a mirroring set.

```
boolean_t CGDisplayIsAlwaysInMirrorSet (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the specified display is in a mirroring set and cannot be removed from this set.

Discussion

Some hardware configurations support the connection of auxiliary displays that always mirror the main display, and therefore cannot be removed from the mirroring set to which they belong.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayIsAsleep

Returns a Boolean value indicating whether a display is sleeping (and is therefore not drawable.)

```
boolean_t CGDisplayIsAsleep (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the specified display is in sleep mode; otherwise, `false`.

Discussion

A display is sleeping when its frame buffer and the attached monitor are in reduced power mode. A sleeping display is still considered to be a part of global display (desktop) space, but it is not drawable.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayIsBuiltin

Returns a Boolean value indicating whether a display is built-in, such as the internal display in portable systems.

```
boolean_t CGDisplayIsBuiltin (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the specified display is considered to be a built-in display; otherwise, `false`.

Discussion

Portable systems typically identify the internal LCD panel as a built-in display.

Note that it is possible and reasonable for a system to have no displays marked as built-in. For example, a portable system running with the lid closed may report no built-in displays.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayIsCaptured

Returns a Boolean value indicating whether a display is captured.

```
boolean_t CGDisplayIsCaptured (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, the specified display is captured; otherwise, `false`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayIsInHWMirrorSet

Returns a Boolean value indicating whether a display is in a hardware mirroring set.

```
boolean_t CGDisplayIsInHWMirrorSet (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value
If `true`, the specified display is a member of a hardware mirroring set; otherwise, `false`.

Discussion
When hardware mirroring is enabled, the contents of a single frame buffer are rendered in all displays in the hardware mirroring set. All drawing operations are directed to the primary display in the set—see `CGDisplayPrimaryDisplay` (page 428).

For more information about display mirroring, see `CGConfigureDisplayMirrorOfDisplay` (page 403).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsInMirrorSet

Returns a Boolean value indicating whether a display is in a mirroring set.

```
boolean_t CGDisplayIsInMirrorSet (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
If `true`, the specified display is a member of a software or hardware mirroring set; otherwise, `false`.

Discussion
For more information about display mirroring, see `CGConfigureDisplayMirrorOfDisplay` (page 403).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsMain

Returns a Boolean value indicating whether a display is the main display.

```
boolean_t CGDisplayIsMain (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
If `true`, the specified display is currently the main display; otherwise, `false`.

Discussion
For information about the characteristics of a main display, see `CGMainDisplayID` (page 444).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsOnline

Returns a Boolean value indicating whether a display is connected or online.

```
boolean_t CGDisplayIsOnline (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
If `true`, the specified display is connected; otherwise, `false`.

Discussion
A display is considered connected or online when the frame buffer hardware is connected to a monitor.

You can use this function to determine if someone has hot-plugged a display to the system. Note that hot-plugging is a hardware feature that may not be present on all displays.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayIsStereo

Returns a Boolean value indicating whether a display is running in a stereo graphics mode.

```
boolean_t CGDisplayIsStereo (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
If `true`, the specified display is running in a stereo graphics mode; otherwise, `false`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayMirrorsDisplay

For a secondary display in a mirroring set, returns the primary display.

```
CGDirectDisplayID CGDisplayMirrorsDisplay (
   CGDirectDisplayID display
);
```

Parameters

display
A secondary display in a mirroring set.

Return Value
Returns the primary display in the mirroring set. Returns `kCGNullDirectDisplay` if the specified display is actually the primary display or is not in a mirroring set.

Discussion
For more information about display mirroring, see `CGConfigureDisplayMirrorOfDisplay` (page 403).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayModelNumber

Returns the model number of a display monitor.

```
uint32_t CGDisplayModelNumber (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A model number for the monitor associated with the specified display, or a constant to indicate an exception—see the discussion below.

Discussion
This function uses I/O Kit to identify the monitor associated with the specified display. The return value depends on the following:

- If I/O Kit can identify the monitor, the product ID code for the monitor is returned.
- If I/O Kit can't identify the monitor, `kDisplayProductIDGeneric` is returned.
- If no monitor is connected, a value of `0xFFFFFFFF` is returned.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayMoveCursorToPoint

Moves the mouse cursor to a specified point relative to the display origin (the upper left corner of the display).

```
CGDisplayErr CGDisplayMoveCursorToPoint (
   CGDirectDisplayID display,
   CGPoint point
);
```

Parameters

display
The display to access.

point
The coordinates of a point in local display space. The origin is the upper left corner of the specified display.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
No events are generated as a result of this move. Points that would lie outside the desktop are clipped to the desktop.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayPixelsHigh

Returns the display height in pixel units.

```
size_t CGDisplayPixelsHigh (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The display height in pixel units.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayPixelsWide

Returns the display width in pixel units.

```
size_t CGDisplayPixelsWide (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The display width in pixel units.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayPrimaryDisplay

Returns the primary display in a hardware mirroring set.

```
CGDirectDisplayID CGDisplayPrimaryDisplay (
   CGDirectDisplayID display
);
```

Parameters

display
A display in a hardware mirror set.

Return Value
The primary display in the mirror set. If `display` is not hardware-mirrored, this function simply returns `display`.

Discussion
In hardware mirroring, the contents of a single frame buffer are rendered in two or more displays simultaneously. The mirrored displays are said to be in a *hardware mirroring set*.

At the discretion of the device driver, one of the displays in a hardware mirroring set is designated as the *primary* display. The device driver binds the drawing engine, hardware accelerator, and 3D engine to the primary display, and directs all drawing operations to this display.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayRegisterReconfigurationCallback

Registers a callback function to be invoked whenever a local display is reconfigured.

```
CGError CGDisplayRegisterReconfigurationCallback (
   CGDisplayReconfigurationCallBack proc,
   void *userInfo
);
```

Parameters

proc

A pointer to the callback function to be registered.

userInfo

A pointer to user-defined data, or `NULL`. The `userInfo` argument is passed back to the callback function each time it's invoked.

Discussion

Whenever local displays are reconfigured, the callback function you register is invoked twice for each display that's added, removed, or currently online—once before the reconfiguration, and once after the reconfiguration. For more information, see the callback type `CGDisplayReconfigurationCallBack` (page 462).

A callback function may be registered multiple times with different user-defined data pointers, resulting in multiple registration entries. For each registration, when notification is no longer needed you should remove the registration by calling the function `CGDisplayRemoveReconfigurationCallback` (page 430).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayRelease

Releases a captured display.

```
CGDisplayErr CGDisplayRelease (
   CGDirectDisplayID display
);
```

Parameters

display

The display to release.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayRemoveReconfigurationCallback

Removes the registration of a callback function that's invoked whenever a local display is reconfigured.

```
CGError CGDisplayRemoveReconfigurationCallback (
   CGDisplayReconfigurationCallBack proc,
   void *userInfo
);
```

Parameters

proc

A pointer to the callback function associated with the registration to be removed.

userInfo

A pointer to user-defined data associated with the registration to be removed, or `NULL`. This is the same pointer that's passed to the function `CGDisplayRegisterReconfigurationCallback` (page 429) when registering the callback.

Discussion

When you call this function, the two arguments must match the registered entry to be removed.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayRestoreColorSyncSettings

Restores the gamma tables to the values in the user's ColorSync display profile.

```
void CGDisplayRestoreColorSyncSettings (
   void
);
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGDisplayRotation

Returns the rotation angle of a display in degrees.

```
double CGDisplayRotation (
    CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

The rotation angle of the display in degrees, or 0 if the display is not valid.

Discussion
This function returns the rotation angle of a display in a clockwise direction. For example, if the specified display is rotated clockwise 90 degrees then this function returns 90.0. After a 90 degree clockwise rotation, the physical bottom of the display is on the left side and the physical top is on the right side.

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

Declared In
`CGDisplayConfiguration.h`

CGDisplaySamplesPerPixel

Returns the number of color components used to represent a pixel.

```
size_t CGDisplaySamplesPerPixel (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The number of color components used to represent a pixel.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayScreenSize

Returns the width and height of a display in millimeters.

```
CGSize CGDisplayScreenSize (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
The size of the specified display in millimeters, or 0 if the display is not valid.

Discussion
If Extended Display Identification Data (EDID) for the display device is not available, the size is estimated based on the device width and height in pixels from `CGDisplayBounds` (page 413), with an assumed resolution of 2.835 pixels/mm or 72 DPI, a reasonable guess for displays predating EDID support.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplaySerialNumber

Returns the serial number of a display monitor.

```
uint32_t CGDisplaySerialNumber (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A serial number for the monitor associated with the specified display, or a constant to indicate an exception—see the discussion below.

Discussion
This function uses I/O Kit to identify the monitor associated with the specified display.

If I/O Kit can identify the monitor:

- If the manufacturer has encoded a serial number for the monitor, the number is returned.
- If there is no encoded serial number, `0x00000000` is returned.

If I/O Kit cannot identify the monitor:

- If a monitor is connected to the display, `0x00000000` is returned.
- If no monitor is connected to the display hardware, a value of `0xFFFFFFFF` is returned.

Note that a serial number is meaningful only in conjunction with a specific vendor and product or model.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplaySetPalette

Sets the palette for a display.

```
CGDisplayErr CGDisplaySetPalette (
   CGDirectDisplayID display,
   const CGDirectPaletteRef palette
);
```

Parameters

display
The display to access.

palette
The display palette to set.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplaySetStereoOperation

Immediately enables or disables stereo operation for a display.

```
CGError CGDisplaySetStereoOperation (
   CGDirectDisplayID display,
   boolean_t stereo,
   boolean_t forceBlueLine,
   CGConfigureOption option
);
```

Parameters

display
The display being configured.

stereo
Pass `true` if you want to enable stereo operation. To disable it, pass `false`.

forceBlueLine
When in stereo operation, a display may need to generate a special stereo sync signal as part of the video output. The sync signal consists of a blue line which occupies the first 25% of the last scanline for the left eye view, and the first 75% of the last scanline for the right eye view. The remainder of the scanline is black. To force the display to generate this sync signal, pass `true`; otherwise pass `false`.

option
A constant that specifies the scope of the display configuration changes. For more information, see "Display Configuration Scopes" (page 477).

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
The system normally detects the presence of a stereo window and automatically switches a display containing a stereo window to stereo operation. This function provides a mechanism to force a display to stereo operation immediately, and to set options (blue line sync signal) when in stereo operation.

On success, the display resolution, mirroring mode, and available display modes may change due to hardware-specific capabilities and limitations. You should check these settings to verify that they are appropriate for your application.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGDisplayConfiguration.h

CGDisplayShowCursor

Decrements the hide cursor count, and shows the mouse cursor if the count is zero.

```
CGDisplayErr CGDisplayShowCursor (
   CGDirectDisplayID display
);
```

Parameters

display
This parameter is not used. By default, you may pass kCGDirectMainDisplay.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
If the hide cursor count is zero, this function shows the cursor regardless of its current location; the display parameter is ignored. In most cases, the caller must be the foreground application to affect the cursor.

Availability
Available in Mac OS X v10.0 and later.

See Also
CGDisplayHideCursor (page 419)

Declared In
CGDirectDisplay.h

CGDisplaySwitchToMode

Switches a display to a different mode.

```
CGDisplayErr CGDisplaySwitchToMode (
   CGDirectDisplayID display,
   CFDictionaryRef mode
);
```

Parameters

display
The display to access.

mode
A display mode dictionary that contains information about the display mode to set. The dictionary passed in must be a dictionary returned by another Quartz display function such as CGDisplayAvailableModes (page 408) or CGDisplayBestModeForParameters (page 410). For a list of the properties in a display mode dictionary, see "Display Mode Standard Properties" (page 479) and "Display Mode Optional Properties" (page 480). For general information about using dictionaries, see *CFDictionary Reference*.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
This function switches the display mode of the specified display. The operation is always synchronous; the function does not return until the mode switch is complete. Note that after switching, display parameters and addresses may change.

The selected display mode persists for the life of the calling program. When the program terminates, the display mode automatically reverts to the permanent setting in the Displays panel of System Preferences.

When changing the display mode of a display in a mirroring set, other displays in the mirroring set will be assigned a mode that's capable of mirroring the bounds of the display being adjusted. To avoid this automatic behavior, you can use the following procedure: call `CGBeginDisplayConfiguration`, call `CGConfigureDisplayMode` for each display to explicitly set the mode, and finally call `CGCompleteDisplayConfiguration`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDisplayUnitNumber

Returns the logical unit number of a display.

```
uint32_t CGDisplayUnitNumber (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A logical unit number for the specified display.

Discussion
The logical unit number represents a particular node in the I/O Kit device tree associated with the display's frame buffer. For a particular hardware configuration, this value will not change when the attached monitor is changed.

The unit number will change if the I/O Kit device tree changes, as when hardware is reconfigured, drivers are replaced, or significant changes occur to I/O Kit, so it should not be assumed to be invariant across login sessions.

For more information about I/O Kit, see the Apple publication *"I/O Kit Fundamentals"*.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayUsesOpenGLAcceleration

Returns a Boolean value indicating whether Quartz is using OpenGL-based window acceleration (Quartz Extreme) to render in a display.

```
boolean_t CGDisplayUsesOpenGLAcceleration (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

If `true`, Quartz Extreme is used to render in the specified display; otherwise, `false`.

Discussion

Quartz Extreme is an OpenGL-based, hardware-accelerated window compositor available in Mac OS X version 10.2 and later. Quartz Extreme requires a minimum hardware configuration to operate.

The information this function provides is typically used to adjust the demands of drawing operations to the capabilities of the display hardware. For example, an application running on an unaccelerated system could disable live window-resizing.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CGDisplayConfiguration.h`

CGDisplayVendorNumber

Returns the vendor number of the specified display's monitor.

```
uint32_t CGDisplayVendorNumber (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

A vendor number for the monitor associated with the specified display, or a constant to indicate an exception—see the discussion below.

Discussion

This function uses I/O Kit to identify the monitor associated with the specified display.

There are three cases:

- If I/O Kit can identify the monitor, the vendor ID is returned.
- If I/O Kit cannot identify the monitor, `kDisplayVendorIDUnknown` is returned.
- If there is no monitor associated with the display, `0xFFFFFFFF` is returned.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGDisplayWaitForBeamPositionOutsideLines

Waits until the beam position moves outside a region in a display screen. This function is not designed for VBL drawing synchronization.

```
CGDisplayErr CGDisplayWaitForBeamPositionOutsideLines (
   CGDirectDisplayID display,
   CGBeamPosition upperScanLine,
   CGBeamPosition lowerScanLine
);
```

Parameters

display
The display to access.

upperScanLine
The upper scan line number.

lowerScanLine
The lower scan line number.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
This function waits until the beam position is outside the range specified by the arguments `upperScanLine` and `lowerScanLine`. If the value of `upperScanLine` is greater than the value of `lowerScanLine`, or if `upperScanLine` and `lowerScanLine` encompass the entire display height, this function returns an error.

Some displays may not use conventional video vertical and horizontal sweep in painting. These displays report a `kCGDisplayRefreshRate` of 0 in the dictionary returned by `CGDisplayCurrentMode` (page 416). Also, some display device drivers may not implement support for this mechanism. On such displays, this function returns at once.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGGetActiveDisplayList

Provides a list of displays that are active (or drawable).

```
CGDisplayErr CGGetActiveDisplayList (
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *activeDspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

maxDisplays

The size of the `activeDspys` array. This value determines the maximum number of displays that can be returned.

activeDspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of active displays. If you pass `NULL`, on return the display count contains the total number of active displays.

dspyCnt

A pointer to a display count variable provided by the caller. On return, the display count contains the actual number of displays returned in the `activeDspys` array. This value is at most `maxDisplays`.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
The first entry in the list of active displays is the main display. In case of mirroring, the first entry is the largest drawable display or, if all are the same size, the display with the greatest pixel depth.

Note that when hardware mirroring is being used between displays, only the primary display is active and appears in the list. When software mirroring is being used, all the mirrored displays are active and appear in the list. For more information about mirroring, see `CGConfigureDisplayMirrorOfDisplay` (page 403).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTCarbonShell

Declared In
`CGDirectDisplay.h`

CGGetDisplaysWithOpenGLDisplayMask

Provides a list of displays that corresponds to the bits set in an OpenGL display mask.

```
CGDisplayErr CGGetDisplaysWithOpenGLDisplayMask (
   CGOpenGLDisplayMask mask,
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *dspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

mask

An OpenGL display mask that identifies one or more displays.

maxDisplays

The size of the `dspys` array. This value determines the maximum number of displays that can be returned.

dspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of displays that corresponds to the bits set in the mask. If you pass `NULL`, on return the display count contains the total number of displays specified in the mask.

dspyCnt

A pointer to a display count variable provided by the caller. On return, the display count contains the actual number of displays returned in the `dspys` array. This value is at most `maxDisplays`.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGGetDisplaysWithPoint

Provides a list of online displays with bounds that include the specified point.

```
CGDisplayErr CGGetDisplaysWithPoint (
   CGPoint point,
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *dspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

point

The coordinates of a point in global display space. The origin is the upper left corner of the main display.

maxDisplays

The size of the `dspys` array. This value determines the maximum number of displays that can be returned.

dspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of displays with bounds that include the point. If you pass `NULL`, on return the display count contains the total number of displays with bounds that include the point.

dspyCnt

A pointer to a display count variable provided by the caller. On return, the display count contains the actual number of displays returned in the `dspys` array. This value is at most `maxDisplays`.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGGetDisplaysWithRect

Gets a list of online displays with bounds that intersect the specified rectangle.

```
CGDisplayErr CGGetDisplaysWithRect (
   CGRect rect,
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *dspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

rect

The location and size of a rectangle in global display space. The origin is the upper left corner of the main display.

maxDisplays

The size of the *dspys* array. This value determines the maximum number of displays that can be returned in the *dspys* parameter. Generally, you should specify a number greater than 0 for this parameter. If you specify 0, the value returned in *dspyCnt* is undefined and this function sets the *dspys* parameter to NULL.

dspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of displays whose bounds intersect the specified rectangle.

dspyCnt

A pointer to a display count variable provided by the caller. On return, this variable contains the number of displays that were returned in the *dspys* parameter. You must provide a non-NULL value for this parameter.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGGetDisplayTransferByFormula

Gets the coefficients of the gamma transfer formula for a display.

```
CGDisplayErr CGGetDisplayTransferByFormula (
   CGDirectDisplayID display,
   CGGammaValue *redMin,
   CGGammaValue *redMax,
   CGGammaValue *redGamma,
   CGGammaValue *greenMin,
   CGGammaValue *greenMax,
   CGGammaValue *greenGamma,
   CGGammaValue *blueMin,
   CGGammaValue *blueMax,
   CGGammaValue *blueGamma
);
```

Parameters

display

The display to access.

redMin

The minimum value of the red channel in the gamma table. The value is a number in the interval [0, redMax).

redMax

The maximum value of the red channel in the gamma table. The value is a number in the interval (redMin, 1].

redGamma

A positive value used to compute the red channel in the gamma table.

greenMin

The minimum value of the green channel in the gamma table. The value is a number in the interval [0, greenMax).

greenMax

The maximum value of the green channel in the gamma table. The value is a number in the interval (greenMin, 1].

greenGamma

A positive value used to compute the green channel in the gamma table.

blueMin

The minimum value of the blue channel in the gamma table. The value is a number in the interval [0, blueMax).

blueMax

The maximum value of the blue channel in the gamma table. The value is a number in the interval (blueMin, 1].

blueGamma

A positive value used to compute the blue channel in the gamma table.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

For information about the gamma transfer formula, see the description of the function `CGSetDisplayTransferByFormula` (page 455).

Availability

Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGGetDisplayTransferByTable

Gets the values in the RGB gamma tables for a display.

```
CGDisplayErr CGGetDisplayTransferByTable (
   CGDirectDisplayID display,
   CGTableCount capacity,
   CGGammaValue *redTable,
   CGGammaValue *greenTable,
   CGGammaValue *blueTable,
   CGTableCount *sampleCount
);
```

Parameters

display
The display to access.

capacity
The number of entries each table can hold.

redTable
A pointer to an array of type `CGGammaValue` with size `capacity`. On return, the array contains the values of the red channel in the display's gamma table.

greenTable
A pointer to an array of type `CGGammaValue` with size `capacity`. On return, the array contains the values of the green channel in the display's gamma table.

blueTable
A pointer to an array of type `CGGammaValue` with size `capacity`. On return, the array contains the values of the blue channel in the display's gamma table.

sampleCount
The number of samples actually copied into each array.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGGetLastMouseDelta

Reports the change in mouse position since the last mouse movement event received by the application.

```
void CGGetLastMouseDelta (
   CGMouseDelta *deltaX,
   CGMouseDelta *deltaY
);
```

Parameters

deltaX

A pointer to a `CGMouseDelta` variable. On return, this variable contains the horizontal change in the mouse position since the last mouse movement event.

deltaY

A pointer to a `CGMouseDelta` variable. On return, this variable contains the vertical change in the mouse position since the last mouse movement event.

Discussion

This function is not recommended for general use. Instead, you should use the mouse tracking functions in the Carbon Event Manager.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGGetOnlineDisplayList

Provides a list of displays that are online (active, mirrored, or sleeping).

```
CGDisplayErr CGGetOnlineDisplayList (
   CGDisplayCount maxDisplays,
   CGDirectDisplayID *onlineDspys,
   CGDisplayCount *dspyCnt
);
```

Parameters

maxDisplays

The size of the `onlineDspys` array. This value determines the maximum number of display IDs that can be returned.

onlineDspys

A pointer to storage provided by the caller for an array of display IDs. On return, the array contains a list of the online displays. If you pass `NULL`, on return the display count contains the total number of online displays.

dspyCnt

A pointer to a display count variable provided by the caller. On return, the display count contains the actual number of displays returned in the `onlineDspys` array. This value is at most `maxDisplays`.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

If the frame buffer hardware is connected, a display is considered connected or online.

When hardware mirroring is used, a display can be online but not active or drawable. Programs which manipulate display settings such as the palette or gamma tables need access to all displays, including hardware mirrors which are not drawable.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDirectDisplay.h`

CGMainDisplayID

Returns the display ID of the main display.

```
CGDirectDisplayID CGMainDisplayID (
   void
);
```

Return Value
The display ID assigned to the main display.

Discussion
The main display is the display with its screen location at (0,0) in global coordinates. In a system without display mirroring, the display with the menu bar is typically the main display.

If mirroring is enabled and the menu bar appears on more than one display, this function provides a reliable way to find the main display.

In case of hardware mirroring, the drawable display becomes the main display. In case of software mirroring, the display with the highest resolution and deepest pixel depth typically becomes the main display.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
LiveVideoMixer2

Declared In
`CGDirectDisplay.h`

CGOpenGLDisplayMaskToDisplayID

Maps an OpenGL display mask to a display ID.

```
CGDirectDisplayID CGOpenGLDisplayMaskToDisplayID (
   CGOpenGLDisplayMask mask
);
```

Parameters

mask
The OpenGL display mask to be converted.

Return Value
The display ID assigned to the specified display mask, or `kCGNullDirectDisplay` if no display matches the mask.

Discussion
OpenGL sometimes identifies a display using a bitmask with one bit set. This function maps such a display mask to the corresponding display ID. If you pass in a mask with multiple bits set, this function returns a display ID matching one of these bits.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDirectDisplay.h`

CGPaletteCreateCopy

Returns a copy of a specified display palette.

```
CGDirectPaletteRef CGPaletteCreateCopy (
   CGDirectPaletteRef palette
);
```

Parameters

palette
The display palette to copy.

Return Value
A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 450).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGPaletteCreateDefaultColorPalette

Returns a new display palette representing the default 8-bit color palette.

```
CGDirectPaletteRef CGPaletteCreateDefaultColorPalette (
   void
);
```

Return Value
A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 450).

Discussion
Palettes are used with 256 color display modes. The default palette is the old default 8-bit Mac OS palette, with white at index 0 and black at index 255.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGPaletteCreateFromPaletteBlendedWithColor

Returns a new tinted display palette. The new palette is derived from an existing palette blended with a solid color, at a specified level of intensity.

```
CGDirectPaletteRef CGPaletteCreateFromPaletteBlendedWithColor (
   CGDirectPaletteRef palette,
   CGPaletteBlendFraction fraction,
   CGDeviceColor color
);
```

Parameters

palette

The palette to blend.

fraction

A value between 0 and 1 that represents the blend intensity. See `CGPaletteBlendFraction` (page 473).

color

The blend color. See `CGDeviceColor` (page 467).

Return Value

A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 450).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteCreateWithByteSamples

Returns a new display palette using 8-bit sample data.

```
CGDirectPaletteRef CGPaletteCreateWithByteSamples (
   CGDeviceByteColor *sampleTable,
   CGTableCount sampleCount
);
```

Parameters

sampleTable

A color table with integer values that represent the intensity of the red,green, and blue components in each table entry. Each value ranges from 0 (no color) to 255 (full intensity). See `CGDeviceByteColor` (page 466).

sampleCount

The number of entries in the specified color table.

Return Value

A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 450).

Availability

Available in Mac OS X v10.0 and later.

Declared In
CGDirectPalette.h

CGPaletteCreateWithCapacity

Returns a new display palette with a specified capacity. The new palette is initialized from the default color palette.

```
CGDirectPaletteRef CGPaletteCreateWithCapacity (
   CGTableCount capacity
);
```

Parameters

capacity
The number of entries in the new palette.

Return Value
A new display palette object. When you no longer need the palette, you should release it using the function CGPaletteRelease (page 450).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectPalette.h

CGPaletteCreateWithDisplay

Returns a copy of the current palette for a display.

```
CGDirectPaletteRef CGPaletteCreateWithDisplay (
   CGDirectDisplayID display
);
```

Parameters

display
The display to access.

Return Value
A new display palette object, or NULL if the current display mode does not support a palette. When you no longer need the palette, you should release it using the function CGPaletteRelease (page 450).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectPalette.h

CGPaletteCreateWithSamples

Returns a new display palette using RGB sample data.

```
CGDirectPaletteRef CGPaletteCreateWithSamples (
   CGDeviceColor *sampleTable,
   CGTableCount sampleCount
);
```

Parameters

sampleTable

A color table with floating point values that represent the intensity of the red,green, and blue components in each table entry. Each value ranges from 0 (no color) to 1 (full intensity). See `CGDeviceColor` (page 467).

sampleCount

The number of entries in the specified color table.

Return Value

A new display palette object. When you no longer need the palette, you should release it using the function `CGPaletteRelease` (page 450).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteGetColorAtIndex

Returns the color value at the specified index.

```
CGDeviceColor CGPaletteGetColorAtIndex (
   CGDirectPaletteRef palette,
   CGTableCount index
);
```

Parameters

palette

The display palette to access.

index

The zero-based index of the desired palette entry.

Return Value

A color value. See `CGDeviceColor` (page 467).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteGetIndexForColor

Returns the index of the display palette entry that most closely matches a specified color value.

```
CGTableCount CGPaletteGetIndexForColor (
   CGDirectPaletteRef palette,
   CGDeviceColor color
);
```

Parameters

palette

The display palette to access.

color

The color value to match. See `CGDeviceColor` (page 467).

Return Value

The index of the display palette entry that most closely matches the specified color value.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteGetNumberOfSamples

Returns the number of colors in a display palette.

```
CGTableCount CGPaletteGetNumberOfSamples (
   CGDirectPaletteRef palette
);
```

Parameters

palette

The display palette to access.

Return Value

The number of colors in the specified display palette.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectPalette.h`

CGPaletteIsEqualToPalette

Returns a Boolean value indicating whether two display palettes are equal.

```
Boolean CGPaletteIsEqualToPalette (
   CGDirectPaletteRef palette1,
   CGDirectPaletteRef palette2
);
```

Parameters

palette1

The first display palette to compare.

palette2
The second display palette to compare.

Return Value
If `true`, the two specified display palettes are equal; otherwise, `false`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGPaletteRelease

Decrements the retain count of a display palette.

```
void CGPaletteRelease (
   CGDirectPaletteRef palette
);
```

Parameters

palette
The display palette to release.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGPaletteSetColorAtIndex

Updates the color value at the specified index in a display palette.

```
void CGPaletteSetColorAtIndex (
   CGDirectPaletteRef palette,
   CGDeviceColor color,
   CGTableCount index
);
```

Parameters

palette
The display palette to access.

color
The new color value.

index
The index of the palette entry to update.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGRegisterScreenRefreshCallback

Registers a callback function to be invoked when local displays are refreshed or modified.

```
CGError CGRegisterScreenRefreshCallback (
   CGScreenRefreshCallback function,
   void *userParameter
);
```

Parameters

function

A pointer to the callback function to be registered.

userParameter

A pointer to user-defined data, or `NULL`. The `userParameter` argument is passed back to the callback function each time it's invoked.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

A callback function may be registered multiple times with different user-defined data pointers, resulting in multiple registration entries. For each registration, when notification is no longer needed you should call the function `CGUnregisterScreenRefreshCallback` (page 458) to remove the registration.

The callback function you register is invoked only if your application has an active event loop. The callback is invoked in the same thread of execution that is processing events within your application.

Special Considerations

In Mac OS X v10.4 and earlier, the result code returned by this function is a random value and should be ignored. In Mac OS X v10.5 and later, the result code is valid.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGRemoteOperation.h`

CGReleaseAllDisplays

Releases all captured displays.

```
CGDisplayErr CGReleaseAllDisplays (
   void
);
```

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

This function releases all captured displays and restores the display modes to the user's preferences. It may be used in conjunction with any of the functions that capture displays, such as `CGCaptureAllDisplays` (page 400).

Availability

Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGReleaseDisplayFadeReservation

Releases a display fade reservation, and unfades the display if needed.

```
CGError CGReleaseDisplayFadeReservation (
   CGDisplayFadeReservationToken myToken
);
```

Parameters

myToken

The current fade reservation token to be released. On return, the reservation token is no longer valid and should be discarded.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
If you call this function while an asynchronous fade operation is running, there are two possible outcomes:

- If the ending blend value is `kCGDisplayBlendNormal`, the fade operation is allowed to run to completion.
- If the ending blend value is not `kCGDisplayBlendNormal`, the fade operation is terminated immediately and the display is returned to normal.

In both cases, the reservation is actually released when the fade operation completes.

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGDisplayFade.h

CGReleaseScreenRefreshRects

Deallocates a list of rectangles that represent changed areas on local displays.

```
void CGReleaseScreenRefreshRects (
   CGRect *rectArray
);
```

Parameters

rectArray

A list of rectangles obtained by calling `CGWaitForScreenRefreshRects` (page 459) or `CGWaitForScreenUpdateRects` (page 460).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGRemoteOperation.h

CGRestorePermanentDisplayConfiguration

Restores the permanent display configuration settings for the current user.

```
void CGRestorePermanentDisplayConfiguration (
   void
);
```

Discussion
This function provides a convenient way to restore the permanent display configuration.

Applications that temporarily change the display configuration—such as applications and games that switch to full-screen display mode—can use this function to undo the changes.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayConfiguration.h`

CGScreenRegisterMoveCallback

Registers a callback function to be invoked when an area of the display is moved.

```
CGError CGScreenRegisterMoveCallback (
   CGScreenUpdateMoveCallback function,
   void *userParameter
);
```

Parameters

function
A pointer to the callback function to be registered.

userParameter
A pointer to user-defined data, or `NULL`. The `userParameter` argument is passed back to the callback function each time it's invoked.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
A callback function may be registered multiple times with different user-defined data pointers, resulting in multiple registration entries. For each registration, when notification is no longer needed you should remove the registration by calling the function `CGScreenUnregisterMoveCallback` (page 454).

The callback function you register is invoked only if your application has an active event loop. The callback is invoked in the same thread of execution that is processing events within your application.

Special Considerations

This function is implemented in Mac OS X version 10.4.3 and later.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGRemoteOperation.h`

CGScreenUnregisterMoveCallback

Removes a previously registered callback function invoked when an area of the display is moved.

```
void CGScreenUnregisterMoveCallback (
   CGScreenUpdateMoveCallback function,
   void *userParameter
);
```

Parameters

function

A pointer to the callback function to be unregistered.

userParameter

A pointer to user-defined data, or `NULL`. You should pass the same value you used when you registered the callback function.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

When you call this function, the two arguments must match the registered entry to be removed.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGRemoteOperation.h`

CGSessionCopyCurrentDictionary

Returns information about the caller's window server session.

```
CFDictionaryRef CGSessionCopyCurrentDictionary (
   void
);
```

Return Value

A window server session dictionary, or `NULL` if the caller is not running within a Quartz GUI session or the window server is disabled. You should release the dictionary when you are finished using it. For information about the key-value pairs in this dictionary, see "Window Server Session Properties" (page 485).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CGSession.h`

CGSetDisplayTransferByByteTable

Sets the byte values in the 8-bit RGB gamma tables for a display.

```
CGDisplayErr CGSetDisplayTransferByByteTable (
   CGDirectDisplayID display,
   CGTableCount tableSize,
   const CGByteValue *redTable,
   const CGByteValue *greenTable,
   const CGByteValue *blueTable
);
```

Parameters

display

The display to access.

tableSize

The number of entries in each table.

redTable

An array of size `tableSize` containing the byte values of the red channel in the display's gamma table.

greenTable

An array of size `tableSize` containing the byte values of the green channel in the display's gamma table.

blueTable

An array of size `tableSize` containing the byte values of the blue channel in the display's gamma table.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

The same table may be passed in for the red, green, and blue channels. The tables are interpolated as needed to generate the number of samples required by the graphics hardware.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGSetDisplayTransferByFormula

Sets the gamma function for a display, by specifying the coefficients of the gamma transfer formula.

```
CGDisplayErr CGSetDisplayTransferByFormula (
   CGDirectDisplayID display,
   CGGammaValue redMin,
   CGGammaValue redMax,
   CGGammaValue redGamma,
   CGGammaValue greenMin,
   CGGammaValue greenMax,
   CGGammaValue greenGamma,
   CGGammaValue blueMin,
   CGGammaValue blueMax,
   CGGammaValue blueGamma
);
```

Parameters

display

The display to access.

redMin

The minimum value of the red channel in the gamma table. The value should be a number in the interval [0, redMax).

redMax

The maximum value of the red channel in the gamma table. The value should be a number in the interval (redMin, 1].

redGamma

A positive value used to compute the red channel in the gamma table.

greenMin

The minimum value of the green channel in the gamma table. The value should be a number in the interval [0, greenMax).

greenMax

The maximum value of the green channel in the gamma table. The value should be a number in the interval (greenMin, 1].

greenGamma

A positive value used to compute the green channel in the gamma table.

blueMin

The minimum value of the blue channel in the gamma table. The value should be a number in the interval [0, blueMax).

blueMax

The maximum value of the blue channel in the gamma table. The value should be a number in the interval (blueMin, 1].

blueGamma

A positive value used to compute the blue channel in the gamma table.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

This function uses the specified parameter values to compute a gamma correction table for the specified display. The values in the table are computed by sampling the following gamma transfer formula for a range of indices from 0 to 1:

```
value = Min + ((Max - Min) * pow(index, Gamma))
```

The resulting values are converted to a machine-specific format and loaded into display hardware.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGSetDisplayTransferByTable

Sets the color gamma function for a display, by specifying the values in the RGB gamma tables.

```
CGDisplayErr CGSetDisplayTransferByTable (
   CGDirectDisplayID display,
   CGTableCount tableSize,
   const CGGammaValue *redTable,
   const CGGammaValue *greenTable,
   const CGGammaValue *blueTable
);
```

Parameters

display
The display to access.

tableSize
The number of entries in each table.

redTable
An array of size `tableSize` containing the values of the red channel in the display's gamma table. The values should be in the range 0.0 to 1.0.

greenTable
An array of size `tableSize` containing the values of the green channel in the display's gamma table. The values should be in the range 0.0 to 1.0.

blueTable
An array of size `tableSize` containing the values of the blue channel in the display's gamma table. The values should be in the range 0.0 to 1.0.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
The same table may be passed in for the red, green, and blue channels. The tables are interpolated as needed to generate the number of samples required by the graphics hardware.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGShieldingWindowID

Returns the window ID of the shield window for a captured display.

```
uint32_t CGShieldingWindowID (
   CGDirectDisplayID display
);
```

Parameters

display

The display to access.

Return Value

The window ID of the shield window for the specified display, or `NULL` if the display is not shielded.

Discussion

To prevent updates by direct-to-screen programs (such as Classic), Quartz draws a shield window that fills the entire screen of a captured display.

This function is not recommended for use in applications. Note that the graphics context associated with this window is not a full-featured drawing context. To get a full-featured drawing context for a captured display, you should use the function `CGDisplayGetDrawingContext` (page 419).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGShieldingWindowLevel

Returns the window level of the shield window for a captured display.

```
int32_t CGShieldingWindowLevel (
   void
);
```

Return Value

The window level of the shield window for a captured display.

Discussion

This function returns a value that is sometimes used to position a window over the shield window for a captured display. Attempting to position a window over a captured display may be unsuccessful—or may present undesirable results such as illegible or invisible content—because of interactions between full-screen graphics (such as OpenGL full-screen drawing contexts) and the graphics hardware. Because of these limitations, and because the implementation of display capture may change in the future, this technique is not recommended.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGDirectDisplay.h`

CGUnregisterScreenRefreshCallback

Removes a previously registered callback function invoked when local displays are refreshed or modified.

```
void CGUnregisterScreenRefreshCallback (
   CGScreenRefreshCallback function,
   void *userParameter
);
```

Parameters

function

A pointer to the callback function to be unregistered.

userParameter

A pointer to user-defined data, or `NULL`. You should pass the same value you used when you registered the callback function.

Discussion

When you call this function, the two arguments must match the registered entry to be removed.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGRemoteOperation.h`

CGWaitForScreenRefreshRects

Waits for screen refresh operations.

```
CGError CGWaitForScreenRefreshRects (
   CGRect **pRectArray,
   CGRectCount *pCount
);
```

Parameters

pRectArray

A pointer to a `CGRect*` variable. On return, the variable contains an array of rectangles that bound the refreshed areas, specified in global coordinates. When you no longer need the array, you should deallocate it by calling `CGReleaseScreenRefreshRects` (page 452).

pCount

A pointer to a `CGRectCount` variable. On return, the variable contains the number of entries in the returned array of rectangles.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

In some applications it may be preferable to wait for screen refresh data synchronously, using this function. You should call this function in a thread other than the main event-processing thread.

As an alternative, Quartz also supports asynchronous notification—see `CGRegisterScreenRefreshCallback` (page 451). If refresh callback functions are registered, this function should not be used.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGRemoteOperation.h`

CGWaitForScreenUpdateRects

Waits for screen update operations.

```
CGError CGWaitForScreenUpdateRects (
   CGScreenUpdateOperation requestedOperations,
   CGScreenUpdateOperation *currentOperation,
   CGRect **pRectArray,
   size_t *pCount,
   CGScreenUpdateMoveDelta *pDelta
);
```

Parameters

requestedOperations

The desired types of screen update operations. There are several possible choices:

- Specify `kCGScreenUpdateOperationRefresh` if you want all move operations to be returned as refresh operations.
- Specify `(kCGScreenUpdateOperationRefresh | kCGScreenUpdateOperationMove)` if you want to distinguish between move and refresh operations.
- Add `kCGScreenUpdateOperationReducedDirtyRectangleCount` to the screen operations if you want to minimize the number of rectangles returned to represent changed areas of the display.

currentOperation

A pointer to a `CGScreenUpdateOperation` variable. On return, the variable indicates the type of update operation (refresh or move).

pRectArray

A pointer to a `CGRect*` variable. On return, the variable contains an array of rectangles that bound the updated areas, specified in global coordinates. When you no longer need the array, you should deallocate it by calling `CGReleaseScreenRefreshRects` (page 452).

pCount

A pointer to a `size_t` variable. On return, the variable contains the number of entries in the returned array of rectangles.

pDelta

A pointer to a `CGScreenUpdateMoveDelta` variable. On return, if the value of the `currentOperation` parameter is `kCGScreenUpdateOperationMove` the variable contains the distance moved.

Return Value

A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion

In some applications it may be preferable to wait for screen update data synchronously, using this function. You should call this function in a thread other than the main event-processing thread.

As an alternative, Quartz also supports asynchronous notification—see `CGRegisterScreenRefreshCallback` (page 451) and `CGScreenRegisterMoveCallback` (page 453). If refresh or move callback functions are registered, this function should not be used.

Special Considerations

This function is implemented in Mac OS X version 10.4.3 and later.

Availability

Available in Mac OS X v10.3 and later.

Declared In
CGRemoteOperation.h

CGWarpMouseCursorPosition

Moves the mouse cursor without generating events.

```
CGError CGWarpMouseCursorPosition (
   CGPoint newCursorPosition
);
```

Parameters

newCursorPosition
The new mouse cursor position in global display coordinates.

Return Value
A result code. See "Quartz Display Services Result Codes" (page 486).

Discussion
You can use this function to 'warp' or alter the cursor position without generating or posting an event. For example, this function is often used to move the cursor position back to the center of the screen by games that do not want the cursor pinned by display edges.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGRemoteOperation.h

CGWindowLevelForKey

Returns the window level that corresponds to one of the standard window types.

```
CGWindowLevel CGWindowLevelForKey (
   CGWindowLevelKey key
);
```

Parameters

key
A window level key constant that represents one of the standard window types. See "Window Level Keys" (page 482).

Return Value
The window level that corresponds to the specified key.

Discussion
This function is not recommended for use in applications. (This function is provided for application frameworks that create and manage windows, such as Carbon and Cocoa.)

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGWindowLevel.h

CGWindowServerCFMachPort

Returns a Core Foundation mach port (CFMachPort) that corresponds to the Mac OS X window server.

```
CFMachPortRef CGWindowServerCFMachPort (
   void
);
```

Return Value
A Core Foundation mach port, or `NULL` if the window server is not running. When you no longer need the port, you should release it using the function `CFRelease`.

Discussion
You can use this function to detect if the window server process exits or is not running. If this function returns `NULL`, the window server is not running. This code example shows how to register a callback function to detect when the window server exits:

```
static void handleWindowServerDeath( CFMachPortRef port, void *info  )
{
    printf( "Window Server port death detected!\n" );
    CFRelease(port);
    exit(1);
}

static void watchForWindowServerDeath()
{
    CFMachPortRef port = CGWindowServerCFMachPort();
    CFMachPortSetInvalidationCallBack(port, handleWindowServerDeath);
}
```

Note that this callback will not work unless your program has an active run loop.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`CGRemoteOperation.h`

Callbacks

CGDisplayReconfigurationCallBack

A client-supplied callback function that's invoked whenever the configuration of a local display is changed.

```
typedef void (*CGDisplayReconfigurationCallBack) (
   CGDirectDisplayID display,
   CGDisplayChangeSummaryFlags flags,
   void *userInfo
);
```

If you name your function `MyDisplayReconfigurationCallBack`, you would declare it like this:

```
void MyDisplayReconfigurationCallBack (
   CGDirectDisplayID display,
```

```
    CGDisplayChangeSummaryFlags flags,
    void *userInfo
);
```

Parameters

display

The display being reconfigured.

flags

Flags that indicate which display configuration parameters are changing.

userInfo

The `userInfo` argument passed to the function `CGDisplayRegisterReconfigurationCallback` (page 429) when the callback function is registered.

Discussion

To register a display reconfiguration callback function, you call the function `CGDisplayRegisterReconfigurationCallback` (page 429). Quartz invokes your callback function when:

- Your application calls a function to reconfigure a local display.
- Your application is listening for events in the event-processing thread, and another application calls a function to reconfigure a local display.
- The user changes the display hardware configuration—for example, by disconnecting a display or changing a system preferences setting.

Before display reconfiguration, Quartz invokes your callback function once for each online display to indicate a pending configuration change. The `flags` argument is always set to `kCGDisplayBeginConfigurationFlag`. Other than the display ID, this callback does not carry other per-display information, as details of how a reconfiguration affects a particular device rely on device-specific behaviors which may not be exposed by a device driver.

After display reconfiguration, Quartz invokes your callback function once for each added, removed, and online display. At this time, all display state reported by Core Graphics, QuickDraw, and the Carbon Display Manager will be up to date. This callback runs after the Carbon Display Manager notification callbacks. The `flags` argument indicates how the display configuration has changed. Note that in the case of removed displays, calls into Quartz with the removed display ID will fail.

The following code example illustrates how to test for specific conditions:

```
void MyDisplayReconfigurationCallBack (
    CGDirectDisplayID display,
    CGDisplayChangeSummaryFlags flags,
    void *userInfo)
{
     if (flags & kCGDisplayAddFlag) {
         // display has been added
     }
     else if (flags & kCGDisplayRemoveFlag) {
         // display has been removed
     }
}
```

Your callback function should avoid attempting to change display configurations, and should not raise exceptions or perform a non-local return such as calling `longjmp`. When you are finished using a callback registration, you should call the function `CGDisplayRemoveReconfigurationCallback` (page 430) to remove it.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGDisplayConfiguration.h`

CGScreenRefreshCallback

A client-supplied callback function that's invoked when an area of the display is modified or refreshed.

```
typedef void (*CGScreenRefreshCallback) (
   CGRectCount count,
   const CGRect * rectArray,
   void * userParameter
);
```

If you name your function `MyScreenRefreshCallback`, you would declare it like this:

```
void MyScreenRefreshCallback (
   CGRectCount count,
   const CGRect * rectArray,
   void * userParameter
);
```

Parameters

count
The number of rectangles in the `rectArray` parameter.

rectArray
A list of the rectangles in the refreshed areas, specified in global coordinates. You should not modify or deallocate memory pointed to by `rectArray`.

userParameter
The user data you specify when you register this callback.

Discussion
To register a screen refresh callback function, you call the function `CGRegisterScreenRefreshCallback` (page 451). Quartz invokes your callback function when operations such as drawing, window movement, scrolling, or display reconfiguration occur on local displays. When you are finished using a callback registration, you should call the function `CGUnregisterScreenRefreshCallback` (page 458) to remove it.

Note that a single rectangle may occupy multiple displays, either by overlapping the displays or by residing on coincident displays when mirroring is active. You can use the function `CGGetDisplaysWithRect` (page 440) to determine the displays a rectangle occupies.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGRemoteOperation.h

CGScreenUpdateMoveCallback

A client-supplied callback function that's invoked when an area of the display is moved.

```
typedef void (*CGScreenUpdateMoveCallback) (
   CGScreenUpdateMoveDelta delta,
   CGRectCount count,
   const CGRect * rectArray,
   void * userParameter
);
```

If you name your function `MyScreenUpdateMoveCallback`, you would declare it like this:

```
void MyScreenUpdateMoveCallback (
   CGScreenUpdateMoveDelta delta,
   CGRectCount count,
   const CGRect * rectArray,
   void * userParameter
);
```

Parameters

delta
The distance the display area has moved.

count
The number of rectangles in the `rectArray` parameter.

rectArray
A list of the rectangles in the moved areas, specified in global coordinates. The rectangles describe the area prior to the move operation. You should not modify or deallocate memory pointed to by `rectArray`.

userParameter
The user data you specify when you register this callback.

Discussion
To register a screen move callback function, you call the function `CGScreenRegisterMoveCallback` (page 453). Quartz invokes your callback function when operations such as window movement or scrolling occur on local displays. When you are finished using a callback registration, you should call the function `CGScreenUnregisterMoveCallback` (page 454) to remove it.

Note that a single rectangle may occupy multiple displays, either by overlapping the displays or by residing on coincident displays when mirroring is active. You can use the function `CGGetDisplaysWithRect` (page 440) to determine the displays a rectangle occupies.

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGRemoteOperation.h

Data Types

CGBeamPosition

Represents a horizontal scan line on a monitor that uses a scanning electron beam to refresh the screen.

```
typedef uint32_t CGBeamPosition;
```

Discussion
CRT and analog-driven displays use a horizontal scanning beam to refresh the screen. The beam position is a number assigned to a horizontal scan line on the screen. Scan lines are numbered `0` to `n-1` from top of screen, where `n` represents the total number of scan lines.

The concept of beam position does not apply to flat-panel LCD displays. While all displays have some concept of scan lines with respect to the frame buffer, LCD displays may not use linear scanning to refresh the screen.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGByteValue

Represents a unit of information in a byte-addressable array or data structure.

```
typedef uint8_t CGByteValue;
```

Discussion
Quartz uses `CGByteValue` to represent integer-based color values in a display palette or a gamma table. For example, see `CGDeviceByteColor` (page 466).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGDeviceByteColor

Represents a color in a Quartz display palette, using 8-bit integer components.

```
struct CGDeviceByteColor {
    CGByteValue red;
    CGByteValue green;
    CGByteValue blue;
};
typedef struct CGDeviceByteColor CGDeviceByteColor;
```

Fields
`red`
The red component of a palette entry.

`green`
The green component of a palette entry.

`blue`
The blue component of a palette entry.

Discussion
This data structure consists of three integer values that represent the intensity of the red,green, and blue components in a display palette entry. Each component ranges from 0 (no color) to 255 (full intensity).

Quartz provides `CGDeviceByteColor` to allow you to create a display palette using integer-based sample data. Once loaded, you can retrieve color data from the palette only as entries of type `CGDeviceColor` (page 467).

For more information about display palettes, see `CGDirectPaletteRef` (page 468).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGDeviceColor

Represents a color in a Quartz display palette.

```
struct CGDeviceColor {
    float red;
    float green;
    float blue;
};
typedef struct CGDeviceColor CGDeviceColor;
```

Fields
`red`
The red component of a palette entry.

`green`
The green component of a palette entry.

`blue`
The blue component of a palette entry.

Discussion
This data structure consists of three floating point values that represent the intensity of the red,green, and blue components in a display palette entry. Each component ranges from 0 (no color) to 1 (full intensity). Values outside this range are clamped to 0 or 1 when the palette is created.

Quartz uses `CGDeviceColor` as the canonical form for a color entry in a display palette. Palette entries can be created and retrieved in this form.

For more information about display palettes, see `CGDirectPaletteRef` (page 468).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectPalette.h

CGDirectDisplayID

Represents a unique identifier for an attached display.

```
typedef uint32_t CGDirectDisplayID;
```

Discussion
In Quartz, the term *display* refers to a graphics hardware system consisting of a framebuffer, a color correction (gamma) table or color palette, and possibly an attached monitor. If no monitor is attached, a display is characterized as offline.

When a monitor is attached, Quartz assigns a unique display identifier (ID). A display ID can persist across processes and system reboot, and typically remains constant as long as certain display parameters do not change.

When assigning a display ID, Quartz considers the following parameters:

- vendor
- model
- serial number
- position in the I/O Kit registry

For information about how to obtain a display ID, see "Finding Displays" (page 392).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDirectPaletteRef

Defines a reference to a Quartz 8-bit display palette.

```
typedef struct _CGDirectPaletteRef * CGDirectPaletteRef;
```

Discussion
A display palette is a bounded set of color values available for display. Some display operating modes have a maximum color depth of 8 bits (256 colors). The CGDirectPalette API is designed for application and game developers that want to create and use display palettes for these older displays.

Quartz uses reference counting to manage display palettes. See "Working With Color Palettes" (page 396) for more information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDisplayBlendFraction

Represents the percentage of blend color used in a fade operation.

```
typedef float CGDisplayBlendFraction;
```

Discussion
The blend fraction ranges from 0 (no color) to 1 (full intensity). If you specify 0, the blend color is not applied. If you specify 1, the user sees only the blend color on the screen.

In a fade operation, Quartz blends a color specified by the application with the current contents of the frame buffer. The blend color can be applied both at the beginning and the end of a fade operation.

Color blending during a fade operation is analogous to alpha blending in Quartz 2D, and the visual appearance is similar. However, the implementation is quite different. In a fade operation, the blend color is applied at the very end of the graphics pipeline, as the frame buffer is transferred to video output.

For example, the Universal Access preference panel in Mac OS X allows you to select a flashing screen effect (sometimes called a visual bell) to accompany the system alert sound. When you select this option, the system uses a Quartz fade operation to produce the flash. The blend color is applied using a blend fraction of 0.5 or 50%.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayConfigRef

Defines a reference to a display configuration transaction.

```
typedef struct _CGDisplayConfigRef * CGDisplayConfigRef;
```

Discussion
This data type makes it possible to

- create a new display configuration transaction using the function `CGBeginDisplayConfiguration` (page 399)
- record a set of configuration changes, each bound to one or more displays
- apply the changes in a single transaction using the function `CGCompleteDisplayConfiguration` (page 401), or discard the changes using the function `CGCancelDisplayConfiguration` (page 400)

There are no restrictions on the order in which you accumulate configuration changes in a transaction.

Configuration changes sometimes conflict with each other. For example, a new origin might be rendered invalid by a subsequent configuration change.

If possible, Quartz uses a “best fit” strategy to resolve conflicts between configuration changes. For example, when you change the resolution of a single display in a two-display system, Quartz automatically re-tiles the displays to prevent separation or overlap of the adjoining edges.

Availability
Available in Mac OS X v10.2 and later.

Declared In
CGDisplayConfiguration.h

CGDisplayCoord

Represents a coordinate position in global display space.

```
typedef int32_t CGDisplayCoord;
```

Discussion
Quartz uses `CGDisplayCoord` to represent the x- and y-coordinates of points in the per-display coordinate system. The origin is defined as the upper-left corner of the screen.

This data type is also used in functions that need to find the address of a specific pixel and monitor. For example, the function `CGDisplayAddressForPosition` (page 407) finds the address in frame buffer memory that corresponds to a given position or point.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDisplayCount

Represents the number of displays in various lists.

```
typedef uint32_t CGDisplayCount;
```

Discussion
Quartz uses `CGDisplayCount` to represent a count of either the current or the maximum number of displays in a display list. For example, see the function `CGGetActiveDisplayList` (page 437).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDisplayErr

Defines a uniform type for result codes returned by functions in Quartz Display Services.

```
typedef CGError CGDisplayErr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGDirectDisplay.h

CGDisplayFadeInterval

Represents the duration in seconds of a fade operation or a fade hardware reservation.

```
typedef float CGDisplayFadeInterval;
```

Discussion
Quartz uses this data type to specify the duration of both fade-out and fade-in operations. Values may range from zero to `kCGMaxDisplayReservationInterval` seconds. A zero value means fade immediately—see `CGDisplayFade` (page 417).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayFadeReservationToken

Defines a token issued by Quartz when reserving one or more displays for a fade operation during a specified interval.

```
typedef uint32_t CGDisplayFadeReservationToken;
```

Discussion
Quartz lets you reserve the display hardware to perform a fade operation. Fade reservations are valid for up to 15 seconds. Only one token is needed for both fade-out and fade-in.

You should release a fade reservation immediately when you no longer need it. If the reservation expires, releasing it is safe but not necessary.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGDisplayReservationInterval

Represents the time interval for a fade reservation.

```
typedef float CGDisplayReservationInterval;
```

Discussion
A fade reservation interval is a period of time during which a specific display is reserved for a fade operation. Fade reservation intervals range from 1 to `kCGMaxDisplayReservationInterval` seconds.

For more information about fade reservations, see the function `CGAcquireDisplayFadeReservation` (page 398). Fade reservation tokens are discussed in `CGDisplayFadeReservationToken` (page 471).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CGDisplayFade.h`

CGError

Defines a uniform type for result codes returned by functions in Quartz Services.

```
typedef int32_t CGError;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGError.h`

CGGammaValue

Represents information used to map a color generated in software to a color supported by the display hardware.

```
typedef float CGGammaValue;
```

Discussion
In Mac OS X, the Display panel in System Preferences is used to set the default gamma for a display. Quartz also allows an application to provide its own custom gamma information, using functions such as `CGSetDisplayTransferByTable` (page 457) and `CGSetDisplayTransferByFormula` (page 455).

These functions take `CGGammaValue` arguments that specify

- a set of gamma table entries ranging from 0 to 1
- the positive real coefficients in a gamma equation

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGMouseDelta

Represents a change in mouse position, in mouse units.

```
typedef int32_t CGMouseDelta;
```

Discussion
A mouse unit is a hardware-specific unit of measure, and generally has higher resolution than pixel units.

Note that the function `CGGetLastMouseDelta` (page 442) is no longer recommended—instead, you should use mouse tracking functions in the Carbon Event Manager.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGOpenGLDisplayMask

Defines a bitmask used in OpenGL to specify a set of attached displays.

```
typedef uint32_t CGOpenGLDisplayMask;
```

Discussion
In Mac OS X, OpenGL can provide information about the capabilities of the hardware renderers driving a specified set of displays. A 32-bit mask is used to specify the displays—each bit in the mask represents a single display.

To learn how to find the mask bit that corresponds to a given display, see the function `CGDisplayIDToOpenGLDisplayMask` (page 420).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGPaletteBlendFraction

Represents the intensity of a solid color used to tint a display palette.

```
typedef float CGPaletteBlendFraction;
```

Discussion
A palette blend-fraction value can range from 0 (no color) to 1 (full intensity). At full intensity, the palette is completely washed out by the color.

For more information, see the function `CGPaletteCreateFromPaletteBlendedWithColor` (page 446).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectPalette.h`

CGRectCount

Represents the size of an array of Quartz rectangles.

```
typedef uint32_t CGRectCount;
```

Discussion
For example, see the function `CGWaitForScreenRefreshRects` (page 459).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGRemoteOperation.h`

CGRefreshRate

Represents a display's refresh rate in frames per second.

```
typedef double CGRefreshRate;
```

Discussion
When requesting a new display mode, you can specify a desired refresh rate as a hint to Quartz. For example, see the function `CGDisplayBestModeForParametersAndRefreshRate` (page 410).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGScreenUpdateMoveDelta

Represents the distance a region on the screen moves in pixel units.

```
struct _CGScreenUpdateMoveDelta {
      int32_t dX, dY;
};
typedef struct _CGScreenUpdateMoveDelta CGScreenUpdateMoveDelta;
```

Discussion
Move operation notifications are restricted to changes that move a region by an integer number of pixels. The fields `dX` and `dY` describe the direction of movement:

- Positive values of `dX` indicate movement to the right.
- Negative values of `dX` indicate movement to the left.
- Positive values of `dY` indicate movement downward.
- Negative values of `dY` indicate movement upward.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`CGRemoteOperation.h`

CGTableCount

Defines a uniform type to represent the number of entries in a table.

```
typedef uint32_t CGTableCount;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGDirectDisplay.h`

CGWindowLevel

Represents a level assigned to a window by an application framework.

```
typedef int32_t CGWindowLevel;
```

Discussion
In Mac OS X, application frameworks support the concept of multiple window levels (or layers). Window levels are assigned and managed by each individual framework.

Note that in an Aqua-compliant application, each document window exists in its own layer. As a result, windows created by different applications can be interleaved.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGWindowLevel.h`

Constants

Display Capture Options

Specify configuration parameters when capturing displays.

```
enum {
    kCGCaptureNoOptions = 0,
    kCGCaptureNoFill = (1 << 0)
};
typedef uint32_t CGCaptureOptions;
```

Constants

`kCGCaptureNoOptions`
Specifies that the system should use the default fill behavior, which is fill with black.
Available in Mac OS X v10.3 and later.
Declared in `CGDirectDisplay.h`.

`kCGCaptureNoFill`
Disables fill with black.
Available in Mac OS X v10.3 and later.
Declared in `CGDirectDisplay.h`.

Discussion
For information about how these constants are used, see the functions `CGDisplayCaptureWithOptions` (page 415) and `CGCaptureAllDisplaysWithOptions` (page 401).

Display Configuration Change Flags

Specify the configuration parameters passed to a display reconfiguration callback function.

```
enum {
    kCGDisplayBeginConfigurationFlag = (1 << 0),
    kCGDisplayMovedFlag              = (1 << 1),
    kCGDisplaySetMainFlag            = (1 << 2),
    kCGDisplaySetModeFlag            = (1 << 3),
    kCGDisplayAddFlag                = (1 << 4),
    kCGDisplayRemoveFlag             = (1 << 5),
    kCGDisplayEnabledFlag            = (1 << 8),
    kCGDisplayDisabledFlag           = (1 << 9),
    kCGDisplayMirrorFlag             = (1 << 10),
    kCGDisplayUnMirrorFlag           = (1 << 11),
    kCGDisplayDesktopShapeChangedFlag = (1 << 12)
};
typedef u_int32_t CGDisplayChangeSummaryFlags;
```

Constants

`kCGDisplayBeginConfigurationFlag`
The display configuration is about to change.
Available in Mac OS X v10.3 and later.
Declared in `CGDisplayConfiguration.h`.

`kCGDisplayMovedFlag`
The location of the upper-left corner of the display in global display space has changed.
Available in Mac OS X v10.3 and later.
Declared in `CGDisplayConfiguration.h`.

`kCGDisplaySetMainFlag`
The display is now the main display.
Available in Mac OS X v10.3 and later.
Declared in `CGDisplayConfiguration.h`.

`kCGDisplaySetModeFlag`
The display mode has changed.
Available in Mac OS X v10.3 and later.
Declared in `CGDisplayConfiguration.h`.

`kCGDisplayAddFlag`
The display has been added to the active display list.
Available in Mac OS X v10.3 and later.
Declared in `CGDisplayConfiguration.h`.

`kCGDisplayRemoveFlag`
The display has been removed from the active display list.
Available in Mac OS X v10.3 and later.
Declared in `CGDisplayConfiguration.h`.

`kCGDisplayEnabledFlag`
The display has been enabled.
Available in Mac OS X v10.3 and later.
Declared in `CGDisplayConfiguration.h`.

`kCGDisplayDisabledFlag`
The display has been disabled.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayMirrorFlag`
The display is now mirroring another display.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayUnMirrorFlag`
The display is no longer mirroring another display.

Available in Mac OS X v10.3 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGDisplayDesktopShapeChangedFlag`
The shape of the desktop (the union of display areas) has changed.

Available in Mac OS X v10.5 and later.

Declared in `CGDisplayConfiguration.h`.

Discussion
For information about how these constants are used, see the callback `CGDisplayReconfigurationCallBack` (page 462).

Display Configuration Scopes

Specify the scope of the changes in a display configuration transaction.

```
enum {
    kCGConfigureForAppOnly = 0,
    kCGConfigureForSession = 1,
    kCGConfigurePermanently = 2
};
```

Constants
`kCGConfigureForAppOnly`
Specifies that changes persist for the lifetime of the current application. After the application terminates, the display configuration settings revert to the current login session.

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGConfigureForSession`
Specifies that changes persist for the lifetime of the current login session. After the current session terminates, the displays revert to the last saved permanent configuration.

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayConfiguration.h`.

`kCGConfigurePermanently`
Specifies that changes persist in future login sessions by the same user. If the requested changes cannot be supported by the Aqua UI (resolution and pixel depth constraints apply), the settings for the current login session are used instead, and any changes have session scope.

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayConfiguration.h`.

Discussion
For information about how these constants are used, see the function `CGCompleteDisplayConfiguration` (page 401).

Display Fade Blend Fractions

Specify the lower and upper bounds for blend color fractions during a display fade operation.

```
#define kCGDisplayBlendNormal (0.0)
#define kCGDisplayBlendSolidColor (1.0)
```

Constants
`kCGDisplayBlendNormal`
Specifies that the blend color is not applied at the start or end of a fade operation.

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayFade.h`.

`kCGDisplayBlendSolidColor`
Specifies that the user sees only the blend color at the start or end of a fade operation.

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayFade.h`.

Discussion
For general information about blend fractions, see the data type `CGDisplayBlendFraction` (page 469). For information about how these constants are used, see the function `CGDisplayFade` (page 417).

Display Fade Constants

Specifies values relating to fade operations.

```
#define kCGMaxDisplayReservationInterval (15.0)
#define kCGDisplayFadeReservationInvalidToken (0)
```

Constants
`kCGMaxDisplayReservationInterval`
Specifies the maximum number of seconds for fade hardware reservations and display fade operations. For general information about fade intervals, see the data type `CGDisplayFadeInterval` (page 471).

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayFade.h`.

`kCGDisplayFadeReservationInvalidToken`
Specifies an invalid fade reservation token. For general information about fade reservation tokens, see the data type `CGDisplayFadeReservationToken` (page 471).

Available in Mac OS X v10.2 and later.

Declared in `CGDisplayFade.h`.

Display ID Defaults

Default values for a display ID.

```
#define kCGDirectMainDisplay (CGMainDisplayID())
#define kCGNullDirectDisplay ((CGDirectDisplayID)0)
```

Constants

`kCGDirectMainDisplay`
Specifies the current main display ID.

Available in Mac OS X v10.0 and later.

Declared in `CGDirectDisplay.h`.

`kCGNullDirectDisplay`
Specifies a value that will never correspond to actual hardware.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

Display Mode Standard Properties

Specify keys for the standard properties in a display mode dictionary.

```
#define kCGDisplayWidth CFSTR("Width")
#define kCGDisplayHeight CFSTR("Height")
#define kCGDisplayMode CFSTR("Mode")
#define kCGDisplayBitsPerPixel CFSTR("BitsPerPixel")
#define kCGDisplayBitsPerSample CFSTR("BitsPerSample")
#define kCGDisplaySamplesPerPixel CFSTR("SamplesPerPixel")
#define kCGDisplayRefreshRate CFSTR("RefreshRate")
#define kCGDisplayModeUsableForDesktopGUICFSTR("UsableForDesktopGUI")
#define kCGDisplayIOFlags CFSTR("IOFlags")
#define kCGDisplayBytesPerRow CFSTR("kCGDisplayBytesPerRow")
```

Constants

`kCGDisplayWidth`
Specifies a CFNumber integer value that represents the width of the display in pixels.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayHeight`
Specifies a CFNumber integer value that represents the height of the display in pixels.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayMode`
Specifies a CFNumber integer value that represents the I/O Kit display mode number.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayBitsPerPixel`
Specifies a CFNumber integer value that represents the number of bits in a pixel.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayBitsPerSample`
Specifies a CFNumber integer value that represents the number of bits in an individual sample (for example, a color value in an RGB pixel).

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplaySamplesPerPixel`
Specifies a CFNumber integer value that represents the number of samples in a pixel.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayRefreshRate`
Specifies a CFNumber double-precision floating point value that represents the refresh rate of a CRT display. Some displays may not use conventional video vertical and horizontal sweep in painting the screen; these displays report a refresh rate of 0.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayModeUsableForDesktopGUI`
Specifies a CFBoolean value that indicates whether the display is suitable for use with the Mac OS X graphical user interface. The criteria include factors such as sufficient width and height and adequate pixel depth.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayIOFlags`
Specifies a CFNumber integer value that contains the I/O Kit display mode flags. For more information, see the header file `IOKit/IOGraphicsTypes.h`.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayBytesPerRow`
Specifies a CFNumber integer value that represents the number of bytes in a row on the display.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

Discussion
To learn how to use these keys to access the values in a display mode dictionary, see *CFDictionary Reference*.

Display Mode Optional Properties

Specify keys for optional properties in a display mode dictionary.

```
#define kCGDisplayModeIsSafeForHardware CFSTR ("kCGDisplayModeIsSafeForHardware")
#define kCGDisplayModeIsInterlaced CFSTR("kCGDisplayModeIsInterlaced")
#define kCGDisplayModeIsStretched CFSTR("kCGDisplayModeIsStretched")
#define kCGDisplayModeIsTelevisionOutput CFSTR ("kCGDisplayModeIsTelevisionOutput")
```

Constants

`kCGDisplayModeIsSafeForHardware`

Specifies a CFBoolean value indicating that the display mode doesn't need a confirmation dialog to be set.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayModeIsInterlaced`

Specifies a CFBoolean value indicating that the I/O Kit interlace mode flag is set.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayModeIsStretched`

Specifies a CFBoolean value indicating that the I/O Kit stretched mode flag is set.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

`kCGDisplayModeIsTelevisionOutput`

Specifies a CFBoolean value indicating that the I/O Kit television output mode flag is set.

Available in Mac OS X v10.2 and later.

Declared in `CGDirectDisplay.h`.

Discussion

A given key is present in a display mode dictionary only if the property applies, and is always associated with a value of `kCFBooleanTrue`. Keys not relevant to a particular display mode will not appear in the mode dictionary.

Reserved Window Levels

Specifies window level constants.

```
#define kCGNumReservedWindowLevels      (16)
```

Constants

`kCGNumReservedWindowLevels`

The number of window levels reserved by Apple for internal use. Application frameworks such as Carbon and Cocoa use this constant during compilation

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

Screen Update Operations

Specify types of screen update operations.

```
enum _CGScreenUpdateOperation {
    kCGScreenUpdateOperationRefresh = 0,
    kCGScreenUpdateOperationMove = (1 << 0),
    kCGScreenUpdateOperationReducedDirtyRectangleCount = (1 <<  31)
};
typedef uint32_t CGScreenUpdateOperation;
```

Constants

`kCGScreenUpdateOperationRefresh`
Specifies a screen refresh operation.
Available in Mac OS X v10.3 and later.
Declared in `CGRemoteOperation.h`.

`kCGScreenUpdateOperationMove`
Specifies a screen move operation.
Available in Mac OS X v10.3 and later.
Declared in `CGRemoteOperation.h`.

`kCGScreenUpdateOperationReducedDirtyRectangleCount`
When presented as part of the requested operations to the function `CGWaitForScreenUpdateRects` (page 460), specifies that the function should try to minimize the number of rectangles returned to represent the changed areas of the display. The function may combine adjacent rectangles within a larger bounding rectangle, which may include unmodified areas of the display.
Available in Mac OS X v10.4 and later.
Declared in `CGRemoteOperation.h`.

Discussion

For information about how these constants are used, see the function `CGWaitForScreenUpdateRects` (page 460).

Window Level Keys

Keys that represent the standard window levels in Mac OS X. Quartz includes these keys to support application frameworks such as Carbon and Cocoa. Applications do not need to use them directly.

```
enum _CGCommonWindowLevelKey {
 kCGBaseWindowLevelKey = 0,
 kCGMinimumWindowLevelKey,
 kCGDesktopWindowLevelKey,
 kCGBackstopMenuLevelKey,
 kCGNormalWindowLevelKey,
 kCGFloatingWindowLevelKey,
 kCGTornOffMenuWindowLevelKey,
 kCGDockWindowLevelKey,
 kCGMainMenuWindowLevelKey,
 kCGStatusWindowLevelKey,
 kCGModalPanelWindowLevelKey,
 kCGPopUpMenuWindowLevelKey,
 kCGDraggingWindowLevelKey,
 kCGScreenSaverWindowLevelKey,
 kCGMaximumWindowLevelKey,
 kCGOverlayWindowLevelKey,
 kCGHelpWindowLevelKey,
 kCGUtilityWindowLevelKey,
 kCGDesktopIconWindowLevelKey,
 kCGCursorWindowLevelKey,
 kCGNumberOfWindowLevelKeys
};
typedef int32_t CGWindowLevelKey;
```

Constants

`kCGBaseWindowLevelKey`
The base key used to define window levels. Do not use.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGMinimumWindowLevelKey`
The lowest available window level.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGDesktopWindowLevelKey`
The level for the desktop.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGBackstopMenuLevelKey`
The level of the backstop menu.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGNormalWindowLevelKey`
The level for normal windows.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGFloatingWindowLevelKey`
The level for floating windows.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGTornOffMenuWindowLevelKey`
The level for torn off menus.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGDockWindowLevelKey`
The level for the dock.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGMainMenuWindowLevelKey`
The level for the menus displayed in the menu bar.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGStatusWindowLevelKey`
The level for status windows.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGModalPanelWindowLevelKey`
The level for modal panels.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGPopUpMenuWindowLevelKey`
The level for pop-up menus.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGDraggingWindowLevelKey`
The level for a window being dragged.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGScreenSaverWindowLevelKey`
The level for screen savers.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGMaximumWindowLevelKey`
The highest allowed window level.
Available in Mac OS X v10.0 and later.
Declared in `CGWindowLevel.h`.

`kCGOverlayWindowLevelKey`
The level for overlay windows.
Available in Mac OS X v10.1 and later.
Declared in `CGWindowLevel.h`.

`kCGHelpWindowLevelKey`
The level for help windows.

Available in Mac OS X v10.1 and later.

Declared in `CGWindowLevel.h`.

`kCGUtilityWindowLevelKey`
The level for utility windows.

Available in Mac OS X v10.1 and later.

Declared in `CGWindowLevel.h`.

`kCGDesktopIconWindowLevelKey`
The level for desktop icons.

Available in Mac OS X v10.1 and later.

Declared in `CGWindowLevel.h`.

`kCGCursorWindowLevelKey`
The level for the cursor.

Available in Mac OS X v10.2 and later.

Declared in `CGWindowLevel.h`.

`kCGNumberOfWindowLevelKeys`
The total number of window levels.

Available in Mac OS X v10.0 and later.

Declared in `CGWindowLevel.h`.

Window Server Session Properties

Specify keys for the standard properties in a window server session dictionary.

```
#define kCGSessionUserIDKey CFSTR("kCGSSessionUserIDKey")
#define kCGSessionUserNameKey CFSTR("kCGSSessionUserNameKey")
#define kCGSessionConsoleSetKey CFSTR("kCGSSessionConsoleSetKey")
#define kCGSessionOnConsoleKey CFSTR("kCGSSessionOnConsoleKey")
#define kCGSessionLoginDoneKey CFSTR("kCGSessionLoginDoneKey")
```

Constants

`kCGSessionUserIDKey`
Specifies a CFNumber 32-bit unsigned integer value that encodes a user ID for the session's current user.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

`kCGSessionUserNameKey`
Specifies a CFString value that encodes the session's short user name as set by the login operation.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

`kCGSessionConsoleSetKey`
: Specifies a CFNumber 32-bit unsigned integer value that represents a set of hardware composing a console.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

`kCGSessionOnConsoleKey`
: Specifies a CFBoolean value indicating whether the session is on a console.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

`kCGSessionLoginDoneKey`
: Specifies a CFBoolean value indicating whether the login operation has been done.

Available in Mac OS X v10.3 and later.

Declared in `CGSession.h`.

Discussion
To learn how to use these keys to access the values in a session dictionary, see *CFDictionary Reference*.

Result Codes

This table lists the result codes returned by functions in Quartz Display Services.

Result Code	Value	Description
`kCGErrorSuccess`	0	The requested operation was completed successfully. Available in Mac OS X v10.0 and later.
`kCGErrorFailure`	1000	A general failure occurred. Available in Mac OS X v10.0 and later.
`kCGErrorIllegalArgument`	1001	One or more of the parameters passed to a function are invalid. Check for `NULL` pointers. Available in Mac OS X v10.0 and later.
`kCGErrorInvalidConnection`	1002	The parameter representing a connection to the window server is invalid. Available in Mac OS X v10.0 and later.
`kCGErrorInvalidContext`	1003	The `CPSProcessSerNum` or context identifier parameter is not valid. Available in Mac OS X v10.0 and later.
`kCGErrorCannotComplete`	1004	The requested operation is inappropriate for the parameters passed in, or the current system state. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kCGErrorNameTooLong	1005	A parameter, typically a C string, is too long to be used without truncation. Available in Mac OS X v10.0 and later.
kCGErrorNotImplemented	1006	Return value from obsolete function stubs present for binary compatibility, but not normally called. Available in Mac OS X v10.0 and later.
kCGErrorRangeCheck	1007	A parameter passed in has a value that is inappropriate, or which does not map to a useful operation or value. Available in Mac OS X v10.0 and later.
kCGErrorTypeCheck	1008	A data type or token was encountered that did not match the expected type or token. Available in Mac OS X v10.0 and later.
kCGErrorNoCurrentPoint	1009	An operation relative to a known point or coordinate could not be done, as there is no known point. Available in Mac OS X v10.0 and later.
kCGErrorInvalidOperation	1010	The requested operation is not valid for the parameters passed in, or the current system state. Available in Mac OS X v10.0 and later.
kCGErrorNoneAvailable	1011	The requested operation could not be completed as the indicated resources were not found. Available in Mac OS X v10.0 and later.

Quartz Event Services Reference

Framework: ApplicationServices/ApplicationServices.h

Declared in CGEvent.h
CGEventSource.h
CGEventTypes.h
CGRemoteOperation.h

Overview

This document describes the C API for event taps, which are filters used to observe and alter the stream of low-level user input events in Mac OS X. Event taps make it possible to monitor and filter input events from several points within the system, prior to their delivery to a foreground application. Event taps complement and extend the capabilities of the Carbon event monitor mechanism, which allows an application to observe input events delivered to other processes (see the function `GetEventMonitorTarget`).

Event taps are designed to serve as a Section 508 enabling technology. For example, consider a software system to assist a person with language impairments, designed to perform keyboard filtering with spoken review. Such a system could use an event tap to monitor all keystrokes, perform dictionary checks and matches, and recite the assembled word back to the user on detection of a word break in the input stream. If acceptable to the user, as indicated by an additional input keystroke or other gesture, the events would be posted into the system for delivery to the foreground application.

Introduced in Mac OS X version 10.4, event taps provide functionality similar to the Win32 functions `SetWinEventHook` when used to establish an out-of-context event hook, and `SendInput`. Quartz Event Services also includes an older set of event-related functions declared in the file `CGRemoteOperation.h`. These functions are still supported, but they are not recommended for new development.

Functions by Task

Working With Quartz Events

`CGEventGetTypeID` (page 501)
Returns the type identifier for the opaque type `CGEventRef`.

`CGEventCreate` (page 493)
Returns a new Quartz event.

`CGEventCreateData` (page 494)
Returns a flattened data representation of a Quartz event.

`CGEventCreateFromData` (page 495)
Returns a Quartz event created from a flattened data representation of the event.

`CGEventCreateMouseEvent` (page 496)
Returns a new Quartz mouse event.

`CGEventCreateKeyboardEvent` (page 495)
Returns a new Quartz keyboard event.

`CGEventCreateScrollWheelEvent` (page 497)
Returns a new Quartz scrolling event.

`CGEventCreateCopy` (page 494)
Returns a copy of an existing Quartz event.

`CGEventCreateSourceFromEvent` (page 498)
Returns a Quartz event source created from an existing Quartz event.

`CGEventSetSource` (page 507)
Sets the event source of a Quartz event.

`CGEventGetType` (page 501)
Returns the event type of a Quartz event (left mouse down, for example).

`CGEventSetType` (page 508)
Sets the event type of a Quartz event (left mouse down, for example).

`CGEventGetTimestamp` (page 500)
Returns the timestamp of a Quartz event.

`CGEventSetTimestamp` (page 507)
Sets the timestamp of a Quartz event.

`CGEventGetLocation` (page 500)
Returns the location of a Quartz mouse event.

`CGEventGetUnflippedLocation` (page 501)
Returns the location of a Quartz mouse event.

`CGEventSetLocation` (page 506)
Sets the location of a Quartz mouse event.

`CGEventGetFlags` (page 499)
Returns the event flags of a Quartz event.

`CGEventSetFlags` (page 505)
Sets the event flags of a Quartz event.

`CGEventKeyboardGetUnicodeString` (page 502)
Returns the Unicode string associated with a Quartz keyboard event.

`CGEventKeyboardSetUnicodeString` (page 503)
Sets the Unicode string associated with a Quartz keyboard event.

`CGEventGetIntegerValueField` (page 499)
Returns the integer value of a field in a Quartz event.

`CGEventSetIntegerValueField` (page 506)
Sets the integer value of a field in a Quartz event.

`CGEventGetDoubleValueField` (page 498)
Returns the floating-point value of a field in a Quartz event.

`CGEventSetDoubleValueField` (page 505)
Sets the floating-point value of a field in a Quartz event.

Working With Quartz Event Taps

`CGEventTapCreate` (page 517)
Creates an event tap.

`CGEventTapCreateForPSN` (page 519)
Creates an event tap for a specified process.

`CGEventTapEnable` (page 520)
Enables or disables an event tap.

`CGEventTapIsEnabled` (page 520)
Returns a Boolean value indicating whether an event tap is enabled.

`CGEventTapPostEvent` (page 521)
Posts a Quartz event from an event tap into the event stream.

`CGEventPost` (page 504)
Posts a Quartz event into the event stream at a specified location.

`CGEventPostToPSN` (page 504)
Posts a Quartz event into the event stream for a specific application.

`CGGetEventTapList` (page 521)
Gets a list of currently installed event taps.

`CGEventMaskBit` (page 503)
Generates an event mask for a single type of event.

Working With Quartz Event Sources

`CGEventSourceGetTypeID` (page 513)
Returns the type identifier for the opaque type `CGEventSourceRef`.

`CGEventSourceCreate` (page 509)
Returns a Quartz event source created with a specified source state.

`CGEventSourceGetKeyboardType` (page 510)
Returns the keyboard type to be used with a Quartz event source.

`CGEventSourceSetKeyboardType` (page 515)
Sets the keyboard type to be used with a Quartz event source.

`CGEventSourceGetSourceStateID` (page 512)
Returns the source state associated with a Quartz event source.

`CGEventSourceButtonState` (page 508)
Returns a Boolean value indicating the current button state of a Quartz event source.

`CGEventSourceKeyState` (page 514)
Returns a Boolean value indicating the current keyboard state of a Quartz event source.

`CGEventSourceFlagsState` (page 510)
Returns the current flags of a Quartz event source.

`CGEventSourceSecondsSinceLastEventType` (page 514)
Returns the elapsed time since the last event for a Quartz event source.

`CGEventSourceCounterForEventType` (page 509)
Returns a count of events of a given type seen since the window server started.

`CGEventSourceGetUserData` (page 513)
Returns the 64-bit user-specified data for a Quartz event source.

`CGEventSourceSetUserData` (page 517)
Sets the 64-bit user-specified data for a Quartz event source.

`CGEventSourceGetLocalEventsFilterDuringSuppressionState` (page 511)
Returns the mask that indicates which classes of local hardware events are enabled during event suppression.

`CGEventSourceSetLocalEventsFilterDuringSuppressionState` (page 515)
Sets the mask that indicates which classes of local hardware events are enabled during event suppression.

`CGEventSourceGetLocalEventsSuppressionInterval` (page 511)
Returns the interval that local hardware events may be suppressed following the posting of a Quartz event.

`CGEventSourceSetLocalEventsSuppressionInterval` (page 516)
Sets the interval that local hardware events may be suppressed following the posting of a Quartz event.

`CGEventSourceGetPixelsPerLine` (page 512)
Gets the scale of pixels per line in a scrolling event source.

`CGEventSourceSetPixelsPerLine` (page 516)
Sets the scale of pixels per line in a scrolling event source.

Obsolete Functions

`CGPostKeyboardEvent` (page 522)
Synthesizes a low-level keyboard event on the local machine.

`CGPostMouseEvent` (page 523)
Synthesizes a low-level mouse-button event on the local machine.

`CGPostScrollWheelEvent` (page 524)
Synthesizes a low-level scrolling event on the local machine.

`CGEnableEventStateCombining` (page 493)
Enables or disables the merging of actual key and mouse state with the application-specified state in a synthetic event.

`CGInhibitLocalEvents` (page 522)
Turns off local hardware events in the current session.

`CGSetLocalEventsFilterDuringSuppressionState` (page 525)
Filters local hardware events from the keyboard and mouse during the short interval after a synthetic event is posted.

`CGSetLocalEventsSuppressionInterval` (page 525)
Sets the time interval in seconds that local hardware events are suppressed after posting a synthetic event.

`CGEventGetSource` (page 500) Deprecated in Mac OS X v10.4
Returns a Quartz event source created from an existing Quartz event. (Deprecated. Use `CGEventCreateSourceFromEvent` (page 498) instead.)

Functions

CGEnableEventStateCombining

Enables or disables the merging of actual key and mouse state with the application-specified state in a synthetic event.

```
CGError CGEnableEventStateCombining (
   boolean_t doCombineState
);
```

Parameters

doCombineState

Pass `true` to specify that the actual key and mouse state are merged with the application-specified state in a synthetic event; otherwise, pass `false`.

Return Value

A result code. See the result codes described in *Quartz Display Services Reference*.

Discussion

By default, the flags that indicate modifier key state (Command, Option, Shift, Control, and so on) from the system's keyboard and from other event sources are ORed together as an event is posted into the system, and current key and mouse button state is considered in generating new events. This function allows your application to enable or disable the merging of event state. When combining is turned off, the event state propagated in the events posted by your application reflect state built up only by your application. The state within your application's generated event will not be combined with the system's current state, so the system-wide state reflecting key and mouse button state will remain unchanged. When called with `doCombineState` equal to `false`, this function initializes local (per application) state tracking information to a state of all keys, modifiers, and mouse buttons up. When called with `doCombineState` equal to `true`, the current global state of keys, modifiers, and mouse buttons are used in generating events.

This function is not recommended for general use because of undocumented special cases and undesirable side effects. The recommended replacement for this function is to use Quartz events and Quartz event sources. This allows you to control exactly which, if any, external event sources will contribute to the state used to create an event.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`CGRemoteOperation.h`

CGEventCreate

Returns a new Quartz event.

```
CGEventRef CGEventCreate (
   CGEventSourceRef source
);
```

Parameters

source

The event source, or `NULL` to use a default source.

Return Value

A new event to be filled in, or `NULL` if the event could not be created. When you no longer need the event, you should release it using the function `CFRelease`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGEvent.h`

CGEventCreateCopy

Returns a copy of an existing Quartz event.

```
CGEventRef CGEventCreateCopy (
   CGEventRef event
);
```

Parameters

event

The event being copied.

Return Value

A copy of the specified event. When you no longer need the copy, you should release it using the function `CFRelease`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGEvent.h`

CGEventCreateData

Returns a flattened data representation of a Quartz event.

```
CFDataRef CGEventCreateData (
   CFAllocatorRef allocator,
   CGEventRef event
);
```

Parameters

allocator

The allocator to use to allocate memory for the data object. To use the current default allocator, pass `NULL` or `kCFAllocatorDefault`.

event
The event to flatten.

Return Value
The flattened data representation of the event, or `NULL` if the `event` parameter is invalid. When you no longer need the data object, you should release it using the function `CFRelease`.

Discussion
You can use this function to flatten an event for network transport to another system.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventCreateFromData

Returns a Quartz event created from a flattened data representation of the event.

```
CGEventRef CGEventCreateFromData (
   CFAllocatorRef allocator,
   CFDataRef eventData
);
```

Parameters

allocator
The allocator to use to allocate memory for the event object. To use the current default allocator, pass `NULL` or `kCFAllocatorDefault`.

eventData
The flattened data representation of the event to reconstruct.

Return Value
An event built from the flattened data representation, or `NULL` if the `eventData` parameter is invalid.

Discussion
You can use this function to reconstruct a Quartz event received by network transport from another system.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventCreateKeyboardEvent

Returns a new Quartz keyboard event.

```
CGEventRef CGEventCreateKeyboardEvent (
   CGEventSourceRef source,
   CGKeyCode virtualKey,
   bool keyDown
);
```

Parameters

source

An event source taken from another event, or `NULL`.

virtualKey

The virtual key code for the event.

keyDown

Pass `true` to specify that the key position is down. To specify that the key position is up, pass `false`. This value is used to determine the type of the keyboard event—see "Event Types" (page 545).

Return Value

A new keyboard event, or `NULL` if the event could not be created. When you no longer need the event, you should release it using the function `CFRelease`.

Discussion

All keystrokes needed to generate a character must be entered, including modifier keys. For example, to produce a 'Z', the SHIFT key must be down, the 'z' key must go down, and then the SHIFT and 'z' key must be released:

```
CGEventRef event1, event2, event3, event4;
event1 = CGEventCreateKeyboardEvent (NULL, (CGKeyCode)56, true);
event2 = CGEventCreateKeyboardEvent (NULL, (CGKeyCode)6, true);
event3 = CGEventCreateKeyboardEvent (NULL, (CGKeyCode)6, false);
event4 = CGEventCreateKeyboardEvent (NULL, (CGKeyCode)56, false);
```

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGEvent.h`

CGEventCreateMouseEvent

Returns a new Quartz mouse event.

```
CGEventRef CGEventCreateMouseEvent (
   CGEventSourceRef source,
   CGEventType mouseType,
   CGPoint mouseCursorPosition,
   CGMouseButton mouseButton
);
```

Parameters

source

An event source taken from another event, or `NULL`.

mouseType

A mouse event type. Pass one of the constants listed in "Event Types" (page 545).

mouseCursorPosition
The position of the mouse cursor in global coordinates.

mouseButton
The button that's changing state. Pass one of the constants listed in "Mouse Buttons" (page 548). This parameter is ignored unless the *mouseType* parameter is `kCGEventOtherMouseDown`, `kCGEventOtherMouseDragged`, or `kCGEventOtherMouseUp`.

Return Value
A new mouse event, or `NULL` if the event could not be created. When you no longer need the event, you should release it using the function `CFRelease`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventCreateScrollWheelEvent

Returns a new Quartz scrolling event.

```
CGEventRef CGEventCreateScrollWheelEvent (
    CGEventSourceRef source,
    CGScrollEventUnit units,
    CGWheelCount wheelCount,
    int32_t wheel1,

);
```

Parameters

source
An event source taken from another event, or `NULL`.

units
The unit of measurement for the scrolling event. Pass one of the constants listed in "Scrolling Event Units" (page 549).

wheelCount
The number of scrolling devices on the mouse, up to a maximum of 3.

wheel1
A value that reflects the movement of the primary scrolling device on the mouse. Scrolling movement is generally represented by small signed integer values, typically in a range from -10 to +10. Large values may have unexpected results, depending on the application that processes the event.

...
Up to two values that reflect the movements of the other scrolling devices on the mouse, if any.

Return Value
A new scrolling event, or `NULL` if the event could not be created. When you no longer need the event, you should release it using the function `CFRelease`.

Discussion
This function allows you to create a scrolling event and customize the event before posting it to the event system.

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

Declared In
`CGEvent.h`

CGEventCreateSourceFromEvent

Returns a Quartz event source created from an existing Quartz event.

```
CGEventSourceRef CGEventCreateSourceFromEvent (
   CGEventRef event
);
```

Parameters

event

The event to access.

Return Value
An event source created from the specified event, or `NULL` if the event was generated with a private event source owned by another process. When you no longer need this event source, you should release it using the function `CFRelease`.

Discussion
Event filters may use the event source to generate events that are compatible with an event being filtered.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventGetDoubleValueField

Returns the floating-point value of a field in a Quartz event.

```
double CGEventGetDoubleValueField (
   CGEventRef event,
   CGEventField field
);
```

Parameters

event

The event to access.

field

A field in the specified event. Pass one of the constants listed in "Event Fields" (page 532).

Return Value
A floating point representation of the current value of the specified field.

Discussion
In cases where the field value is represented within the event by a fixed point number or an integer, the result is scaled to the appropriate range as part of creating the floating point representation.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventGetFlags

Returns the event flags of a Quartz event.

```
CGEventFlags CGEventGetFlags (
   CGEventRef event
);
```

Parameters

event
The event to access.

Return Value
The current flags of the specified event. For more information, see "Event Flags" (page 540).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventGetIntegerValueField

Returns the integer value of a field in a Quartz event.

```
int64_t CGEventGetIntegerValueField (
   CGEventRef event,
   CGEventField field
);
```

Parameters

event
The event to access.

field
A field in the specified event. Pass one of the constants listed in "Event Fields" (page 532).

Return Value
A 64-bit integer representation of the current value of the specified field.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventGetLocation

Returns the location of a Quartz mouse event.

```
CGPoint CGEventGetLocation (
   CGEventRef event
);
```

Parameters

event

The mouse event to locate.

Return Value

The current location of the specified mouse event in global display coordinates.

Availability

Available in Mac OS X v10.4 and later.

Declared In

CGEvent.h

CGEventGetSource

Returns a Quartz event source created from an existing Quartz event. (Deprecated in Mac OS X v10.4. Use `CGEventCreateSourceFromEvent` (page 498) instead.)

```
CGEventSourceRef CGEventGetSource (
    CGEventRef event
);
```

Availability

Available in Mac OS X v10.4 through Mac OS X v10.4.

Deprecated in Mac OS X v10.4.

Declared In

CGEvent.h

CGEventGetTimestamp

Returns the timestamp of a Quartz event.

```
CGEventTimestamp CGEventGetTimestamp (
   CGEventRef event
);
```

Parameters

event

The event to access.

Return Value

The current timestamp of the specified event.

Availability

Available in Mac OS X v10.4 and later.

Declared In
CGEvent.h

CGEventGetType

Returns the event type of a Quartz event (left mouse down, for example).

```
CGEventType CGEventGetType (
   CGEventRef event
);
```

Parameters

event
The event to access.

Return Value
The current event type of the specified event. The return value is one of the constants listed in "Event Types" (page 545).

Availability
Available in Mac OS X v10.4 and later.

See Also
CGEventSetType (page 508)

Declared In
CGEvent.h

CGEventGetTypeID

Returns the type identifier for the opaque type CGEventRef.

```
CFTypeID CGEventGetTypeID (
   void
);
```

Return Value
The Core Foundation type identifier for the opaque type CGEventRef (page 528).

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEvent.h

CGEventGetUnflippedLocation

Returns the location of a Quartz mouse event.

```
CGPoint CGEventGetUnflippedLocation (
    CGEventRef event
);
```

Parameters

event

The mouse event whose location you wish to obtain.

Return Value

The current location of the specified mouse event relative to the lower-left corner of the main display.

Discussion

This function returns the location of the mouse cursor associated with the event. The coordinate system used is relative to the lower-left corner of the main display, and is compatible with the global coordinate system used by the Application Kit.

Note that the y-coordinate of the returned location is off by one from an idealized coordinate system originating at the lower-left corner of the main display. Effectively, the function is defined as follows:

```
CGPoint p = CGEventGetLocation(event);
p.y = main_display_height - p.y;
/* not p.y = (main_display_height - 1) - p.y */
return p;
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CGEvent.h`

CGEventKeyboardGetUnicodeString

Returns the Unicode string associated with a Quartz keyboard event.

```
void CGEventKeyboardGetUnicodeString (
   CGEventRef event,
   UniCharCount maxStringLength,
   UniCharCount *actualStringLength,
   UniChar unicodeString[]
);
```

Parameters

event

The keyboard event to access.

maxStringLength

The length of the array you provide in the `unicodeString` parameter.

actualStringLength

A pointer to a `UniCharCount` variable. On return, the variable contains the actual count of Unicode characters in the event data.

unicodeString

A pointer to a `UniChar` array. You are responsible for allocating storage for the array. On return, your array contains the Unicode string associated with the specified event.

Discussion
When you call this function and specify a `NULL` string or a maximum string length of 0, the function still returns the actual count of Unicode characters in the event data.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventKeyboardSetUnicodeString

Sets the Unicode string associated with a Quartz keyboard event.

```
void CGEventKeyboardSetUnicodeString (
   CGEventRef event,
   UniCharCount stringLength,
   const UniChar unicodeString[]
);
```

Parameters

event
The keyboard event to access.

stringLength
The length of the array you provide in the `unicodeString` parameter.

unicodeString
An array that contains the new Unicode string associated with the specified event.

Discussion
By default, the system translates the virtual key code in a keyboard event into a Unicode string based on the keyboard ID in the event source. This function allows you to manually override this string. Note that application frameworks may ignore the Unicode string in a keyboard event and do their own translation based on the virtual keycode and perceived event state.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventMaskBit

Generates an event mask for a single type of event.

```
CGEventMask CGEventMaskBit (
   CGEventType eventType
);
```

Parameters

eventType
An event type constant. Pass one of the constants listed in "Event Types" (page 545).

Return Value
An event mask that represents the specified event.

Discussion
This macro converts an event type constant into a mask. You can use this mask to specify that an event tap should observe the event. For more information, see `CGEventMask` (page 528).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventTypes.h`

CGEventPost

Posts a Quartz event into the event stream at a specified location.

```
void CGEventPost (
   CGEventTapLocation tap,
   CGEventRef event
);
```

Parameters

tap
The location at which to post the event. Pass one of the constants listed in "Event Tap Locations" (page 543).

event
The event to post.

Discussion
This function posts the specified event immediately before any event taps instantiated for that location, and the event passes through any such taps.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventPostToPSN

Posts a Quartz event into the event stream for a specific application.

```
void CGEventPostToPSN (
   void *processSerialNumber,
   CGEventRef event
);
```

Parameters

processSerialNumber
The process to receive the event.

event
The event to post.

Discussion
This function makes it possible for an application to establish an event routing policy, for example, by tapping events at the `kCGAnnotatedSessionEventTap` location and then posting the events to another desired process.

This function posts the specified event immediately before any event taps instantiated for the specified process, and the event passes through any such taps.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventSetDoubleValueField

Sets the floating-point value of a field in a Quartz event.

```
void CGEventSetDoubleValueField (
   CGEventRef event,
   CGEventField field,
   double value
);
```

Parameters

event
The event to access.

field
A field in the specified event. Pass one of the constants listed in "Event Fields" (page 532).

value
The new value of the specified field.

Discussion
Before calling this function, the event type must be set using a typed event creation function such as `CGEventCreateMouseEvent` (page 496), or by calling `CGEventSetType` (page 508).

In cases where the field's value is represented within the event by a fixed point number or integer, the `value` parameter is scaled as needed and converted to the appropriate type.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventSetFlags

Sets the event flags of a Quartz event.

```
void CGEventSetFlags (
   CGEventRef event,
   CGEventFlags flags
);
```

Parameters

event

The event to access.

location

The new flags of the specified event. See "Event Flags" (page 540).

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEvent.h

CGEventSetIntegerValueField

Sets the integer value of a field in a Quartz event.

```
void CGEventSetIntegerValueField (
   CGEventRef event,
   CGEventField field,
   int64_t value
);
```

Parameters

event

The event to access.

field

A field in the specified event. Pass one of the constants listed in "Event Fields" (page 532).

value

The new value of the specified field.

Discussion
Before calling this function, the event type must be set using a typed event creation function such as `CGEventCreateMouseEvent` (page 496), or by calling `CGEventSetType` (page 508).

If you are creating a mouse event generated by a tablet, call this function and specify the field `kCGMouseEventSubtype` with a value of `kCGEventMouseSubtypeTabletPoint` or `kCGEventMouseSubtypeTabletProximity` before setting other parameters.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEvent.h

CGEventSetLocation

Sets the location of a Quartz mouse event.

```
void CGEventSetLocation (
   CGEventRef event,
   CGPoint location
);
```

Parameters

event

The mouse event whose location to set.

location

The new location of the specified mouse event in global display coordinates.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEvent.h

CGEventSetSource

Sets the event source of a Quartz event.

```
void CGEventSetSource (
   CGEventRef event,
   CGEventSourceRef source
);
```

Parameters

event

The event to access.

source

The new event source of the specified event.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEvent.h

CGEventSetTimestamp

Sets the timestamp of a Quartz event.

```
void CGEventSetTimestamp (
   CGEventRef event,
   CGEventTimestamp timestamp
);
```

Parameters

event

The event to access.

timestamp

The new timestamp of the specified event.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventSetType

Sets the event type of a Quartz event (left mouse down, for example).

```
void CGEventSetType (
   CGEventRef event,
   CGEventType type
);
```

Parameters

event
The event to access.

type
The new event type of the specified event. The return value is one of the constants listed in "Event Types" (page 545).

Availability
Available in Mac OS X v10.4 and later.

See Also
`CGEventGetType` (page 501)

Declared In
`CGEvent.h`

CGEventSourceButtonState

Returns a Boolean value indicating the current button state of a Quartz event source.

```
bool CGEventSourceButtonState (
   CGEventSourceStateID sourceState,
   CGMouseButton button
);
```

Parameters

sourceState
The source state to access. Pass one of the constants listed in "Event Source States" (page 541).

button
The mouse button to test. Pass one of the constants listed in "Mouse Buttons" (page 548).

Return Value
If `true`, the button is down. If `false`, the button is up.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEventSource.h

CGEventSourceCounterForEventType

Returns a count of events of a given type seen since the window server started.

```
uint32_t CGEventSourceCounterForEventType (
   CGEventSourceStateID source,
   CGEventType evType
);
```

Parameters

sourceState
The source state to access. Pass one of the constants listed in "Event Source States" (page 541).

eventType
The event type to access. To get the count of input events—keyboard, mouse, or tablet—specify `kCGAnyInputEventType`.

Return Value
The count of events of the specified type seen since the window server started.

Discussion
Quartz provides these counters for applications that monitor user activity. For example, an application could prompt a typist to take a break to reduce repetitive stress injuries.

Modifier keys produce `kCGEventFlagsChanged` events, not `kCGEventKeyDown` events, and do so both on press and release. The volume, brightness, and CD eject keys on some keyboards (both desktop and laptop) do not generate key up or key down events.

For various reasons, the number of key up and key down events may not be the same when all keyboard keys are up. As a result, a mismatch does not necessarily indicate that some keys are down.

Key autorepeat events are not counted.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEventSource.h

CGEventSourceCreate

Returns a Quartz event source created with a specified source state.

```
CGEventSourceRef CGEventSourceCreate (
   CGEventSourceStateID sourceState
);
```

Parameters

sourceState
The event state table to use for this event source. Pass one of the constants listed in "Event Source States" (page 541).

Return Value
A new event source, or `NULL` if the specified source state is not valid. When you no longer need the event source, you should release it using the function `CFRelease`.

Discussion
If two or more event sources are using the same source state and one of them is released, the remaining event sources will behave as if all keys and buttons on input devices are up in generating new events from this source.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventSource.h`

CGEventSourceFlagsState

Returns the current flags of a Quartz event source.

```
CGEventFlags CGEventSourceFlagsState (
   CGEventSourceStateID sourceState
);
```

Parameters

sourceState
The source state to access. Pass one of the constants listed in "Event Source States" (page 541).

Return Value
The current flags of the specified event source. For more information, see "Event Flags" (page 540).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventSource.h`

CGEventSourceGetKeyboardType

Returns the keyboard type to be used with a Quartz event source.

```
CGEventSourceKeyboardType CGEventSourceGetKeyboardType (
   CGEventSourceRef source
);
```

Parameters

source
The event source to access. Pass one of the constants listed in "Event Source States" (page 541).

Return Value
The keyboard type to be used with the specified event source.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEventSource.h

CGEventSourceGetLocalEventsFilterDuringSuppressionState

Returns the mask that indicates which classes of local hardware events are enabled during event suppression.

```
CGEventFilterMask CGEventSourceGetLocalEventsFilterDuringSuppressionState (
   CGEventSourceRef source,
   CGEventSuppressionState state
);
```

Parameters

source

The event source to access.

state

The type of event suppression interval during which the filter is applied. Pass one of the constants listed in "Event Suppression States" (page 543).

Return Value
A mask that specifies the categories of local hardware events to enable during the event suppression interval. See "Event Filter Masks" (page 540).

Discussion
You can configure the system to suppress local hardware events from the keyboard or mouse during a short interval after a Quartz event is posted or during a synthetic mouse drag (mouse movement with the left or only mouse button down). For information about setting this local events filter, see `CGEventSourceSetLocalEventsFilterDuringSuppressionState` (page 515).

This function lets you specify an event source and a suppression state (event suppression interval or mouse drag), and returns a filter mask of event categories to be passed through during suppression.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEventSource.h

CGEventSourceGetLocalEventsSuppressionInterval

Returns the interval that local hardware events may be suppressed following the posting of a Quartz event.

```
CFTimeInterval CGEventSourceGetLocalEventsSuppressionInterval (
   CGEventSourceRef source
);
```

Parameters

source

The event source to access.

Discussion
By default, the system does not suppress local hardware events from the keyboard or mouse during a short interval after a Quartz event is posted. You can use the function `CGEventSourceSetLocalEventsFilterDuringSuppressionState` (page 515) to modify this behavior.

This function gets the period of time in seconds that local hardware events may be suppressed after posting a Quartz event created with the specified event source. You can use the function `CGEventSourceSetLocalEventsSuppressionInterval` (page 516) to change this time interval.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventSource.h`

CGEventSourceGetPixelsPerLine

Gets the scale of pixels per line in a scrolling event source.

```
double CGEventSourceGetPixelsPerLine (
    CGEventSourceRef source
);
```

Parameters

source
The event source to access.

Return Value
The scale of pixels per line in a scrolling event.

Discussion
This function returns the scale of pixels per line in the specified event source. For example, if the scale in the event source is 10.5 pixels per line, this function would return 10.5. Every scrolling event can be interpreted to be scrolling by pixel or by line. By default, the scale is about ten pixels per line. You can alter the scale with the function `CGEventSourceSetPixelsPerLine`.

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

See Also
`CGEventSourceSetPixelsPerLine` (page 516)

Declared In
`CGEventSource.h`

CGEventSourceGetSourceStateID

Returns the source state associated with a Quartz event source.

```
CGEventSourceStateID CGEventSourceGetSourceStateID (
   CGEventSourceRef source
);
```

Parameters

source
The event source to access.

Return Value
The source state associated with the specified event source.

Discussion
This function returns the ID of the source state table associated with an event source.

For event sources created with the `kCGEventSourceStatePrivate` source state, this function returns the ID of the private source state table created for the event source. This unique ID may be passed to the `CGEventSourceCreate` function to create a second event source sharing the same state table. This may be useful, for example, in creating separate mouse and keyboard sources which share a common private state.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventSource.h`

CGEventSourceGetTypeID

Returns the type identifier for the opaque type `CGEventSourceRef`.

```
CFTypeID CGEventSourceGetTypeID (
   void
);
```

Return Value
The Core Foundation type identifier for the opaque type `CGEventSourceRef` (page 529).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventSource.h`

CGEventSourceGetUserData

Returns the 64-bit user-specified data for a Quartz event source.

```
int64_t CGEventSourceGetUserData (
   CGEventSourceRef source
);
```

Parameters

source
The event source to access.

Return Value
The user-specified data.

Discussion
Each input event includes 64 bits of user-specified data. This function gets the user-specified data for all events created by the specified event source. This data may also be obtained per event using the `CGEventGetIntegerValueField` (page 499) function.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventSource.h`

CGEventSourceKeyState

Returns a Boolean value indicating the current keyboard state of a Quartz event source.

```
bool CGEventSourceKeyState (
   CGEventSourceStateID sourceState,
   CGKeyCode key
);
```

Parameters

sourceState
The source state to access. Pass one of the constants listed in "Event Source States" (page 541).

key
The virtual key code to test.

Return Value
If `true`, the key is down. If `false`, the key is up.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventSource.h`

CGEventSourceSecondsSinceLastEventType

Returns the elapsed time since the last event for a Quartz event source.

```
CFTimeInterval CGEventSourceSecondsSinceLastEventType (
   CGEventSourceStateID source,
   CGEventType eventType
);
```

Parameters

source
The source state to access. Pass one of the constants listed in "Event Source States" (page 541).

eventType
The event type to access. To get the elapsed time since the previous input event—keyboard, mouse, or tablet—specify `kCGAnyInputEventType`.

Return Value
The time in seconds since the previous input event of the specified type.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventSource.h`

CGEventSourceSetKeyboardType

Sets the keyboard type to be used with a Quartz event source.

```
void CGEventSourceSetKeyboardType (
   CGEventSourceRef source,
   CGEventSourceKeyboardType keyboardType
);
```

Parameters

source
The event source to access.

keyboardType
The keyboard type to be used with the specified event source.

Availability

Available in Mac OS X v10.4 and later.

Declared In

CGEventSource.h

CGEventSourceSetLocalEventsFilterDuringSuppressionState

Sets the mask that indicates which classes of local hardware events are enabled during event suppression.

```
void CGEventSourceSetLocalEventsFilterDuringSuppressionState (
   CGEventSourceRef source,
   CGEventFilterMask filter,
   CGEventSuppressionState state
);
```

Parameters

source
The event source to access.

filter
A mask that specifies the categories of local hardware events to enable during the event suppression interval. See "Event Filter Masks" (page 540).

state
The type of event suppression interval during which the filter is applied. Pass one of the constants listed in "Event Suppression States" (page 543).

Discussion

By default, the system does not suppress local hardware events from the keyboard or mouse during a short interval after a Quartz event is posted—see `CGEventSourceSetLocalEventsSuppressionInterval` (page 516)—and during a synthetic mouse drag (mouse movement with the left or only mouse button down).

Some applications may want to disable events from some of the local hardware during this interval. For example, if you post mouse events only, you may wish to suppress local mouse events and permit local keyboard events to pass through. This function lets you specify an event source, a suppression state (event suppression interval or mouse drag), and a filter mask of event classes to be passed through. The new local events filter takes effect with the next Quartz event you post using this event source.

Availability

Available in Mac OS X v10.4 and later.

Declared In
CGEventSource.h

CGEventSourceSetLocalEventsSuppressionInterval

Sets the interval that local hardware events may be suppressed following the posting of a Quartz event.

```
void CGEventSourceSetLocalEventsSuppressionInterval (
   CGEventSourceRef source,
   CFTimeInterval seconds
);
```

Parameters

source
The event source to access.

seconds
The period of time in seconds that local hardware events (keyboard or mouse) are suppressed after posting a Quartz event created with the specified event source. The value should be a number in the range [0.0, 10.0].

Discussion
By default, the system does not suppress local hardware events from the keyboard or mouse during a short interval after a Quartz event is posted. You can use the function CGEventSourceSetLocalEventsFilterDuringSuppressionState (page 515) to modify this behavior.

This function sets the period of time in seconds that local hardware events may be suppressed after posting a Quartz event created with the specified event source. The default suppression interval is 0.25 seconds.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEventSource.h

CGEventSourceSetPixelsPerLine

Sets the scale of pixels per line in a scrolling event source.

```
void CGEventSourceSetPixelsPerLine (
    CGEventSourceRef source,
    double pixelsPerLine
);
```

Parameters

source
The event source to access.

pixelsPerLine
The scale of pixels per line in the specified event source.

Discussion
This function sets the scale of pixels per line in the specified event source. For example, if you pass the value 12.0 in the *pixelsPerLine* parameter, the scale of pixels per line in the event source would be changed to 12.0. Every scrolling event can be interpreted to be scrolling by pixel or by line. By default, the scale is about ten pixels per line. You can retrieve the scale with the function `CGEventSourceGetPixelsPerLine`.

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

See Also
`CGEventSourceGetPixelsPerLine` (page 512)

Declared In
`CGEventSource.h`

CGEventSourceSetUserData

Sets the 64-bit user-specified data for a Quartz event source.

```
void CGEventSourceSetUserData (
   CGEventSourceRef source,
   int64_t userData
);
```

Parameters

source
The event source to access.

userData
The user-specified data. For example, you could specify a vendor hardware ID.

Discussion
Each input event includes 64 bits of user-specified data. This function sets the user-specified data for all events created by the specified event source. This data may also be set per event using the `CGEventGetIntegerValueField` (page 499) function.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventSource.h`

CGEventTapCreate

Creates an event tap.

```
CFMachPortRef CGEventTapCreate (
   CGEventTapLocation tap,
   CGEventTapPlacement place,
   CGEventTapOptions options,
   CGEventMask eventsOfInterest,
   CGEventTapCallBack callback,
   void *refcon
);
```

Parameters

tap

The location of the new event tap. Pass one of the constants listed in "Event Tap Locations" (page 543). Only processes running as the root user may locate an event tap at the point where HID events enter the window server; for other users, this function returns `NULL`.

place

The placement of the new event tap in the list of active event taps. Pass one of the constants listed in "Event Tap Placement" (page 544).

options

A constant that specifies whether the new event tap is a passive listener or an active filter.

eventsOfInterest

A bit mask that specifies the set of events to be observed. For a list of possible events, see "Event Types" (page 545). For information on how to specify the mask, see `CGEventMask` (page 528). If the event tap is not permitted to monitor one or more of the events specified in the `eventsOfInterest` parameter, then the appropriate bits in the mask are cleared. If that action results in an empty mask, this function returns `NULL`.

callback

An event tap callback function that you provide. Your callback function is invoked from the run loop to which the event tap is added as a source. The thread safety of the callback is defined by the run loop's environment. To learn more about event tap callbacks, see `CGEventTapCallBack` (page 526).

refcon

A pointer to user-defined data. This pointer is passed into the callback function specified in the `callback` parameter.

Return Value

A Core Foundation mach port that represents the new event tap, or `NULL` if the event tap could not be created. When you are finished using the event tap, you should release the mach port using the function `CFRelease`. Releasing the mach port also releases the tap.

Discussion

Event taps receive key up and key down events if one of the following conditions is true:

- The current process is running as the root user.
- Access for assistive devices is enabled. In Mac OS X v10.4, you can enable this feature using System Preferences, Universal Access panel, Keyboard view.

After creating an event tap, you can add it to a run loop as follows:

1. Pass the event tap to the `CFMachPortCreateRunLoopSource` function to create a run loop event source.
2. Call the `CFRunLoopAddSource` function to add the source to the appropriate run loop.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGEventTapCreateForPSN

Creates an event tap for a specified process.

```
CFMachPortRef CGEventTapCreateForPSN (
   void *processSerialNumber,
   CGEventTapPlacement place,
   CGEventTapOptions options,
   CGEventMask eventsOfInterest,
   CGEventTapCallBack callback,
   void *refcon
);
```

Parameters

processSerialNumber
The process to monitor.

place
The placement of the new event tap in the list of active event taps. Pass one of the constants listed in "Event Tap Placement" (page 544).

options
A constant that specifies whether the new event tap is a passive listener or an active filter.

eventsOfInterest
A bit mask that specifies the set of events to be observed. For a list of possible events, see "Event Types" (page 545). For information on how to specify the mask, see `CGEventMask` (page 528). If the event tap is not permitted to monitor one or more of the events specified in the `eventsOfInterest` parameter, then the appropriate bits in the mask are cleared. If that action results in an empty mask, this function returns `NULL`.

callback
An event tap callback function that you provide. Your callback function is invoked from the run loop to which the event tap is added as a source. The thread safety of the callback is defined by the run loop's environment. To learn more about event tap callbacks, see `CGEventTapCallBack` (page 526).

refcon
A pointer to user-defined data. This pointer is passed into the callback function specified in the `callback` parameter.

Return Value
A Core Foundation mach port that represents the new event tap, or `NULL` if the event tap could not be created. When you are finished using the event tap, you should release the mach port using the function `CFRelease`. Releasing the mach port also releases the tap.

Discussion
This function creates an event tap that receives events being routed by the window server to the specified process. For more information about creating event taps, see `CGEventTapCreate` (page 517).

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEvent.h

CGEventTapEnable

Enables or disables an event tap.

```
void CGEventTapEnable (
   CFMachPortRef myTap,
   bool enable
);
```

Parameters

myTap

The event tap to enable or disable.

enable

Pass `true` to enable the event tap. To disable it, pass `false`.

Discussion
Event taps are normally enabled when created. If an event tap becomes unresponsive, or if a user requests that event taps be disabled, then a `kCGEventTapDisabled` event is passed to the event tap callback function. Event taps may be re-enabled by calling this function.

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEvent.h

CGEventTapIsEnabled

Returns a Boolean value indicating whether an event tap is enabled.

```
bool CGEventTapIsEnabled (
   CFMachPortRef myTap
);
```

Parameters

myTap

The event tap to test.

Return Value
If `true`, the specified event tap is enabled; otherwise, `false`.

Discussion
For more information, see the function `CGEventTapEnable` (page 520).

Availability
Available in Mac OS X v10.4 and later.

Declared In
CGEvent.h

CGEventTapPostEvent

Posts a Quartz event from an event tap into the event stream.

```
void CGEventTapPostEvent (
   CGEventTapProxy proxy,
   CGEventRef event
);
```

Parameters

proxy

A proxy that identifies the event tap posting the event. Your event tap callback function is passed this proxy when it is invoked.

event

The event to post.

Discussion

You can use this function to post a new event at the same point to which an event returned from an event tap callback function would be posted. The new event enters the system before the event returned by the callback enters the system. Events posted into the system will be seen by all taps placed after the tap posting the event.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGEvent.h`

CGGetEventTapList

Gets a list of currently installed event taps.

```
CGError CGGetEventTapList (
   CGTableCount maxNumberOfTaps,
   CGEventTapInformation tapList[],
   CGTableCount *eventTapCount
);
```

Parameters

maxNumberOfTaps

The length of the array you provide in the `tapList` parameter.

tapList

An array of event tap information structures. You are responsible for allocating storage for the array. On return, your array contains a list of currently installed event taps. If you pass `NULL` in this parameter, the `maxNumberOfTaps` parameter is ignored, and the `eventTapCount` variable is filled in with the number of event taps that are currently installed.

eventTapCount

A pointer to a `CGTableCount` variable. On return, the variable contains actual number of array elements filled in.

Return Value

A result code. See the result codes described in *Quartz Display Services Reference*.

Discussion
Each call to this function has the side effect of resetting the minimum and maximum latency values in the `tapList` parameter to the corresponding average values. Values reported in these fields reflect the minimum and maximum values seen since the preceding call, or the instantiation of the tap. This allows a monitoring tool to evaluate the best and worst case latency over time and under various operating conditions.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEvent.h`

CGInhibitLocalEvents

Turns off local hardware events in the current session.

```
CGError CGInhibitLocalEvents (
   boolean_t doInhibit
);
```

Parameters

`doInhibit`
Pass `true` to specify that local hardware events on the remote system should be inhibited; otherwise, pass `false`.

Return Value
A result code. See the result codes described in *Quartz Display Services Reference*.

Discussion
This function is typically used during remote operation of a system to disconnect the keyboard and mouse for a short period of time, as in automated system testing or telecommuting applications.

The `CGInhibitLocalEvents` function is not recommended for general use because of undocumented special cases and undesirable side effects. For example, this function can permanently disable the keyboard and mouse, rendering the system unusable. The recommended replacement for this function is `CGEventSourceSetLocalEventsFilterDuringSuppressionState` (page 515).

Special Considerations

In Mac OS X v10.2 and earlier, this function inhibits local events only after a synthetic keyboard or mouse event is posted by the calling application. In Mac OS X v10.3 and later, event inhibition takes effect immediately. If your application terminates for any reason, event inhibition on the remote system is immediately turned off.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGRemoteOperation.h`

CGPostKeyboardEvent

Synthesizes a low-level keyboard event on the local machine.

```
CGError CGPostKeyboardEvent (
   CGCharCode keyChar,
   CGKeyCode virtualKey,
   boolean_t keyDown
);
```

Parameters

keyChar

The value of the character to generate, or 0 to specify that the system should guess an appropriate value based on the default key mapping.

virtualKey

The virtual key code for the event. See `CGKeyCode` (page 531).

keyDown

Pass `true` to specify that the key position is down; otherwise, pass `false`.

Return Value

A result code. See the result codes described in *Quartz Display Services Reference*.

Discussion

This function is not recommended for general use because of undocumented special cases and undesirable side effects. The recommended replacement for this function is `CGEventCreateKeyboardEvent` (page 495), which allows you to create a keyboard event and customize the event before posting it to the event system.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGRemoteOperation.h`

CGPostMouseEvent

Synthesizes a low-level mouse-button event on the local machine.

```
CGError CGPostMouseEvent (
   CGPoint mouseCursorPosition,
   boolean_t updateMouseCursorPosition,
   CGButtonCount buttonCount,
   boolean_t mouseButtonDown,
   ...
);
```

Parameters

mouseCursorPosition

The new coordinates of the mouse in global display space.

updateMouseCursorPosition

Pass `true` if the on-screen cursor should be moved to the location specified in the `mouseCursorPosition` parameter; otherwise, pass `false`.

buttonCount

The number of mouse buttons, up to a maximum of 32.

mouseButtonDown

Pass `true` to specify that the primary or left mouse button is down; otherwise, pass `false`.

`...`
: Zero or more Boolean values that specify whether the remaining mouse buttons are down (`true`) or up (`false`). The second value, if any, should specify the state of the secondary mouse button (right). A third value would specify the state of the center button, and the remaining buttons would be in USB device order.

Return Value
A result code. See the result codes described in *Quartz Display Services Reference*.

Discussion
Based on the arguments you pass to this function, the function generates the appropriate mouse-down, mouse-up, mouse-move, or mouse-drag events by comparing the new state with the current state.

This function is not recommended for general use because of undocumented special cases and undesirable side effects. The recommended replacement for this function is `CGEventCreateMouseEvent` (page 496), which allows you to create a mouse event and customize the event before posting it to the event system.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGRemoteOperation.h`

CGPostScrollWheelEvent

Synthesizes a low-level scrolling event on the local machine.

```
CGError CGPostScrollWheelEvent (
   CGWheelCount wheelCount,
   int32_t wheel1,
   ...
);
```

Parameters

`wheelCount`
: The number of scrolling devices, up to a maximum of 3.

`wheel1`
: A value that reflects the movement of the primary scrolling device on the mouse.

`...`
: Up to two values that reflect the movements of the other scrolling devices on the mouse (if any).

Return Value
A result code. See the result codes described in *Quartz Display Services Reference*.

Discussion
Scrolling movement is generally represented by small signed integer values, typically in a range from -10 to +10. Large values may have unexpected results, depending on the application that processes the event.

This function is not recommended for general use because of undocumented special cases and undesirable side effects. The recommended replacement for this function is `CGEventCreateScrollWheelEvent` (page 497), which allows you to create a scrolling event and customize the event before posting it to the event system.

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGRemoteOperation.h

CGSetLocalEventsFilterDuringSuppressionState

Filters local hardware events from the keyboard and mouse during the short interval after a synthetic event is posted.

```
CGError CGSetLocalEventsFilterDuringSuppressionState (
   CGEventFilterMask filter,
   CGEventSuppressionState state
);
```

Parameters

filter

The class of local hardware events to enable after a synthetic event is posted. Pass one of the constants listed in "Event Filter Masks" (page 540).

state

The type of interval during which the filter is applied. Pass one of the constants listed in "Event Suppression States" (page 543).

Return Value
A result code. See the result codes described in *Quartz Display Services Reference*.

Discussion
By default, the system suppresses local hardware events from the keyboard and mouse during a short interval after a synthetic event is posted and during a synthetic mouse drag (mouse movement with the left or only mouse button down).

Some applications may want to enable events from some of the local hardware. For example, if you post mouse events only, you may wish to permit local keyboard hardware events to pass through.

This function lets you specify a state (event suppression interval or mouse drag), and a mask of event categories to be passed through. The new filter state takes effect with the next synthetic event you post.

This function is not recommended for general use because of undocumented special cases and undesirable side effects. The recommended replacement for this function is `CGEventSourceSetLocalEventsFilterDuringSuppressionState` (page 515), which allows the filter behavior to be associated only with events created from a specific event source.

Availability
Available in Mac OS X v10.3 and later.

Declared In
CGRemoteOperation.h

CGSetLocalEventsSuppressionInterval

Sets the time interval in seconds that local hardware events are suppressed after posting a synthetic event.

```
CGError CGSetLocalEventsSuppressionInterval (
   CFTimeInterval seconds
);
```

Parameters

seconds

The desired time interval in seconds. The value should be a number in the range [0.0, 10.0].

Return Value

A result code. If the `seconds` parameter is outside the allowed range, returns `kCGErrorRangeCheck`.

Discussion

This function determines how long local events matching an event filter are to be suppressed following the posting of a synthetic event. The default time interval for event suppression is 0.25 seconds.

This function is not recommended for general use because of undocumented special cases and undesirable side effects. The recommended replacement for this function is `CGEventSourceSetLocalEventsSuppressionInterval` (page 516), which allows the suppression interval to be adjusted for a specific event source, affecting only events posted using that event source.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGRemoteOperation.h`

Callbacks

CGEventTapCallBack

A client-supplied callback function that's invoked whenever an associated event tap receives a Quartz event.

```
typedef CGEventRef (*CGEventTapCallBack) (
   CGEventTapProxy proxy,
   CGEventType type,
   CGEventRef event,
   void *refcon
);
```

If you name your function `MyEventTapCallBack`, you would declare it like this:

```
CGEventRef MyEventTapCallBack (
   CGEventTapProxy proxy,
   CGEventType type,
   CGEventRef event,
   void *refcon
);
```

Parameters

proxy

A proxy for the event tap. See `CGEventTapProxy` (page 531). This callback function may pass this proxy to other functions such as the event-posting routines.

type
The event type of this event. See "Event Types" (page 545).

event
The incoming event. This event is owned by the caller, and you do not need to release it.

refcon
A pointer to user-defined data. You specify this pointer when you create the event tap. Several different event taps could use the same callback function, each tap with its own user-defined data.

Discussion
If the event tap is an active filter, your callback function should return one of the following:

- The (possibly modified) event that is passed in. This event is passed back to the event system.
- A newly-constructed event. After the new event has been passed back to the event system, the new event will be released along with the original event.
- `NULL` if the event passed in is to be deleted.

If the event tap is an passive listener, your callback function may return the event that is passed in, or `NULL`. In either case, the event stream is not affected.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventTypes.h`

Data Types

CGButtonCount

Represents the number of buttons being set in a synthetic mouse event.

```
typedef uint32_t CGButtonCount;
```

Discussion
In mouse events, the button count parameter ranges from 0 to 31. See the function `CGPostMouseEvent` (page 523).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGRemoteOperation.h`

CGCharCode

Represents a character generated by pressing one or more keys on a keyboard.

```
typedef uint16_t CGCharCode;
```

Discussion
This data type represents a 16-bit character code. Values of this type may or may not correspond to UTF-16 character codes. See the function `CGPostKeyboardEvent` (page 522).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGRemoteOperation.h`

CGEventMask

Defines a mask that identifies the set of Quartz events to be observed in an event tap.

```
typedef uint64_t CGEventMask;
```

Discussion
When you call either `CGEventTapCreate` (page 517) or `CGEventTapCreateForPSN` (page 519) to register an event tap, you supply a bit mask that identifies the set of events to be observed. You specify each event using one of the event type constants listed in "Event Types" (page 545). To form the bit mask, use the `CGEventMaskBit` macro to convert each constant into an event mask and then OR the individual masks together. For example:

```
CGEventMask mask = CGEventMaskBit(kCGEventLeftMouseDown) |
                   CGEventMaskBit(kCGEventLeftMouseUp);
```

You can also supply a mask to observe all events:

```
CGEventMask mask = kCGEventMaskForAllEvents;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventTypes.h`

CGEventRef

Defines an opaque type that represents a low-level hardware event.

```
typedef struct __CGEvent *CGEventRef;
```

Discussion
Low-level hardware events of this type are referred to as Quartz events. A typical event in Mac OS X originates when the user manipulates an input device such as a mouse or a keyboard. The device driver associated with that device, through the I/O Kit, creates a low-level event, puts it in the window server's event queue, and notifies the window server. The window server creates a Quartz event, annotates the event, and dispatches the event to the appropriate run-loop port of the target process. There the event is picked up by the Carbon Event Manager and forwarded to the event-handling mechanism appropriate to the application environment. You can use event taps to gain access to Quartz events at several different steps in this process.

This opaque type is derived from CFType and inherits the properties that all Core Foundation types have in common. For more information, see *CFType Reference*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventTypes.h`

CGEventSourceKeyboardType

Defines a code that represents the type of keyboard used with a specified event source.

```
typedef uint32_t CGEventSourceKeyboardType;
```

Discussion
This code is the same keyboard type identifier used with the `UCKeyTranslate` function to drive keyboard translation.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventTypes.h`

CGEventSourceRef

Defines an opaque type that represents the source of a Quartz event.

```
typedef struct __CGEventSource * CGEventSourceRef;
```

Discussion
A Quartz event source is an object that contains accumulated state related to event generation and event posting. Every event source has an associated global event state table called a source state. When you call `CGEventSourceCreate` (page 509) to create an event source, you specify which source state to use. For more information about source states, see "Event Source States" (page 541).

A typical use of an event source would be to obtain the source from a Quartz event received by an event tap callback function, and then to use that source for any new events created as a result of the received event. This has the effect of marking the events as being related.

This opaque type is derived from CFType and inherits the properties that all Core Foundation types have in common. For more information, see *CFType Reference*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventTypes.h`

CGEventTapInformation

Defines the structure used to report information about event taps.

```
typedef struct CGEventTapInformation
{
    uint32_t            eventTapID;
    CGEventTapLocation  tapPoint;
    CGEventTapOptions   options;
    CGEventMask         eventsOfInterest;
    pid_t               tappingProcess;
    pid_t               processBeingTapped;
    bool                enabled;
    float               minUsecLatency;
    float               avgUsecLatency;
    float               maxUsecLatency;
} CGEventTapInformation;
```

Fields

`eventTapID`
The unique identifier for the event tap.

`tapPoint`
The location of the event tap. See "Event Tap Locations" (page 543).

`options`
The type of event tap (passive listener or active filter).

`eventsOfInterest`
The mask that identifies the set of events to be observed.

`tappingProcess`
The process ID of the application that created the event tap.

`processBeingTapped`
The process ID of the target application (non-zero only if the event tap was created using the function `CGEventTapCreateForPSN` (page 519).

`enabled`
`TRUE` if the event tap is currently enabled; otherwise `FALSE`.

`minUsecLatency`
Minimum latency in microseconds. In this data structure, **latency** is defined as the time in microseconds it takes for an event tap to process and respond to an event passed to it.

`avgUsecLatency`
Average latency in microseconds. This is a weighted average that gives greater weight to more recent events.

`maxUsecLatency`
Maximum latency in microseconds.

Discussion

To learn how to obtain information about event taps, see the function `CGGetEventTapList` (page 521).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGEventTypes.h`

CGEventTapProxy

Defines an opaque type that represents state within the client application that's associated with an event tap.

```
typedef struct __CGEventTapProxy * CGEventTapProxy;
```

Discussion
An event tap proxy object is passed to your event tap callback function when it receives a new Quartz event. Your callback function needs the proxy to post Quartz events using the function `CGEventTapPostEvent` (page 521).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventTypes.h`

CGEventTimestamp

Defines the elapsed time in nanoseconds since startup that a Quartz event occurred.

```
typedef uint64_t CGEventTimestamp;
```

Discussion
An event timestamp is a big, unsigned, 64-bit number. That's big, really big. You just won't believe how vastly, hugely, mind-bogglingly big it is. You may think your application has been running for a long time, but that's just peanuts to an event timestamp.

For information about how event timestamps are used, see the functions `CGEventGetTimestamp` (page 500) and `CGEventSetTimestamp` (page 507).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGEventTypes.h`

CGKeyCode

Represents the virtual key codes used in keyboard events.

```
typedef uint16_t CGKeyCode;
```

Discussion
In Mac OS X, the hardware scan codes generated by keyboards are mapped to a set of virtual key codes that are hardware-independent. Pressing a given key always generates the same virtual key code on any supported keyboard.

As keys are pressed, the system uses the virtual key codes to create low-level keyboard events. For information on how to simulate a keyboard event, see the function `CGEventCreateKeyboardEvent` (page 495).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGRemoteOperation.h

CGWheelCount

Represents the number of wheels being set in a scroll wheel event.

```
typedef uint32_t CGWheelCount;
```

Discussion
See the function CGPostScrollWheelEvent (page 524).

Availability
Available in Mac OS X v10.0 and later.

Declared In
CGRemoteOperation.h

Constants

Event Fields

Constants used as keys to access specialized fields in low-level events.

```
enum _CGEventField {
    kCGMouseEventNumber = 0,
    kCGMouseEventClickState = 1,
    kCGMouseEventPressure = 2,
    kCGMouseEventButtonNumber = 3,
    kCGMouseEventDeltaX = 4,
    kCGMouseEventDeltaY = 5,
    kCGMouseEventInstantMouser = 6,
    kCGMouseEventSubtype = 7,
    kCGKeyboardEventAutorepeat = 8,
    kCGKeyboardEventKeycode = 9,
    kCGKeyboardEventKeyboardType = 10,
    kCGScrollWheelEventDeltaAxis1 = 11,
    kCGScrollWheelEventDeltaAxis2 = 12,
    kCGScrollWheelEventDeltaAxis3 = 13,
    kCGScrollWheelEventFixedPtDeltaAxis1 = 93,
    kCGScrollWheelEventFixedPtDeltaAxis2 = 94,
    kCGScrollWheelEventFixedPtDeltaAxis3 = 95,
    kCGScrollWheelEventPointDeltaAxis1 = 96,
    kCGScrollWheelEventPointDeltaAxis2 = 97,
    kCGScrollWheelEventPointDeltaAxis3 = 98,
    kCGScrollWheelEventInstantMouser = 14,
    kCGTabletEventPointX = 15,
    kCGTabletEventPointY = 16,
    kCGTabletEventPointZ = 17,
    kCGTabletEventPointButtons = 18,
    kCGTabletEventPointPressure = 19,
    kCGTabletEventTiltX = 20,
    kCGTabletEventTiltY = 21,
    kCGTabletEventRotation = 22,
    kCGTabletEventTangentialPressure = 23,
    kCGTabletEventDeviceID = 24,
    kCGTabletEventVendor1 = 25,
    kCGTabletEventVendor2 = 26,
    kCGTabletEventVendor3 = 27,
    kCGTabletProximityEventVendorID = 28,
    kCGTabletProximityEventTabletID = 29,
    kCGTabletProximityEventPointerID = 30,
    kCGTabletProximityEventDeviceID = 31,
    kCGTabletProximityEventSystemTabletID = 32,
    kCGTabletProximityEventVendorPointerType = 33,
    kCGTabletProximityEventVendorPointerSerialNumber = 34,
    kCGTabletProximityEventVendorUniqueID = 35,
    kCGTabletProximityEventCapabilityMask = 36,
    kCGTabletProximityEventPointerType = 37,
    kCGTabletProximityEventEnterProximity = 38,
    kCGEventTargetProcessSerialNumber = 39,
    kCGEventTargetUnixProcessID = 40,
    kCGEventSourceUnixProcessID = 41,
    kCGEventSourceUserData = 42,
    kCGEventSourceUserID = 43,
    kCGEventSourceGroupID = 44,
    kCGEventSourceStateID = 45,
    kCGScrollWheelEventIsContinuous = 88
};
typedef uint32_t CGEventField;
```

Constants

`kCGMouseEventNumber`
Key to access an integer field that contains the mouse button event number. Matching mouse-down and mouse-up events will have the same event number.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGMouseEventClickState`
Key to access an integer field that contains the mouse button click state. A click state of 1 represents a single click. A click state of 2 represents a double-click. A click state of 3 represents a triple-click.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGMouseEventPressure`
Key to access a double field that contains the mouse button pressure. The pressure value may range from 0 to 1, with 0 representing the mouse being up. This value is commonly set by tablet pens mimicking a mouse.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGMouseEventButtonNumber`
Key to access an integer field that contains the mouse button number. For information about the possible values, see "Mouse Buttons" (page 548).

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGMouseEventDeltaX`
Key to access an integer field that contains the horizontal mouse delta since the last mouse movement event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGMouseEventDeltaY`
Key to access an integer field that contains the vertical mouse delta since the last mouse movement event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGMouseEventInstantMouser`
Key to access an integer field. The value is non-zero if the event should be ignored by the Inkwell subsystem.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGMouseEventSubtype`
Key to access an integer field that encodes the mouse event subtype as a `kCFNumberIntType`.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGKeyboardEventAutorepeat`
Key to access an integer field, non-zero when this is an autorepeat of a key-down, and zero otherwise.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGKeyboardEventKeycode`
: Key to access an integer field that contains the virtual keycode of the key-down or key-up event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGKeyboardEventKeyboardType`
: Key to access an integer field that contains the keyboard type identifier.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventDeltaAxis1`
: Key to access an integer field that contains scrolling data. This field typically contains the change in vertical position since the last scrolling event from a Mighty Mouse scroller or a single-wheel mouse scroller.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventDeltaAxis2`
: Key to access an integer field that contains scrolling data. This field typically contains the change in horizontal position since the last scrolling event from a Mighty Mouse scroller.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventDeltaAxis3`
: This field is not used.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventFixedPtDeltaAxis1`
: Key to access a field that contains scrolling data. The scrolling data represents a line-based or pixel-based change in vertical position since the last scrolling event from a Mighty Mouse scroller or a single-wheel mouse scroller. The scrolling data uses a fixed-point 16.16 signed integer format. For example, if the field contains a value of 1.0, the integer 0x00010000 is returned by `CGEventGetIntegerValueField`. If this key is passed to `CGEventGetDoubleValueField`, the fixed-point value is converted to a double value.

Available in Mac OS X v10.5 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventFixedPtDeltaAxis2`
: Key to access a field that contains scrolling data. The scrolling data represents a line-based or pixel-based change in horizontal position since the last scrolling event from a Mighty Mouse scroller. The scrolling data uses a fixed-point 16.16 signed integer format. For example, if the field contains a value of 1.0, the integer 0x00010000 is returned by `CGEventGetIntegerValueField`. If this key is passed to `CGEventGetDoubleValueField`, the fixed-point value is converted to a double value.

Available in Mac OS X v10.5 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventFixedPtDeltaAxis3`
: This field is not used.

Available in Mac OS X v10.5 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventPointDeltaAxis1`
Key to access an integer field that contains pixel-based scrolling data. The scrolling data represents the change in vertical position since the last scrolling event from a Mighty Mouse scroller or a single-wheel mouse scroller.

Available in Mac OS X v10.5 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventPointDeltaAxis2`
Key to access an integer field that contains pixel-based scrolling data. The scrolling data represents the change in horizontal position since the last scrolling event from a Mighty Mouse scroller.

Available in Mac OS X v10.5 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventPointDeltaAxis3`
This field is not used.

Available in Mac OS X v10.5 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventInstantMouser`
Key to access an integer field that indicates whether the event should be ignored by the Inkwell subsystem. If the value is non-zero, the event should be ignored.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventPointX`
Key to access an integer field that contains the absolute X coordinate in tablet space at full tablet resolution.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventPointY`
Key to access an integer field that contains the absolute Y coordinate in tablet space at full tablet resolution.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventPointZ`
Key to access an integer field that contains the absolute Z coordinate in tablet space at full tablet resolution.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventPointButtons`
Key to access an integer field that contains the tablet button state. Bit 0 is the first button, and a set bit represents a closed or pressed button. Up to 16 buttons are supported.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventPointPressure`
Key to access a double field that contains the tablet pen pressure. A value of 0.0 represents no pressure, and 1.0 represents maximum pressure.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventTiltX`
Key to access a double field that contains the horizontal tablet pen tilt. A value of 0.0 represents no tilt, and 1.0 represents maximum tilt.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventTiltY`
Key to access a double field that contains the vertical tablet pen tilt. A value of 0.0 represents no tilt, and 1.0 represents maximum tilt.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventRotation`
Key to access a double field that contains the tablet pen rotation.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventTangentialPressure`
Key to access a double field that contains the tangential pressure on the device. A value of 0.0 represents no pressure, and 1.0 represents maximum pressure.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventDeviceID`
Key to access an integer field that contains the system-assigned unique device ID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventVendor1`
Key to access an integer field that contains a vendor-specified value.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventVendor2`
Key to access an integer field that contains a vendor-specified value.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletEventVendor3`
Key to access an integer field that contains a vendor-specified value.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventVendorID`
Key to access an integer field that contains the vendor-defined ID, typically the USB vendor ID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventTabletID`
Key to access an integer field that contains the vendor-defined tablet ID, typically the USB product ID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventPointerID`
Key to access an integer field that contains the vendor-defined ID of the pointing device.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventDeviceID`
Key to access an integer field that contains the system-assigned device ID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventSystemTabletID`
Key to access an integer field that contains the system-assigned unique tablet ID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventVendorPointerType`
Key to access an integer field that contains the vendor-assigned pointer type.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventVendorPointerSerialNumber`
Key to access an integer field that contains the vendor-defined pointer serial number.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventVendorUniqueID`
Key to access an integer field that contains the vendor-defined unique ID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventCapabilityMask`
Key to access an integer field that contains the device capabilities mask.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventPointerType`
Key to access an integer field that contains the pointer type.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTabletProximityEventEnterProximity`
Key to access an integer field that indicates whether the pen is in proximity to the tablet. The value is non-zero if the pen is in proximity to the tablet and zero when leaving the tablet.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventTargetProcessSerialNumber`
Key to access a field that contains the event target process serial number. The value is a 64-bit long word.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventTargetUnixProcessID`
Key to access a field that contains the event target Unix process ID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventSourceUnixProcessID`
Key to access a field that contains the event source Unix process ID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventSourceUserData`
Key to access a field that contains the event source user-supplied data, up to 64 bits.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventSourceUserID`
Key to access a field that contains the event source Unix effective UID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventSourceGroupID`
Key to access a field that contains the event source Unix effective GID.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventSourceStateID`
Key to access a field that contains the event source state ID used to create this event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGScrollWheelEventIsContinuous`
Key to access an integer field that indicates whether a scrolling event contains continuous, pixel-based scrolling data. The value is non-zero when the scrolling data is pixel-based and zero when the scrolling data is line-based.

Available in Mac OS X v10.5 and later.

Declared in `CGEventTypes.h`.

Discussion
These constants are used as keys to access certain specialized event fields when using low-level accessor functions such as `CGEventGetIntegerValueField` (page 499), `CGEventSetIntegerValueField` (page 506), `CGEventGetDoubleValueField` (page 498), and `CGEventSetDoubleValueField` (page 505).

Event Filter Masks

Specify masks for classes of low-level events that can be filtered during event suppression states.

```
enum CGEventFilterMask {
    kCGEventFilterMaskPermitLocalMouseEvents = 0x00000001,
    kCGEventFilterMaskPermitLocalKeyboardEvents = 0x00000002,
    kCGEventFilterMaskPermitSystemDefinedEvents = 0x00000004,
    kCGEventFilterMaskPermitAllEvents = kCGEventFilterMaskPermitLocalMouseEvents
 | kCGEventFilterMaskPermitLocalKeyboardEvents |
kCGEventFilterMaskPermitSystemDefinedEvents
};
typedef uint32_t CGEventFilterMask;
```

Event Flags

Constants that indicate the modifier key state at the time an event is created, as well as other event-related states.

```
enum _CGEventFlags {
    kCGEventFlagMaskAlphaShift =    NX_ALPHASHIFTMASK,
    kCGEventFlagMaskShift =         NX_SHIFTMASK,
    kCGEventFlagMaskControl =       NX_CONTROLMASK,
    kCGEventFlagMaskAlternate =     NX_ALTERNATEMASK,
    kCGEventFlagMaskCommand =       NX_COMMANDMASK,
    kCGEventFlagMaskHelp =          NX_HELPMASK,
    kCGEventFlagMaskSecondaryFn =   NX_SECONDARYFNMASK,
    kCGEventFlagMaskNumericPad =    NX_NUMERICPADMASK,
    kCGEventFlagMaskNonCoalesced =  NX_NONCOALSESCEDMASK
};
typedef uint64_t CGEventFlags;
```

Constants

`kCGEventFlagMaskAlphaShift`

Indicates that the Caps Lock key is down for a keyboard, mouse, or flag-changed event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventFlagMaskShift`

Indicates that the Shift key is down for a keyboard, mouse, or flag-changed event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventFlagMaskControl`

Indicates that the Control key is down for a keyboard, mouse, or flag-changed event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventFlagMaskAlternate`

Indicates that the Alt or Option key is down for a keyboard, mouse, or flag-changed event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventFlagMaskCommand`
Indicates that the Command key is down for a keyboard, mouse, or flag-changed event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventFlagMaskHelp`
Indicates that the Help modifier key is down for a keyboard, mouse, or flag-changed event. This key is not present on most keyboards, and is different than the Help key found in the same row as Home and Page Up.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventFlagMaskSecondaryFn`
Indicates that the Fn (Function) key is down for a keyboard, mouse, or flag-changed event. This key is found primarily on laptop keyboards.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventFlagMaskNumericPad`
Identifies key events from the numeric keypad area on extended keyboards.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventFlagMaskNonCoalesced`
Indicates that mouse and pen movement events are not being coalesced.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

Discussion
These constants specify masks for the bits in an event flags bit mask. Event flags indicate the modifier key state at the time an event is created, as well as other event-related states. Event flags are used in accessor functions such as `CGEventGetFlags` (page 499), `CGEventSetFlags` (page 505), and `CGEventSourceFlagsState` (page 510).

Event Source States

Constants that specify the possible source states of an event source.

```
enum {
    kCGEventSourceStatePrivate = -1,
    kCGEventSourceStateCombinedSessionState = 0,
    kCGEventSourceStateHIDSystemState = 1
};
typedef uint32_t CGEventSourceStateID;
```

Constants
`kCGEventSourceStatePrivate`
Specifies that an event source should use a private event state table.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventSourceStateCombinedSessionState`
Specifies that an event source should use the event state table that reflects the combined state of all event sources posting to the current user login session.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventSourceStateHIDSystemState`
Specifies that an event source should use the event state table that reflects the combined state of all hardware event sources posting from the HID system.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

Discussion
A source state refers to a global event state table. These tables contain accumulated information on modifier flag state, keyboard key state, mouse button state, and related internal parameters placed in effect by posting events with associated sources.

Two pre-existing event state tables are defined:

- The `kCGEventSourceStateCombinedSessionState` table reflects the combined state of all event sources posting to the current user login session. If your program is posting events from within a login session, you should use this source state when you create an event source.
- The `kCGEventSourceStateHIDSystemState` table reflects the combined state of all hardware event sources posting from the HID system. If your program is a daemon or a user space device driver interpreting hardware state and generating events, you should use this source state when you create an event source.

Specialized applications such as remote control programs may want to generate and track event source state independent of other processes. These programs should use the `kCGEventSourceStatePrivate` value in creating their event source. An independent state table and unique source state ID (`CGEventSourceStateID`) are created to track the event source's state. This independent state table is owned by the creating event source and released with it.

Event Source Token

Specifies any input event type.

```
#define kCGAnyInputEventType ((CGEventType)(~0))
```

Constants
`kCGAnyInputEventType`
A constant that specifies any input event type.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

Discussion
This constant is typically used with the function `CGEventSourceSecondsSinceLastEventType` (page 514) to specify that you want the elapsed time since the last input event of any type.

Event Suppression States

Specify the event suppression states that can occur after posting an event.

```
enum CGEventSuppressionState {
    kCGEventSuppressionStateSuppressionInterval = 0,
    kCGEventSuppressionStateRemoteMouseDrag = 1,
    kCGNumberOfEventSuppressionStates = 2
};
typedef uint32_t CGEventSuppressionState;
```

Constants

`kCGEventSuppressionStateSuppressionInterval`

Specifies that certain local hardware events may be suppressed for a short interval after posting an event.

Available in Mac OS X v10.3 and later.

Declared in `CGRemoteOperation.h`.

`kCGEventSuppressionStateRemoteMouseDrag`

Specifies that certain local hardware events may be suppressed during a mouse drag operation (mouse movement with the left or only mouse button down).

Available in Mac OS X v10.3 and later.

Declared in `CGRemoteOperation.h`.

Discussion

These constants specify the types of event suppression intervals during which an event filter is applied after posting an event.

Event Tap Locations

Constants that specify possible tapping points for events.

```
enum _CGEventTapLocation {
    kCGHIDEventTap = 0,
    kCGSessionEventTap,
    kCGAnnotatedSessionEventTap
};
typedef uint32_t CGEventTapLocation;
```

Constants

`kCGHIDEventTap`

Specifies that an event tap is placed at the point where HID system events enter the window server.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGSessionEventTap`

Specifies that an event tap is placed at the point where HID system and remote control events enter a login session.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGAnnotatedSessionEventTap`
: Specifies that an event tap is placed at the point where session events have been annotated to flow to an application.

  Available in Mac OS X v10.4 and later.

  Declared in `CGEventTypes.h`.

Discussion
In addition to the three tapping points described above, an event tap may also be placed where annotated events are delivered to a specific application. For more information, see the function `CGEventTapCreateForPSN` (page 519).

Event Tap Options

Constants that specify whether a new event tap is an active filter or a passive listener.

```
enum _CGEventTapOptions {
    kCGEventTapOptionDefault = 0x00000000,
    kCGEventTapOptionListenOnly = 0x00000001
};
typedef uint32_t CGEventTapOptions;
```

Constants
`kCGEventTapOptionDefault`
: Specifies that a new event tap is an active filter. (Applications targeting Mac OS X v10.4 should use the literal value to create an active filter event tap, as this constant was omitted from the header.)

  Available in Mac OS X v10.5 and later.

  Declared in `CGEventTypes.h`.

`kCGEventTapOptionListenOnly`
: Specifies that a new event tap is a passive listener.

  Available in Mac OS X v10.4 and later.

  Declared in `CGEventTypes.h`.

Discussion
When you create an event tap, you indicate whether it is a passive listener or active event filter. A passive listener receives events but cannot modify or divert them. An active filter may pass an event through unmodified, modify an event, or discard an event.

Event Tap Placement

Constants that specify where a new event tap is inserted into the list of active event taps.

```
enum _CGEventTapPlacement {
    kCGHeadInsertEventTap = 0,
    kCGTailAppendEventTap
};
typedef uint32_t CGEventTapPlacement;
```

Constants

`kCGHeadInsertEventTap`
Specifies that a new event tap should be inserted before any pre-existing event taps at the same location.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGTailAppendEventTap`
Specifies that a new event tap should be inserted after any pre-existing event taps at the same location.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

Discussion
Event taps may be inserted at a specified location at the head of pre-existing filters, or appended after any pre-existing filters.

Event Types

Constants that specify the different types of input events.

```
enum _CGEventType {
    kCGEventNull                = NX_NULLEVENT,
    kCGEventLeftMouseDown       = NX_LMOUSEDOWN,
    kCGEventLeftMouseUp         = NX_LMOUSEUP,
    kCGEventRightMouseDown      = NX_RMOUSEDOWN,
    kCGEventRightMouseUp        = NX_RMOUSEUP,
    kCGEventMouseMoved          = NX_MOUSEMOVED,
    kCGEventLeftMouseDragged    = NX_LMOUSEDRAGGED,
    kCGEventRightMouseDragged   = NX_RMOUSEDRAGGED,
    kCGEventKeyDown             = NX_KEYDOWN,
    kCGEventKeyUp               = NX_KEYUP,
    kCGEventFlagsChanged        = NX_FLAGSCHANGED,
    kCGEventScrollWheel         = NX_SCROLLWHEELMOVED,
    kCGEventTabletPointer       = NX_TABLETPOINTER,
    kCGEventTabletProximity     = NX_TABLETPROXIMITY,
    kCGEventOtherMouseDown      = NX_OMOUSEDOWN,
    kCGEventOtherMouseUp        = NX_OMOUSEUP,
    kCGEventOtherMouseDragged   = NX_OMOUSEDRAGGED,
    kCGEventTapDisabledByTimeout = 0xFFFFFFFE,
    kCGEventTapDisabledByUserInput = 0xFFFFFFFF
};
typedef uint32_t CGEventType;
```

Constants

`kCGEventNull`
Specifies a null event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventLeftMouseDown`
Specifies a mouse down event with the left button.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventLeftMouseUp`
Specifies a mouse up event with the left button.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventRightMouseDown`
Specifies a mouse down event with the right button.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventRightMouseUp`
Specifies a mouse up event with the right button.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventMouseMoved`
Specifies a mouse moved event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventLeftMouseDragged`
Specifies a mouse drag event with the left button down.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventRightMouseDragged`
Specifies a mouse drag event with the right button down.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventKeyDown`
Specifies a key down event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventKeyUp`
Specifies a key up event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventFlagsChanged`
Specifies a key changed event for a modifier or status key.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventScrollWheel`
Specifies a scroll wheel moved event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventTabletPointer`
Specifies a tablet pointer event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventTabletProximity`
Specifies a tablet proximity event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventOtherMouseDown`
Specifies a mouse down event with one of buttons 2-31.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventOtherMouseUp`
Specifies a mouse up event with one of buttons 2-31.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventOtherMouseDragged`
Specifies a mouse drag event with one of buttons 2-31 down.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventTapDisabledByTimeout`
Specifies an event indicating the event tap is disabled because of timeout.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventTapDisabledByUserInput`
Specifies an event indicating the event tap is disabled because of user input.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

Discussion
These constants are used:

- In the functions `CGEventTapCreate` (page 517) and `CGEventTapCreateForPSN` (page 519) to specify the events of interest for the new event tap.
- To indicate the event type passed to your event tap callback function.
- In the function `CGEventCreateMouseEvent` (page 496) to specify the type of mouse event.
- In the functions `CGEventGetType` (page 501) and `CGEventSetType` (page 508) to identify the event type.
- In the functions `CGEventSourceCounterForEventType` (page 509) and `CGEventSourceSecondsSinceLastEventType` (page 514) to indicate the event type.

Note that tablet devices may generate mouse events with embedded tablet data, or tablet pointer and proximity events. Tablet mouse events allow tablets to be used with applications that are not tablet-aware.

Event Type Mask

Specifies an event mask that represents all event types.

```
#define kCGEventMaskForAllEvents (~(CGEventMask)0)
```

Constants

`kCGEventMaskForAllEvents`

An event mask that specifies all event types.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

Discussion

This constant is typically used with the functions `CGEventTapCreate` (page 517) and `CGEventTapCreateForPSN` (page 519) to register an event tap that observes all input events.

Mouse Buttons

Constants that specify buttons on a one, two, or three-button mouse.

```
enum _CGMouseButton {
    kCGMouseButtonLeft = 0,
    kCGMouseButtonRight = 1,
    kCGMouseButtonCenter = 2
};
typedef uint32_t CGMouseButton;
```

Constants

`kCGMouseButtonLeft`

Specifies the only mouse button on a one-button mouse, or the left mouse button on a two-button or three-button mouse.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGMouseButtonRight`

Specifies the right mouse button on a two-button or three-button mouse.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGMouseButtonCenter`

Specifies the center mouse button on a three-button mouse.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

Discussion

Quartz supports up to 32 mouse buttons. The first three buttons are specified using these three constants. Additional buttons are specified in USB order using the integers 3 to 31.

These constants are used:

- In the function `CGEventCreateMouseEvent` (page 496) to specify the button that's changing state.
- In the function `CGEventSourceButtonState` (page 508) to specify the button that's being tested.
- To specify the value of the `kCGMouseEventButtonNumber` event field when modifying an event.

Mouse Subtypes

Constants used with the `kCGMouseEventSubtype` event field.

```
enum _CGEventMouseSubtype {
    kCGEventMouseSubtypeDefault = 0,
    kCGEventMouseSubtypeTabletPoint = 1,
    kCGEventMouseSubtypeTabletProximity = 2
};
typedef uint32_t CGEventMouseSubtype;
```

Constants

`kCGEventMouseSubtypeDefault`
Specifies that the event is an ordinary mouse event, and does not contain additional tablet device information.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventMouseSubtypeTabletPoint`
Specifies that the mouse event originated from a tablet device, and that the various `kCGTabletEvent` field selectors may be used to obtain tablet-specific data from the mouse event.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

`kCGEventMouseSubtypeTabletProximity`
Specifies that the mouse event originated from a tablet device with the pen in proximity but not necessarily touching the tablet, and that the various `kCGTabletProximity` field selectors may be used to obtain tablet-specific data from the mouse event. This is often used with mouse move events originating from a tablet.

Available in Mac OS X v10.4 and later.

Declared in `CGEventTypes.h`.

Discussion

Tablets may generate specially annotated mouse events that contain values associated with the `kCGMouseEventSubtype` event field. To learn how to set these values, see the function `CGEventSetIntegerValueField` (page 506).

Scrolling Event Units

Constants that specify the unit of measurement for a scrolling event.

```
enum {
    kCGScrollEventUnitPixel = 0,
    kCGScrollEventUnitLine = 1
};
typedef uint32_t CGScrollEventUnit;
```

Constants

`kCGScrollEventUnitPixel`

Specifies that the unit of measurement is pixels.

Available in Mac OS X v10.5 and later.

Declared in `CGEventTypes.h`.

`kCGScrollEventUnitLine`

Specifies that the unit of measurement is lines.

Available in Mac OS X v10.5 and later.

Declared in `CGEventTypes.h`.

Discussion

You may pass one of these constants to the function `CGEventCreateScrollWheelEvent` (page 497) to specify the unit of measurement for the event. The constant `kCGScrollEventUnitPixel` produces an event that most applications interpret as a smooth scrolling event. By default, the scale is about ten pixels per line. You can alter the scale with the function `CGEventSourceSetPixelsPerLine` (page 516).

Other References

CGImageProperties Reference

Framework: ApplicationServices/ImageIO

Declared in CGImageProperties.h

Overview

CGImageProperties Reference defines constants that represent characteristics of images used by the Image I/O framework.

Constants

Format-Specific Dictionaries

Properties that have an associated dictionary of file-format or metadata-format specific key-value pairs.

```
CFStringRef kCGImagePropertyTIFFDictionary;
CFStringRef kCGImagePropertyGIFDictionary;
CFStringRef kCGImagePropertyJFIFDictionary;
CFStringRef kCGImagePropertyExifDictionary;
CFStringRef kCGImagePropertyPNGDictionary;
CFStringRef kCGImagePropertyIPTCDictionary;
CFStringRef kCGImagePropertyGPSDictionary;
CFStringRef kCGImagePropertyRawDictionary;
CFStringRef kCGImagePropertyCIFFDictionary;
CFStringRef kCGImageProperty8BIMDictionary;
CFStringRef kCGImagePropertyDNGDictionary;
CFStringRef kCGImagePropertyExifAuxDictionary;
```

Constants

`kCGImagePropertyTIFFDictionary`

A dictionary of key-value pairs for an image that uses Tagged Image File Format (TIFF). See "TIFF Dictionary Keys" (page 580).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGIFDictionary`

A dictionary of key-value pairs for an image that uses Graphics Interchange Format (GIF). See "GIF Dictionary Keys" (page 568).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyJFIFDictionary`
A dictionary of key-value pairs for an image that uses JPEG File Interchange Format (JFIF). See "JFIF Dictionary Keys" (page 578).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifDictionary`
A dictionary of key-value pairs for an image that uses Exchangeable Image File Format (EXIF). See "EXIF Dictionary Keys" (page 559).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyPNGDictionary`
A dictionary of key-value pairs for an image that uses Portable Network Graphics (PNG) format. See "PNG Dictionary Keys" (page 579).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCDictionary`
A dictionary of key-value pairs for an image that uses International Press Telecommunications Council (IPTC) metadata. See "IPTC Dictionary Keys" (page 572).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDictionary`
A dictionary of key-value pairs for an image that has Global Positioning System (GPS) information. See "GPS Dictionary Keys" (page 568).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyRawDictionary`
A dictionary of key-value pairs for an image that contains minimally processed, or raw, data.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFDictionary`
A dictionary of key-value pairs for an image that uses Camera Image File Format (CIFF). See "CIFF Dictionary Keys" (page 584).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImageProperty8BIMDictionary`
A dictionary of key-value pairs for an Adobe Photoshop image. See "8BIM Dictionary Keys" (page 584).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyDNGDictionary`
A dictionary of key-value pairs for an image that uses the Digital Negative (DNG) archival format. See "DNG Dictionary Keys" (page 583).

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifAuxDictionary`
An auxiliary dictionary of key-value pairs for an image that uses Exchangeable Image File Format (EXIF).

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

Discussion
If any of these constants are returned by the functions`CGImageSourceCopyProperties` (page 234) or `CGImageSourceCopyPropertiesAtIndex` (page 235) the associated value is a dictionary of file-format or metadata-format specific key-value pairs.

Declared In
`CGImageProperties.h`

Camera Maker Dictionaries

Properties that have an associated dictionary of key-value pairs for a specific camera manufacturer.

```
CFStringRef kCGImagePropertyMakerCanonDictionary;
CFStringRef kCGImagePropertyMakerNikonDictionary;
CFStringRef kCGImagePropertyMakerMinoltaDictionary;
CFStringRef kCGImagePropertyMakerFujiDictionary;
CFStringRef kCGImagePropertyMakerOlympusDictionary;
CFStringRef kCGImagePropertyMakerPentaxDictionary;
```

Constants
`kCGImagePropertyMakerCanonDictionary`
A dictionary of key-value pairs for an image from a Canon camera. See "Canon Camera Dictionary Keys" (page 589).

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonDictionary`
A dictionary of key-value pairs for an image from a Nikon camera. See "Nikon Camera Dictionary Keys" (page 586).

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerMinoltaDictionary`
A dictionary of key-value pairs for an image from a Minolta camera.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerFujiDictionary`
A dictionary of key-value pairs for an image from a Fuji camera.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerOlympusDictionary`
A dictionary of key-value pairs for an image from a Olympus camera.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerPentaxDictionary`
A dictionary of key-value pairs for an image from a Pentax camera.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

Image Source Container Properties

Properties that apply to the container in general but not necessarily to any individual image in the container.

```
CFStringRef kCGImagePropertyFileSize;
```

Constants
`kCGImagePropertyFileSize`
The size of the image file in bytes, if known. If present, this key is a `CFNumber` value.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

Discussion
These properties can be returned by the function `CGImageSourceCopyProperties` (page 234).

Declared In
`CGImageProperties.h`

Individual Image Properties

Properties that apply to an individual image in an image source.

```
CFStringRef kCGImagePropertyDPIHeight;
CFStringRef kCGImagePropertyDPIWidth;
CFStringRef kCGImagePropertyPixelWidth;
CFStringRef kCGImagePropertyPixelHeight;
CFStringRef kCGImagePropertyDepth;
CFStringRef kCGImagePropertyOrientation;
CFStringRef kCGImagePropertyIsFloat;
CFStringRef kCGImagePropertyIsIndexed;
CFStringRef kCGImagePropertyHasAlpha;
CFStringRef kCGImagePropertyColorModel;
CFStringRef kCGImagePropertyProfileName;
```

Constants
`kCGImagePropertyDPIHeight`
The resolution, in dots per inch, in the x dimension. If present, this key is a `CFNumber` value.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyDPIWidth`
The resolution, in dots per inch, in the y dimension. If present, this key is a `CFNumber` value.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyPixelWidth`
The number of pixels in the x dimension. If present, this key is a `CFNumber` value.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyPixelHeight`
The number of pixels in the y dimension. If present, this key is a `CFNumber` value.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyDepth`
The number of bits in each color sample of each pixel. If present, this key is a `CFNumber` value.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyOrientation`
The intended display orientation of the image. If present, this key is a `CFNumber` value with the same value as defined by the TIFF and EXIF specifications. The value specifies where the origin `(0,0)` of the image is locates, as shown in Table 32-1. If not present, a value of 1 is assumed.

Table 32-1

Value	Location of the origin of the image
1	Top, left
2	Top, right
3	Bottom, right
4	Bottom, left
5	Left, top
6	Right, top
7	Right, bottom
8	Left, bottom

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIsFloat`
Whether or not the image contains floating-point pixel samples. The value of this key is `kCFBooleanTrue` if the image contains them.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIsIndexed`
Whether or not the image contains indexed pixel samples (sometimes called paletted samples). The value of this key is `kCFBooleanTrue` if the image contains them.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyHasAlpha`
Whether or not the image has an alpha channel. The value of this key is `kCFBooleanTrue` if the image contains an alpha channel.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyColorModel`
The color model of the image such as, "RGB", "CMYK", "Gray", or "Lab". The value of this key is CFStringRef.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyProfileName`
The name of the optional ICC profile embedded in the image, if known. If present, the value of this key is a CFStringRef.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

Discussion
These properties can be returned by the function `CGImageSourceCopyPropertiesAtIndex` (page 235).

Declared In
`CGImageProperties.h`

Color Model Values

Values for the color model property.

```
const CFStringRef kCGImagePropertyColorModelRGB;
const CFStringRef kCGImagePropertyColorModelGray;
const CFStringRef kCGImagePropertyColorModelCMYK;
const CFStringRef kCGImagePropertyColorModelLab;
```

Constants
`kCGImagePropertyColorModelRGB`
An RGB color model.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyColorModelGray`
A Gray color model.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyColorModelCMYK`
A CMYK color model.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyColorModelLab`
A Lab color model.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

Discussion
A color model describes how color values are represented mathematically. A color space is a color model combined with a definition of how to interpret values within the model.

Declared In
`CGImageProperties.h`

EXIF Dictionary Keys

Keys for for an image that uses Exchangeable Image File Format (EXIF).

```
const CFStringRef kCGImagePropertyExifExposureTime;
const CFStringRef kCGImagePropertyExifFNumber;
const CFStringRef kCGImagePropertyExifExposureProgram;
const CFStringRef kCGImagePropertyExifSpectralSensitivity;
const CFStringRef kCGImagePropertyExifISOSpeedRatings;
const CFStringRef kCGImagePropertyExifOECF;
const CFStringRef kCGImagePropertyExifVersion;
const CFStringRef kCGImagePropertyExifDateTimeOriginal;
const CFStringRef kCGImagePropertyExifDateTimeDigitized;
const CFStringRef kCGImagePropertyExifComponentsConfiguration;
const CFStringRef kCGImagePropertyExifCompressedBitsPerPixel;
const CFStringRef kCGImagePropertyExifShutterSpeedValue;
const CFStringRef kCGImagePropertyExifApertureValue;
const CFStringRef kCGImagePropertyExifBrightnessValue;
const CFStringRef kCGImagePropertyExifExposureBiasValue;
const CFStringRef kCGImagePropertyExifMaxApertureValue;
const CFStringRef kCGImagePropertyExifSubjectDistance;
const CFStringRef kCGImagePropertyExifMeteringMode;
const CFStringRef kCGImagePropertyExifLightSource;
const CFStringRef kCGImagePropertyExifFlash;
const CFStringRef kCGImagePropertyExifFocalLength;
const CFStringRef kCGImagePropertyExifSubjectArea;
const CFStringRef kCGImagePropertyExifMakerNote;
const CFStringRef kCGImagePropertyExifUserComment;
const CFStringRef kCGImagePropertyExifSubsecTime;
const CFStringRef kCGImagePropertyExifSubsecTimeOrginal;
const CFStringRef kCGImagePropertyExifSubsecTimeDigitized;
const CFStringRef kCGImagePropertyExifFlashPixVersion;
const CFStringRef kCGImagePropertyExifColorSpace;
const CFStringRef kCGImagePropertyExifPixelXDimension;
const CFStringRef kCGImagePropertyExifPixelYDimension;
const CFStringRef kCGImagePropertyExifRelatedSoundFile;
const CFStringRef kCGImagePropertyExifFlashEnergy;
const CFStringRef kCGImagePropertyExifSpatialFrequencyResponse;
const CFStringRef kCGImagePropertyExifFocalPlaneXResolution;
const CFStringRef kCGImagePropertyExifFocalPlaneYResolution;
const CFStringRef kCGImagePropertyExifFocalPlaneResolutionUnit;
const CFStringRef kCGImagePropertyExifSubjectLocation;
const CFStringRef kCGImagePropertyExifExposureIndex;
const CFStringRef kCGImagePropertyExifSensingMethod;
const CFStringRef kCGImagePropertyExifFileSource;
const CFStringRef kCGImagePropertyExifSceneType;
const CFStringRef kCGImagePropertyExifCFAPattern;
const CFStringRef kCGImagePropertyExifCustomRendered;
const CFStringRef kCGImagePropertyExifExposureMode;
const CFStringRef kCGImagePropertyExifWhiteBalance;
const CFStringRef kCGImagePropertyExifDigitalZoomRatio;
const CFStringRef kCGImagePropertyExifFocalLenIn35mmFilm;
const CFStringRef kCGImagePropertyExifSceneCaptureType;
const CFStringRef kCGImagePropertyExifGainControl;
const CFStringRef kCGImagePropertyExifContrast;
const CFStringRef kCGImagePropertyExifSaturation;
const CFStringRef kCGImagePropertyExifSharpness;
const CFStringRef kCGImagePropertyExifDeviceSettingDescription;
const CFStringRef kCGImagePropertyExifSubjectDistRange;
const CFStringRef kCGImagePropertyExifImageUniqueID;
const CFStringRef kCGImagePropertyExifGamma;
```

Constants

`kCGImagePropertyExifExposureTime`
The exposure time.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFNumber`
The F number.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifExposureProgram`
The exposure program.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSpectralSensitivity`
The spectral sensitivity of each channel.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifISOSpeedRatings`
ISO speed ratings.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifOECF`
The opto-electrical conversion function (OECF), which defines the relationship between the optical input of the camera and the image values.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifVersion`
The version.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifDateTimeOriginal`
The original date and time.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifDateTimeDigitized`
The digitized date and time.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifComponentsConfiguration`
The components configuration. For compressed data, specifies that the channels of each component are arranged in increasing numeric order (from first component to the fourth).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifCompressedBitsPerPixel`
The compressed bits per pixel.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifShutterSpeedValue`
The shutter speed value.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifApertureValue`
The aperture value.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifBrightnessValue`
The brightness value.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifExposureBiasValue`
The exposure bias value.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifMaxApertureValue`
The maximum aperture value.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSubjectDistance`
The distance to the subject, in meters.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifMeteringMode`
The metering mode.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifLightSource`
The light source.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFlash`
The flash status when the image was shot.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFocalLength`
The focal length.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSubjectArea`
The subject area.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifMakerNote`
A maker note.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifUserComment`
A user comment.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSubsecTime`
The fraction of seconds for the date and time tag.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSubsecTimeOrginal`
The fraction of seconds for the original date and time tag.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSubsecTimeDigitized`
The fraction of seconds for the digitized time tag.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFlashPixVersion`
The FlashPix version supported by an FPXR file. FlashPix is a format for multi-resolution, tiled images, that facilitates fast onscreen viewing.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifColorSpace`
The color space.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifPixelXDimension`
The pixel x dimension.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifPixelYDimension`
The pixel y dimension.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifRelatedSoundFile`
A related sound file.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFlashEnergy`
The strobe energy when the image was captures, in beam candle power seconds.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSpatialFrequencyResponse`
The spatial frequency table and spatial frequency response values in the direction of image width, image height, and diagonal directions. See ISO 12233..

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFocalPlaneXResolution`
The number of image-width pixels (x) per focal plane resolution unit.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFocalPlaneYResolution`
The number of image-height pixels (y)per focal plane resolution unit.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFocalPlaneResolutionUnit`
The unit of measurement for the focal plane x and y tags.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSubjectLocation`
The location of the scene's primary subject.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifExposureIndex`
The selected exposure index.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSensingMethod`
The sensor type of the camera or input device.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFileSource`
The image source.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSceneType`
The scene type.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifCFAPattern`
The color filter array (CFA) pattern, which is the geometric patter of the image sensor for a 1-chip color sensor area.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifCustomRendered`
Special rendering performed on the image data.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifExposureMode`
The exposure mode setting.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifWhiteBalance`
The white balance mode.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifDigitalZoomRatio`
The digital zoom ratio.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifFocalLenIn35mmFilm`
The equivalent focal length in 35 mm film.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSceneCaptureType`
The scene capture type (standard, landscape, portrait, night).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifGainControl`
The gain adjustment applied to the image.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyExifContrast`
The contrast applied to the image.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSaturation`
The saturation applied to the image.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSharpness`
The sharpness applied to the image.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifDeviceSettingDescription`
For a particular camera mode, indicates the conditions for taking the picture.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifSubjectDistRange`
The subject distance range.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifImageUniqueID`
The unique ID of the image.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifGamma`
The gamma setting.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

EXIF Auxiliary Dictionary Keys

Auxiliary keys for for an image that uses Exchangeable Image File Format (EXIF).

```
const CFStringRef kCGImagePropertyExifAuxLensInfo;
const CFStringRef kCGImagePropertyExifAuxLensModel;
const CFStringRef kCGImagePropertyExifAuxSerialNumber;
const CFStringRef kCGImagePropertyExifAuxLensID;
const CFStringRef kCGImagePropertyExifAuxLensSerialNumber;
const CFStringRef kCGImagePropertyExifAuxImageNumber;
const CFStringRef kCGImagePropertyExifAuxFlashCompensation;
const CFStringRef kCGImagePropertyExifAuxOwnerName;
const CFStringRef kCGImagePropertyExifAuxFirmware;
```

Constants

`kCGImagePropertyExifAuxLensInfo`
Lens information.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifAuxLensModel`
The lens model.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifAuxSerialNumber`
The serial number.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifAuxLensID`
The lens ID.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifAuxLensSerialNumber`
The lens serial number.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifAuxImageNumber`
The image number.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifAuxFlashCompensation`
Flash compensation.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifAuxOwnerName`
The owner name.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyExifAuxFirmware`
Firmware information.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

GIF Dictionary Keys

Keys for an image that uses Graphics Interchange Format (GIF).

```
const CFStringRef kCGImagePropertyGIFLoopCount;
const CFStringRef kCGImagePropertyGIFDelayTime;
const CFStringRef kCGImagePropertyGIFImageColorMap;
const CFStringRef kCGImagePropertyGIFHasGlobalColorMap;
```

Constants
`kCGImagePropertyGIFLoopCount`
The loop count.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGIFDelayTime`
The delay time.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGIFImageColorMap`
The image color map.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGIFHasGlobalColorMap`
Whether or not the GIF has a global color map.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

GPS Dictionary Keys

Keys for an image that has Global Positioning System (GPS) information.

```
const CFStringRef kCGImagePropertyGPSVersion;
const CFStringRef kCGImagePropertyGPSLatitudeRef;
const CFStringRef kCGImagePropertyGPSLatitude;
const CFStringRef kCGImagePropertyGPSLongitudeRef;
const CFStringRef kCGImagePropertyGPSLongitude;
const CFStringRef kCGImagePropertyGPSAltitudeRef;
const CFStringRef kCGImagePropertyGPSAltitude;
const CFStringRef kCGImagePropertyGPSTimeStamp;
const CFStringRef kCGImagePropertyGPSSatellites;
const CFStringRef kCGImagePropertyGPSStatus;
const CFStringRef kCGImagePropertyGPSMeasureMode;
const CFStringRef kCGImagePropertyGPSDOP;
const CFStringRef kCGImagePropertyGPSSpeedRef;
const CFStringRef kCGImagePropertyGPSSpeed;
const CFStringRef kCGImagePropertyGPSTrackRef;
const CFStringRef kCGImagePropertyGPSTrack;
const CFStringRef kCGImagePropertyGPSImgDirectionRef;
const CFStringRef kCGImagePropertyGPSImgDirection;
const CFStringRef kCGImagePropertyGPSMapDatum;
const CFStringRef kCGImagePropertyGPSDestLatitudeRef;
const CFStringRef kCGImagePropertyGPSDestLatitude;
const CFStringRef kCGImagePropertyGPSDestLongitudeRef;
const CFStringRef kCGImagePropertyGPSDestLongitude;
const CFStringRef kCGImagePropertyGPSDestBearingRef;
const CFStringRef kCGImagePropertyGPSDestBearing;
const CFStringRef kCGImagePropertyGPSDestDistanceRef;
const CFStringRef kCGImagePropertyGPSDestDistance;
const CFStringRef kCGImagePropertyGPSProcessingMethod;
const CFStringRef kCGImagePropertyGPSAreaInformation;
const CFStringRef kCGImagePropertyGPSDateStamp;
const CFStringRef kCGImagePropertyGPSDifferental;
```

Constants

`kCGImagePropertyGPSVersion`

The version.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSLatitudeRef`

Whether the latitude is northern or southern.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSLatitude`

The latitude.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSLongitudeRef`

Whether the longitude is east or west.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSLongitude`
The longitude.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSAltitudeRef`
The reference altitude.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSAltitude`
The altitude.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSTimeStamp`
The time as UTC (Coordinated Universal Time).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSSatellites`
The satellites used for GPS measurements.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSStatus`
The status of the GPS receiver.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSMeasureMode`
The measurement mode.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDOP`
The data degree of precision (DOP).

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSSpeedRef`
The unit for expressing the GPS receiver speed of movement.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSSpeed`
The GPS receiver speed of movement.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSTrackRef`
The reference for the direction of GPS receiver movement.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSTrack`
The direction of GPS receiver movement.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSImgDirectionRef`
The reference for the direction of the image.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSImgDirection`
The direction of the image.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSMapDatum`
The geodetic survey data used by the GPS receiver.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDestLatitudeRef`
Whether the latitude of the destination point is northern or southern.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDestLatitude`
The latitude of the destination point.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDestLongitudeRef`
Whether the longitude of the destination point is east or west.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDestLongitude`
The longitude of the destination point.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDestBearingRef`
The reference for giving the bearing to the destination point.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDestBearing`
The bearing to the destination point.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDestDistanceRef`
The units for expressing the distance to the destination point.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDestDistance`
The distance to the destination point.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSProcessingMethod`
The name of the method used for finding a location.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSAreaInformation`
The name of the GPS area.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDateStamp`
The data and time information relative to Coordinated Universal Time (UTC).
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyGPSDifferental`
Whether differential correction is applied to the GPS receiver.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

IPTC Dictionary Keys

Keys for an image that uses International Press Telecommunications Council (IPTC) metadata.

```
const CFStringRef kCGImagePropertyIPTCObjectTypeReference;
const CFStringRef kCGImagePropertyIPTCObjectAttributeReference;
const CFStringRef kCGImagePropertyIPTCObjectName;
const CFStringRef kCGImagePropertyIPTCEditStatus;
const CFStringRef kCGImagePropertyIPTCEditorialUpdate;
const CFStringRef kCGImagePropertyIPTCUrgency;
const CFStringRef kCGImagePropertyIPTCSubjectReference;
const CFStringRef kCGImagePropertyIPTCCategory;
const CFStringRef kCGImagePropertyIPTCSupplementalCategory;
const CFStringRef kCGImagePropertyIPTCFixtureIdentifier;
const CFStringRef kCGImagePropertyIPTCKeywords;
const CFStringRef kCGImagePropertyIPTCContentLocationCode;
const CFStringRef kCGImagePropertyIPTCContentLocationName;
const CFStringRef kCGImagePropertyIPTCReleaseDate;
const CFStringRef kCGImagePropertyIPTCReleaseTime;
const CFStringRef kCGImagePropertyIPTCExpirationDate;
const CFStringRef kCGImagePropertyIPTCExpirationTime;
const CFStringRef kCGImagePropertyIPTCSpecialInstructions;
const CFStringRef kCGImagePropertyIPTCActionAdvised;
const CFStringRef kCGImagePropertyIPTCReferenceService;
const CFStringRef kCGImagePropertyIPTCReferenceDate;
const CFStringRef kCGImagePropertyIPTCReferenceNumber;
const CFStringRef kCGImagePropertyIPTCDateCreated;
const CFStringRef kCGImagePropertyIPTCTimeCreated;
const CFStringRef kCGImagePropertyIPTCDigitalCreationDate;
const CFStringRef kCGImagePropertyIPTCDigitalCreationTime;
const CFStringRef kCGImagePropertyIPTCOriginatingProgram;
const CFStringRef kCGImagePropertyIPTCProgramVersion;
const CFStringRef kCGImagePropertyIPTCObjectCycle;
const CFStringRef kCGImagePropertyIPTCByline;
const CFStringRef kCGImagePropertyIPTCBylineTitle;
const CFStringRef kCGImagePropertyIPTCCity;
const CFStringRef kCGImagePropertyIPTCSubLocation;
const CFStringRef kCGImagePropertyIPTCProvinceState;
const CFStringRef kCGImagePropertyIPTCCountryPrimaryLocationCode;
const CFStringRef kCGImagePropertyIPTCCountryPrimaryLocationName;
const CFStringRef kCGImagePropertyIPTCOriginalTransmissionReference;
const CFStringRef kCGImagePropertyIPTCHeadline;
const CFStringRef kCGImagePropertyIPTCCredit;
const CFStringRef kCGImagePropertyIPTCSource;
const CFStringRef kCGImagePropertyIPTCCopyrightNotice;
const CFStringRef kCGImagePropertyIPTCContact;
const CFStringRef kCGImagePropertyIPTCCaptionAbstract;
const CFStringRef kCGImagePropertyIPTCWriterEditor;
const CFStringRef kCGImagePropertyIPTCImageType;
const CFStringRef kCGImagePropertyIPTCImageOrientation;
const CFStringRef kCGImagePropertyIPTCLanguageIdentifier;
const CFStringRef kCGImagePropertyIPTCStarRating;
```

Constants

`kCGImagePropertyIPTCObjectTypeReference`

The object type.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCObjectAttributeReference`
The object attribute.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCObjectName`
The object name.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCEditStatus`
The edit status.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCEditorialUpdate`
An editorial update.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCUrgency`
The urgency level.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCSubjectReference`
The subject.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCCategory`
The category.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCSupplementalCategory`
A supplemental category.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCFixtureIdentifier`
A fixture identifier.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCKeywords`
Keywords.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCContentLocationCode`
The content location code.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCContentLocationName`
The content location name.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCReleaseDate`
The release date.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCReleaseTime`
The release time.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCExpirationDate`
The expiration date.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCExpirationTime`
The expiration time.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCSpecialInstructions`
Special instructions.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCActionAdvised`
The advised action.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCReferenceService`
The reference service.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCReferenceDate`
The reference date.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCReferenceNumber`
The reference number.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCDateCreated`
The date created.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCTimeCreated`
The time created.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCDigitalCreationDate`
The digital creation date.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCDigitalCreationTime`
The digital creation time.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCOriginatingProgram`
The originating program.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCProgramVersion`
The program version.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCObjectCycle`
The object cycle.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCByline`
The byline.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCBylineTitle`
The byline title.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCCity`
The city.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCSubLocation`
The sublocation.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCProvinceState`
The province or state.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCCountryPrimaryLocationCode`
The country primary location code.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCCountryPrimaryLocationName`
The country primary location name.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCOriginalTransmissionReference`
The original transmission reference.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCHeadline`
The headline.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCCredit`
Credit information.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCSource`
The source.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCCopyrightNotice`
The copyright notice.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCContact`
Contact information.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCCaptionAbstract`
The caption abstract.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCWriterEditor`
The writer or editor.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCImageType`
The image type.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCImageOrientation`
The image orientation.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCLanguageIdentifier`
The language identifier.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyIPTCStarRating`
The star rating.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

Discussion
IPTC constants are metadata elements of the Information Interchange Model (IIM) used to provide information about images. The IIM was developer by the Newspaper Association of America (NAA) and the International Press Telecommunications Council (IPTC).

Declared In
`CGImageProperties.h`

JFIF Dictionary Keys

Keys for an image that uses JPEG File Interchange Format (JFIF).

```
const CFStringRef kCGImagePropertyJFIFVersion;
const CFStringRef kCGImagePropertyJFIFXDensity;
const CFStringRef kCGImagePropertyJFIFYDensity;
const CFStringRef kCGImagePropertyJFIFDensityUnit;
const CFStringRef kCGImagePropertyJFIFIsProgressive;
```

Constants

`kCGImagePropertyJFIFVersion`
The version.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyJFIFXDensity`
The x density.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyJFIFYDensity`
The y density.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyJFIFDensityUnit`
The density unit.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyJFIFIsProgressive`
Whether or not the image is progressive.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

PNG Dictionary Keys

Keys for an image that uses Portable Network Graphics (PNG) format.

```
const CFStringRef kCGImagePropertyPNGGamma;
const CFStringRef kCGImagePropertyPNGInterlaceType;
const CFStringRef kCGImagePropertyPNGXPixelsPerMeter;
const CFStringRef kCGImagePropertyPNGYPixelsPerMeter;
const CFStringRef kCGImagePropertyPNGsRGBIntent;
const CFStringRef kCGImagePropertyPNGChromaticities;
```

Constants

`kCGImagePropertyPNGGamma`
The gamma value.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyPNGInterlaceType`
The interlace type.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyPNGXPixelsPerMeter`
The number of x pixels per meter.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyPNGYPixelsPerMeter`
The number of y pixels per meter.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyPNGsRGBIntent`
The sRGB intent.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyPNGChromaticities`
The chromaticities.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

TIFF Dictionary Keys

Keys for an image that uses Tagged Image File Format (TIFF).

```
const CFStringRef kCGImagePropertyTIFFCompression;
const CFStringRef kCGImagePropertyTIFFPhotometricInterpretation;
const CFStringRef kCGImagePropertyTIFFDocumentName;
const CFStringRef kCGImagePropertyTIFFImageDescription;
const CFStringRef kCGImagePropertyTIFFMake;
const CFStringRef kCGImagePropertyTIFFModel;
const CFStringRef kCGImagePropertyTIFFOrientation;
const CFStringRef kCGImagePropertyTIFFXResolution;
const CFStringRef kCGImagePropertyTIFFYResolution;
const CFStringRef kCGImagePropertyTIFFResolutionUnit;
const CFStringRef kCGImagePropertyTIFFSoftware;
const CFStringRef kCGImagePropertyTIFFTransferFunction;
const CFStringRef kCGImagePropertyTIFFDateTime;
const CFStringRef kCGImagePropertyTIFFArtist;
const CFStringRef kCGImagePropertyTIFFHostComputer;
const CFStringRef kCGImagePropertyTIFFCopyright;
const CFStringRef kCGImagePropertyTIFFWhitePoint;
const CFStringRef kCGImagePropertyTIFFPrimaryChromaticities;
```

Constants

`kCGImagePropertyTIFFCompression`
The compression scheme used on the image data.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFPhotometricInterpretation`
The color space of the image data.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFDocumentName`
The document name.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFImageDescription`
The image description.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFMake`
The camera or input device make.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFModel`
A camera or input device model.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFOrientation`
The image orientation.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFXResolution`
The number of pixels per resolution unit in the image width direction.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFYResolution`
The number of pixels per resolution unit in the image height direction.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFResolutionUnit`
The units of resolution.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFSoftware`
The name and version of the software used for image creation.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFTransferFunction`
The transfer function, in tabular format, used to map pixel components from a nonlinear form into a linear form.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFDateTime`
The date and time.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFArtist`
The artist.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFHostComputer`
The computer or operation system used when the image was created.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFCopyright`
Copyright information.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFWhitePoint`
The white point.
Available in Mac OS X v10.4 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyTIFFPrimaryChromaticities`
The chromaticities of the primaries of the image.

Available in Mac OS X v10.4 and later.

Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

DNG Dictionary Keys

Keys for an image that uses the Digital Negative (DNG) archival format.

```
CFStringRef  kCGImagePropertyDNGVersion;
CFStringRef  kCGImagePropertyDNGBackwardVersion;
CFStringRef  kCGImagePropertyDNGUniqueCameraModel;
CFStringRef kCGImagePropertyDNGLocalizedCameraModel;
CFStringRef  kCGImagePropertyDNGCameraSerialNumber;
CFStringRef  kCGImagePropertyDNGLensInfo;
```

Constants
`kCGImagePropertyDNGVersion`
An encoding of the four-tier version number.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyDNGBackwardVersion`
The oldest version for which a file is compatible.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyDNGUniqueCameraModel`
A unique, nonlocalized name for the camera mode.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyDNGLocalizedCameraModel`
The localized camera model name.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyDNGCameraSerialNumber`
The camera serial number.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyDNGLensInfo`
Information about the lens used for the image.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

8BIM Dictionary Keys

A key for an Adobe Photoshop image.

```
CFStringRef  kCGImageProperty8BIMLayerNames;
```

Constants

`kCGImageProperty8BIMLayerNames`

The layer names for an Adobe Photoshop file.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

Declared In

`CGImageProperties.h`

CIFF Dictionary Keys

Keys for an image that uses Camera Image File Format (CIFF).

```
CFStringRef  kCGImagePropertyCIFFDescription;
CFStringRef  kCGImagePropertyCIFFFirmware;
CFStringRef  kCGImagePropertyCIFFOwnerName;
CFStringRef  kCGImagePropertyCIFFImageName;
CFStringRef  kCGImagePropertyCIFFImageFileName;
CFStringRef  kCGImagePropertyCIFFReleaseMethod;
CFStringRef  kCGImagePropertyCIFFReleaseTiming;
CFStringRef  kCGImagePropertyCIFFRecordID;
CFStringRef  kCGImagePropertyCIFFSelfTimingTime;
CFStringRef  kCGImagePropertyCIFFCameraSerialNumber;
CFStringRef  kCGImagePropertyCIFFImageSerialNumber;
CFStringRef  kCGImagePropertyCIFFContinuousDrive;
CFStringRef  kCGImagePropertyCIFFFocusMode;
CFStringRef  kCGImagePropertyCIFFMeteringMode;
CFStringRef  kCGImagePropertyCIFFShootingMode;
CFStringRef  kCGImagePropertyCIFFLensMaxMM;
CFStringRef  kCGImagePropertyCIFFLensMinMM;
CFStringRef  kCGImagePropertyCIFFLensModel;
CFStringRef  kCGImagePropertyCIFFWhiteBalanceIndex;
CFStringRef  kCGImagePropertyCIFFFlashExposureComp;
CFStringRef  kCGImagePropertyCIFFMeasuredEV;
```

Constants

`kCGImagePropertyCIFFDescription`

The camera description..

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFFirmware`

The firmware version.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFOwnerName`
The owner name.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFImageName`
The image name.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFImageFileName`
The image file name.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFReleaseMethod`
The release method.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFReleaseTiming`
The release timing.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFRecordID`
The record ID>
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFSelfTimingTime`
The self timing time.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFCameraSerialNumber`
The camera serial number.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFImageSerialNumber`
The image serial number.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFContinuousDrive`
The continuous drive mode.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFFocusMode`
The focus mode.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFMeteringMode`
The metering mode.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFShootingMode`
The shooting mode.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFLensMaxMM`
The maximum lens length.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFLensMinMM`
The minimum lens length.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFLensModel`
The lens model.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFWhiteBalanceIndex`
The white balance index.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFFlashExposureComp`
The flash exposure compensation.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyCIFFMeasuredEV`
The measured EV.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

Nikon Camera Dictionary Keys

Keys for an image from a Nikon camera.

```
CFStringRef  kCGImagePropertyMakerNikonISOSetting;
CFStringRef  kCGImagePropertyMakerNikonColorMode;
CFStringRef  kCGImagePropertyMakerNikonQuality;
CFStringRef  kCGImagePropertyMakerNikonWhiteBalanceMode;
CFStringRef  kCGImagePropertyMakerNikonSharpenMode;
CFStringRef  kCGImagePropertyMakerNikonFocusMode;
CFStringRef  kCGImagePropertyMakerNikonFlashSetting;
CFStringRef  kCGImagePropertyMakerNikonISOSelection;
CFStringRef  kCGImagePropertyMakerNikonFlashExposureComp;
CFStringRef  kCGImagePropertyMakerNikonImageAdjustment;
CFStringRef  kCGImagePropertyMakerNikonLensAdapter;
CFStringRef  kCGImagePropertyMakerNikonLensType;
CFStringRef  kCGImagePropertyMakerNikonLensInfo;
CFStringRef  kCGImagePropertyMakerNikonFocusDistance;
CFStringRef  kCGImagePropertyMakerNikonDigitalZoom;
CFStringRef  kCGImagePropertyMakerNikonShootingMode;
CFStringRef  kCGImagePropertyMakerNikonShutterCount;
CFStringRef  kCGImagePropertyMakerNikonCameraSerialNumber;
```

Constants

`kCGImagePropertyMakerNikonISOSetting`
The ISO setting.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonColorMode`
The color mode.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonQuality`
The quality setting.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonWhiteBalanceMode`
The white balance mode.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonSharpenMode`
The sharpening mode.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonFocusMode`
The focus mode.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonFlashSetting`
The flash setting.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonISOSelection`
The ISO selection.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonFlashExposureComp`
The flash exposure compensation.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonImageAdjustment`
Image adjustment setting.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonLensAdapter`
The lens adapter.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonLensType`
The lens type.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonLensInfo`
Lens information.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonFocusDistance`
The focus distance.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonDigitalZoom`
The digital zoom setting.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonShootingMode`
The shooting mode.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonShutterCount`
The shutter count.
Available in Mac OS X v10.5 and later.
Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerNikonCameraSerialNumber`
The camera serial number.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

Canon Camera Dictionary Keys

Keys for an image from a Canon camera.

```
CFStringRef  kCGImagePropertyMakerCanonOwnerName;
CFStringRef  kCGImagePropertyMakerCanonCameraSerialNumber;
CFStringRef  kCGImagePropertyMakerCanonImageSerialNumber;
CFStringRef  kCGImagePropertyMakerCanonFlashExposureComp;
CFStringRef  kCGImagePropertyMakerCanonContinuousDrive;
CFStringRef  kCGImagePropertyMakerCanonLensModel;
CFStringRef  kCGImagePropertyMakerCanonFirmware;
CFStringRef  kCGImagePropertyMakerCanonAspectRatioInfo;
```

Constants
`kCGImagePropertyMakerCanonOwnerName`
The owner name.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerCanonCameraSerialNumber`
The camera serial number.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerCanonImageSerialNumber`
The image serial number.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerCanonFlashExposureComp`
The flash exposure compensation.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerCanonContinuousDrive`
The presence of a continuous drive.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerCanonLensModel`
The lens model.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerCanonFirmware`
The firmware version.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

`kCGImagePropertyMakerCanonAspectRatioInfo`
The image aspect ratio.

Available in Mac OS X v10.5 and later.

Declared in `CGImageProperties.h`.

Declared In
`CGImageProperties.h`

CGAffineTransform Reference

Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGAffineTransform.h
Companion guide	Quartz 2D Programming Guide

Overview

The `CGAffineTransform` data structure represents a matrix used for affine transformations. A transformation specifies how points in one coordinate system map to points in another coordinate system. An affine transformation is a special type of mapping that preserves parallel lines in a path but does not necessarily preserve lengths or angles. Scaling, rotation, and translation are the most commonly used manipulations supported by affine transforms, but skewing is also possible.

Quartz provides functions that create, concatenate, and apply affine transformations using the `CGAffineTransform` data structure. For information on how to use affine transformation functions, see *Quartz 2D Programming Guide*.

You typically do not need to create an affine transform directly—*CGContext Reference* describes functions that modify the current affine transform. If you don't plan to reuse an affine transform, you may want to use `CGContextScaleCTM` (page 99), `CGContextRotateCTM` (page 98), `CGContextTranslateCTM` (page 130), or `CGContextConcatCTM` (page 76).

Functions by Task

Creating an Affine Transformation Matrix

`CGAffineTransformMake` (page 594)
Returns an affine transformation matrix constructed from values you provide.

`CGAffineTransformMakeRotation` (page 596)
Returns an affine transformation matrix constructed from a rotation value you provide.

`CGAffineTransformMakeScale` (page 596)
Returns an affine transformation matrix constructed from scaling values you provide.

`CGAffineTransformMakeTranslation` (page 597)
Returns an affine transformation matrix constructed from translation values you provide.

Modifying Affine Transformations

`CGAffineTransformTranslate` (page 600)
Returns an affine transformation matrix constructed by translating an existing affine transform.

`CGAffineTransformScale` (page 599)
Returns an affine transformation matrix constructed by scaling an existing affine transform.

`CGAffineTransformRotate` (page 598)
Returns an affine transformation matrix constructed by rotating an existing affine transform.

`CGAffineTransformInvert` (page 593)
Returns an affine transformation matrix constructed by inverting an existing affine transform.

`CGAffineTransformConcat` (page 592)
Returns an affine transformation matrix constructed by combining two existing affine transforms.

Applying Affine Transformations

`CGPointApplyAffineTransform` (page 600)
Returns the point resulting from an affine transformation of an existing point.

`CGSizeApplyAffineTransform` (page 601)
Returns the height and width resulting from a transformation of an existing height and width.

`CGRectApplyAffineTransform` (page 601)
Applies an affine transform to a rectangle.

Evaluating Affine Transforms

`CGAffineTransformIsIdentity` (page 594)
Checks whether an affine transform is the identity transform.

`CGAffineTransformEqualToTransform` (page 593)
Checks whether two affine transforms are equal.

Functions

CGAffineTransformConcat

Returns an affine transformation matrix constructed by combining two existing affine transforms.

```
CGAffineTransform CGAffineTransformConcat (
   CGAffineTransform t1,
   CGAffineTransform t2
);
```

Parameters

t1
The first affine transform.

t2

The second affine transform. This affine transform is concatenated to the first affine transform.

Return Value
A new affine transformation matrix. That is, t' = t1*t2.

Discussion
Concatenation combines two affine transformation matrices by multiplying them together. You might perform several concatenations in order to create a single affine transform that contains the cumulative effects of several transformations.

Note that matrix operations are not commutative—the order in which you concatenate matrices is important. That is, the result of multiplying matrix `t1` by matrix `t2` does not necessarily equal the result of multiplying matrix `t2` by matrix `t1`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGAffineTransform.h`

CGAffineTransformEqualToTransform

Checks whether two affine transforms are equal.

```
bool CGAffineTransformEqualToTransform (
   CGAffineTransform t1,
   CGAffineTransform t2
);
```

Parameters

t1

An affine transform.

t2

An affine transform.

Return Value
Returns `true` if `t1` and `t2` are equal, `false` otherwise.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGAffineTransform.h`

CGAffineTransformInvert

Returns an affine transformation matrix constructed by inverting an existing affine transform.

```
CGAffineTransform CGAffineTransformInvert (
   CGAffineTransform t
);
```

Parameters

t

An existing affine transform.

Return Value
A new affine transformation matrix. If the affine transform passed in parameter `t` cannot be inverted, Quartz returns the affine transform unchanged.

Discussion
Inversion is generally used to provide reverse transformation of points within transformed objects. Given the coordinates (x,y), which have been transformed by a given matrix to new coordinates (x′,y′), transforming the coordinates (x′,y′) by the inverse matrix produces the original coordinates (x,y).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGAffineTransform.h`

CGAffineTransformIsIdentity

Checks whether an affine transform is the identity transform.

```
bool CGAffineTransformIsIdentity (
   CGAffineTransform t
);
```

Parameters

t

The affine transform to check.

Return Value
Returns `true` if `t` is the identity transform, `false` otherwise.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGAffineTransform.h`

CGAffineTransformMake

Returns an affine transformation matrix constructed from values you provide.

```
CGAffineTransform CGAffineTransformMake (
   CGFloat a,
   CGFloat b,
   CGFloat c,
   CGFloat d,
   CGFloat tx,
   CGFloat ty
);
```

Parameters

a
The value at position [1,1] in the matrix.

b
The value at position [1,2] in the matrix.

c
The value at position [2,1] in the matrix.

d
The value at position [2,2] in the matrix.

tx
The value at position [3,1] in the matrix.

ty
The value at position [3,2] in the matrix.

Return Value
A new affine transform matrix constructed from the values you specify.

Discussion
This function creates a `CGAffineTransform` structure that represents a new affine transformation matrix, which you can use (and reuse, if you want) to transform a coordinate system. The matrix takes the following form:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Because the third column is always `(0,0,1)`, the `CGAffineTransform` data structure returned by this function contains values for only the first two columns.

If you want only to transform an object to be drawn, it is not necessary to construct an affine transform to do so. The most direct way to transform your drawing is by calling the appropriate `CGContext` function to adjust the current transformation matrix.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGAffineTransform.h`

CGAffineTransformMakeRotation

Returns an affine transformation matrix constructed from a rotation value you provide.

```
CGAffineTransform CGAffineTransformMakeRotation (
   CGFloat angle
);
```

Parameters

angle

The angle, in radians, by which this matrix rotates the coordinate system axes. A positive value specifies clockwise rotation, a negative value specifies counterclockwise.

Return Value

A new affine transformation matrix.

Discussion

This function creates a `CGAffineTransform` structure, which you can use (and reuse, if you want) to rotate a coordinate system. The matrix takes the following form:

$$\begin{bmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Because the third column is always `(0,0,1)`, the `CGAffineTransform` data structure returned by this function contains values for only the first two columns.

These are the resulting equations that Quartz uses to apply the rotation to a point (x, y):

$$x' = x\cos a - y\sin a$$
$$y' = x\sin a + y\cos a$$

If you want only to rotate an object to be drawn, it is not necessary to construct an affine transform to do so. The most direct way to rotate your drawing is by calling the function `CGContextRotateCTM` (page 98).

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGAffineTransform.h`

CGAffineTransformMakeScale

Returns an affine transformation matrix constructed from scaling values you provide.

```
CGAffineTransform CGAffineTransformMakeScale (
   CGFloat sx,
   CGFloat sy
);
```

Parameters

sx

The factor by which to scale the x-axis of the coordinate system.

sy

The factor by which to scale the y-axis of the coordinate system.

Return Value
A new affine transformation matrix.

Discussion
This function creates a `CGAffineTransform` structure, which you can use (and reuse, if you want) to scale a coordinate system. The matrix takes the following form:

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Because the third column is always `(0,0,1)`, the `CGAffineTransform` data structure returned by this function contains values for only the first two columns.

These are the resulting equations that Quartz uses to scale the coordinates of a point (x,y):

$$x' = x \cdot s_x$$
$$y' = y \cdot s_y$$

If you want only to scale an object to be drawn, it is not necessary to construct an affine transform to do so. The most direct way to scale your drawing is by calling the function `CGContextScaleCTM` (page 99).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGAffineTransform.h`

CGAffineTransformMakeTranslation

Returns an affine transformation matrix constructed from translation values you provide.

```
CGAffineTransform CGAffineTransformMakeTranslation (
   CGFloat tx,
   CGFloat ty
);
```

Parameters

tx

The value by which to move the x-axis of the coordinate system.

ty

The value by which to move the y-axis of the coordinate system.

Return Value
A new affine transform matrix.

Discussion
This function creates a `CGAffineTransform` structure. which you can use (and reuse, if you want) to move a coordinate system. The matrix takes the following form:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Because the third column is always `(0,0,1)`, the `CGAffineTransform` data structure returned by this function contains values for only the first two columns.

These are the resulting equations Quartz uses to apply the translation to a point (x,y):

$$x' = x + t_x$$
$$y' = y + t_y$$

If you want only to move the location where an object is drawn, it is not necessary to construct an affine transform to do so. The most direct way to move your drawing is by calling the function `CGContextTranslateCTM` (page 130).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGAffineTransform.h`

CGAffineTransformRotate

Returns an affine transformation matrix constructed by rotating an existing affine transform.

```
CGAffineTransform CGAffineTransformRotate (
   CGAffineTransform t,
   CGFloat angle
);
```

Parameters

t

An existing affine transform.

angle

The angle, in radians, by which to rotate the affine transform.

Return Value
A new affine transformation matrix.

Discussion
You use this function to create a new affine transformation matrix by adding a rotation value to an existing affine transform. The resulting structure represents a new affine transform, which you can use (and reuse, if you want) to rotate a coordinate system.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGAffineTransform.h`

CGAffineTransformScale

Returns an affine transformation matrix constructed by scaling an existing affine transform.

```
CGAffineTransform CGAffineTransformScale (
   CGAffineTransform t,
   CGFloat sx,
   CGFloat sy
);
```

Parameters

t

An existing affine transform.

sx

The value by which to scale x values of the affine transform.

sy

The value by which to scale y values of the affine transform.

Return Value
A new affine transformation matrix.

Discussion
You use this function to create a new affine transformation matrix by adding scaling values to an existing affine transform. The resulting structure represents a new affine transform, which you can use (and reuse, if you want) to scale a coordinate system.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
HID Calibrator

Declared In
`CGAffineTransform.h`

CGAffineTransformTranslate

Returns an affine transformation matrix constructed by translating an existing affine transform.

```
CGAffineTransform CGAffineTransformTranslate (
   CGAffineTransform t,
   CGFloat tx,
   CGFloat ty
);
```

Parameters

t

An existing affine transform.

tx

The value by which to move x values with the affine transform.

ty

The value by which to move y values with the affine transform.

Return Value
A new affine transformation matrix.

Discussion
You use this function to create a new affine transform by adding translation values to an existing affine transform. The resulting structure represents a new affine transform, which you can use (and reuse, if you want) to move a coordinate system.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGAffineTransform.h`

CGPointApplyAffineTransform

Returns the point resulting from an affine transformation of an existing point.

```
CGPoint CGPointApplyAffineTransform (
   CGPoint point,
   CGAffineTransform t
);
```

Parameters

point

A point that specifies the x- and y-coordinates to transform.

t

The affine transform to apply.

Return Value
A new point resulting from applying the specified affine transform to the existing point.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGAffineTransform.h`

CGRectApplyAffineTransform

Applies an affine transform to a rectangle.

```
CGRect CGRectApplyAffineTransform (
   CGRect rect,
   CGAffineTransform t
);
```

Parameters

rect

The rectangle whose corner points you want to transform.

t

The affine transform to apply to the `rect` parameter.

Return Value
The transformed rectangle.

Discussion
Because affine transforms do not preserve rectangles in general, the function `CGRectApplyAffineTransform` returns the smallest rectangle that contains the transformed corner points of the `rect` parameter. If the affine transform `t` consists solely of scaling and translation operations, then the returned rectangle coincides with the rectangle constructed from the four transformed corners.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CGAffineTransform.h`

CGSizeApplyAffineTransform

Returns the height and width resulting from a transformation of an existing height and width.

```
CGSize CGSizeApplyAffineTransform (
   CGSize size,
   CGAffineTransform t
);
```

Parameters

size

A size that specifies the height and width to transform.

t

The affine transform to apply.

Return Value
A new size resulting from applying the specified affine transform to the existing size.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGAffineTransform.h`

Data Types

CGAffineTransform

A structure for holding an affine transformation matrix.

```
struct CGAffineTransform {
    CGFloat a;
    CGFloat b;
    CGFloat c;
    CGFloat d;
    CGFloat tx;
    CGFloat ty;
};
typedef struct CGAffineTransform CGAffineTransform;
```

Fields

`a`
The entry at position [1,1] in the matrix.

`b`
The entry at position [1,2] in the matrix.

`c`
The entry at position [2,1] in the matrix.

`d`
The entry at position [2,2] in the matrix.

`tx`
The entry at position [3,1] in the matrix.

`ty`
The entry at position [3,2] in the matrix.

Discussion
In Quartz 2D, an affine transformation matrix is used to rotate, scale, translate, or skew the objects you draw in a graphics context. The `CGAffineTransform` type provides functions for creating, concatenating, and applying affine transformations.

In Quartz, affine transforms are represented by a 3 by 3 matrix:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Because the third column is always `(0,0,1)`, the `CGAffineTransform` data structure contains values for only the first two columns.

Conceptually, a Quartz affine transform multiplies a row vector representing each point (x,y) in your drawing by this matrix, producing a vector that represents the corresponding point (x′,y′):

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \times \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Given the 3 by 3 matrix, Quartz uses the following equations to transform a point (x, y) in one coordinate system into a resultant point (x′,y′) in another coordinate system.

$$x' = ax + cy + t_x$$
$$y' = bx + dy + t_y$$

The matrix thereby "links" two coordinate systems—it specifies how points in one coordinate system map to points in another.

Note that you do not typically need to create affine transforms directly. If you want only to draw an object that is scaled or rotated, for example, it is not necessary to construct an affine transform to do so. The most direct way to manipulate your drawing—whether by movement, scaling, or rotation—is to call the functions `CGContextTranslateCTM` (page 130), `CGContextScaleCTM` (page 99), or `CGContextRotateCTM` (page 98), respectively. You should generally only create an affine transform if you want to reuse it later.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGAffineTransform.h`

Constants

CGAffineTransformIdentity

The identity transform.

```
const CGAffineTransform CGAffineTransformIdentity;
```

Constants

`CGAffineTransformIdentity`

The identity transform: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Available in Mac OS X v10.0 and later.

Declared in `CGAffineTransform.h`.

Declared In

`CGAffineTransform.h`

CGGeometry Reference

Framework:	ApplicationServices/ApplicationServices.h
Declared in	CGGeometry.h
Companion guide	Quartz 2D Programming Guide

Overview

CGGeometry Reference defines structures for geometric primitives and functions that operate on them. The data structure `CGPoint` represents a point in a two-dimensional coordinate system. The data structure `CGRect` represents the location and dimensions of a rectangle. The data structure `CGSize` represents the dimensions of width and height.

Functions by Task

Creating a Geometric Primitive From a Dictionary Representation

`CGPointCreateDictionaryRepresentation` (page 607)
Returns a dictionary representation of the provided point.

`CGSizeCreateDictionaryRepresentation` (page 623)
Returns a dictionary representation of the provided size.

`CGRectCreateDictionaryRepresentation` (page 610)
Returns a dictionary representation of the provided rectangle.

Creating a Dictionary Representation From a Geometric Primitive

`CGPointMakeWithDictionaryRepresentation` (page 609)
Fills in a `CGPoint` structure using the contents of the provided dictionary.

`CGSizeMakeWithDictionaryRepresentation` (page 624)
Fills in a `CGSize` structure using the contents of the provided dictionary.

`CGRectMakeWithDictionaryRepresentation` (page 621)
Fills in a `CGRect` structure using the contents of the provided dictionary.

Creating a Geometric Primitive From Values

`CGPointMake` (page 608)
Returns a `CGPoint` structure filled in with the coordinate values you provide.

`CGRectMake` (page 620)
Returns a `CGRect` structure filled in with the coordinate and dimension values you provide.

`CGSizeMake` (page 624)
Returns a `CGSize` structure filled in with dimension values you provide.

Modifying Rectangles

`CGRectDivide` (page 611)
Divides a source rectangle into two component rectangles.

`CGRectInset` (page 616)
Returns a rectangle that is smaller or larger than the source rectangle, with the same center point.

`CGRectIntegral` (page 617)
Returns the smallest rectangle that results from converting the source rectangle values to integers.

`CGRectIntersection` (page 617)
Returns the intersection of two rectangles.

`CGRectOffset` (page 621)
Returns a rectangle with an origin that is offset from that of the source rectangle.

`CGRectStandardize` (page 622)
Returns a rectangle with a positive width and height.

`CGRectUnion` (page 623)
Returns the smallest rectangle that contains the two provided rectangles.

Comparing Values

`CGPointEqualToPoint` (page 608)
Returns whether two points are equal.

`CGSizeEqualToSize` (page 623)
Returns whether two sizes are equal.

`CGRectEqualToRect` (page 611)
Returns whether two rectangles are equal in size and position.

`CGRectIntersectsRect` (page 618)
Returns whether two rectangles intersect.

Checking for Membership

`CGRectContainsPoint` (page 609)
Returns whether a rectangle contains a specified point.

`CGRectContainsRect` (page 610)
Returns whether the first rectangle contains the second rectangle.

Getting Min, Mid, and Max Values

`CGRectGetMinX` (page 614)
Returns the x-coordinate that establishes the left edge of a rectangle.

`CGRectGetMinY` (page 615)
Returns the y-coordinate that establishes the bottom edge of a rectangle.

`CGRectGetMidX` (page 613)
Returns the x- coordinate that establishes the center of a rectangle.

`CGRectGetMidY` (page 614)
Returns the y-coordinate that establishes the center of a rectangle.

`CGRectGetMaxX` (page 612)
Returns the x-coordinate that establishes the right edge of a rectangle.

`CGRectGetMaxY` (page 613)
Returns the y-coordinate that establishes the top edge of a rectangle.

Getting Height and Width

`CGRectGetHeight` (page 612)
Returns the height of a rectangle.

`CGRectGetWidth` (page 615)
Returns the width of a rectangle.

Checking Rectangle Characteristics

`CGRectIsEmpty` (page 618)
Returns whether a rectangle has zero width or height, or is a null rectangle.

`CGRectIsNull` (page 620)
Returns whether a rectangle is invalid.

`CGRectIsInfinite` (page 619)
Returns whether a rectangle is infinite.

`CGRectIsIntegral` (page 619)
Returns whether the origin and size of the rectangle can be represented exactly as integers.

Functions

CGPointCreateDictionaryRepresentation

Returns a dictionary representation of the provided point.

```
CFDictionaryRef CGPointCreateDictionaryRepresentation(
    CGPoint point
);
```

Parameters

point

A point.

Return Value
The dictionary representation of the point.

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

Declared In
`CGGeometry.h`

CGPointEqualToPoint

Returns whether two points are equal.

```
bool CGPointEqualToPoint (
   CGPoint point1,
   CGPoint point2
);
```

Parameters

point1

The first point to examine.

point2

The second point to examine.

Return Value
Returns `1` if the two specified points are the same; otherwise, `0`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGGeometry.h`

CGPointMake

Returns a `CGPoint` structure filled in with the coordinate values you provide.

```
CGPoint CGPointMake (
   CGFloat x,
   CGFloat y
);
```

Parameters

x

The x-coordinate of the point to construct.

y
The y-coordinate of the point to construct.

Return Value
Returns a `CGPoint` structure, representing a single (x,y) coordinate pair.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CALayerEssentials
CarbonSketch

Declared In
`CGGeometry.h`

CGPointMakeWithDictionaryRepresentation

Fills in a `CGPoint` structure using the contents of the provided dictionary.

```
bool CGPointMakeWithDictionaryRepresentation(
    CFDictionaryRef dict,
    CGPoint *point
);
```

Parameters
dict
A dictionary that was previously returned from the function `CGPointCreateDictionaryRepresentation` (page 607).

point
On return, the point created from the provided dictionary.

Return Value
`true` if successful; `false` otherwise.

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

Declared In
`CGGeometry.h`

CGRectContainsPoint

Returns whether a rectangle contains a specified point.

```
bool CGRectContainsPoint (
   CGRect rect,
   CGPoint point
);
```

Parameters
rect
The rectangle to examine.

point
The point to examine.

Return Value
Returns `1` if the specified point is located within the specified rectangle; otherwise, `0`.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGGeometry.h`

CGRectContainsRect

Returns whether the first rectangle contains the second rectangle.

```
bool CGRectContainsRect (
   CGRect rect1,
   CGRect rect2
);
```

Parameters
rect1
The rectangle to examine for containment of the rectangle passed in `rect2`.

rect2
The rectangle to examine for being contained in the rectangle passed in `rect1`.

Return Value
Returns `1` if the rectangle specified by `rect2` is contained in the rectangle passed in `rect1`; otherwise, `0`. The first rectangle contains the second if the union of the two rectangles is equal to the first rectangle.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGGeometry.h`

CGRectCreateDictionaryRepresentation

Returns a dictionary representation of the provided rectangle.

```
CFDictionaryRef CGRectCreateDictionaryRepresentation(
    CGRect rect
);
```

Parameters
rect
A rectangle.

Return Value
The dictionary representation of the rectangle.

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

Declared In
`CGGeometry.h`

CGRectDivide

Divides a source rectangle into two component rectangles.

```
void CGRectDivide (
   CGRect rect,
   CGRect *slice,
   CGRect *remainder,
   CGFloat amount,
   CGRectEdge edge
);
```

Parameters

rect
The source `CGRect` structure.

slice
On input, a pointer to an uninitialized `CGRect` structure. On return, a `CGRect` structure filled in with the specified edge and values that extends the distance beyond the edge specified by the *amount* parameter.

remainder
On input, a pointer to an uninitialized rectangle `CGRect` structure. On return, the `CGRect` structure contains the portion of the source `CGRect` structure that remains after `CGRectEdge` produces the "slice" rectangle.

amount
A distance from the rectangle side that is specified in the *edge* parameter. This distance defines the line, parallel to the specified side, that Quartz uses to divide the source `CGRect` structure.

edge
A `CGRectEdge` value (`CGRectMinXEdge` (page 628), `CGRectMinYEdge` (page 628), `CGRectMaxXEdge` (page 628), or `CGRectMaxYEdge` (page 628)) that specifies the side of the rectangle from which the distance passed in the `amount` parameter is measured. `CGRectDivide` produces a "slice" rectangle that contains the specified edge and extends `amount` distance beyond it.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGGeometry.h`

CGRectEqualToRect

Returns whether two rectangles are equal in size and position.

```
bool CGRectEqualToRect (
   CGRect rect1,
   CGRect rect2
);
```

Parameters

rect1

The first rectangle to examine.

rect2

The second rectangle to examine.

Return Value

Returns `1` if the two specified rectangles have equal size and origin values, or are both `null`. Otherwise, returns `0`.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGGeometry.h`

CGRectGetHeight

Returns the height of a rectangle.

```
CGFloat CGRectGetHeight (
   CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value

The height of the specified rectangle.

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

CarbonSketch

HID Calibrator

HID Config Save

HID Explorer

WhackedTV

Declared In

`CGGeometry.h`

CGRectGetMaxX

Returns the x-coordinate that establishes the right edge of a rectangle.

```
CGFloat CGRectGetMaxX (
   CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value
The x-coordinate of the top-right corner of the specified rectangle.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
HID Calibrator
HID Explorer

Declared In
CGGeometry.h

CGRectGetMaxY

Returns the y-coordinate that establishes the top edge of a rectangle.

```
CGFloat CGRectGetMaxY (
   CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value
The y-coordinate of the top-right corner of the specified rectangle.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
HID Explorer

Declared In
CGGeometry.h

CGRectGetMidX

Returns the x- coordinate that establishes the center of a rectangle.

```
CGFloat CGRectGetMidX (
   CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value
The x-coordinate of the center of the specified rectangle.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
HID Calibrator

Declared In
`CGGeometry.h`

CGRectGetMidY

Returns the y-coordinate that establishes the center of a rectangle.

```
CGFloat CGRectGetMidY (
   CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value
The y-coordinate of the center of the specified rectangle.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
HID Calibrator
HID Explorer

Declared In
`CGGeometry.h`

CGRectGetMinX

Returns the x-coordinate that establishes the left edge of a rectangle.

```
CGFloat CGRectGetMinX (
   CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value
The x-coordinate of the bottom-left corner of the specified rectangle.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch
HID Config Save
HID Explorer

Declared In
CGGeometry.h

CGRectGetMinY

Returns the y-coordinate that establishes the bottom edge of a rectangle.

```
CGFloat CGRectGetMinY (
   CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value
The y-coordinate of the bottom-left corner of the specified rectangle.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch
HID Config Save
HID Explorer

Declared In
CGGeometry.h

CGRectGetWidth

Returns the width of a rectangle.

```
CGFloat CGRectGetWidth (
   CGRect rect
);
```

Parameters

rect

The rectangle to examine.

Return Value
The width of the specified rectangle.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch
HID Calibrator
HID Config Save
HID Explorer
WhackedTV

Declared In
`CGGeometry.h`

CGRectInset

Returns a rectangle that is smaller or larger than the source rectangle, with the same center point.

```
CGRect CGRectInset (
   CGRect rect,
   CGFloat dx,
   CGFloat dy
);
```

Parameters

rect

The source `CGRect` structure.

dx

The x-coordinate value to use for adjusting the source rectangle. To create an inset rectangle, specify a positive value. To create a larger, encompassing rectangle, specify a negative value.

dy

The y-coordinate value to use for adjusting the source rectangle. To create an inset rectangle, specify a positive value. To create a larger, encompassing rectangle, specify a negative value.

Return Value
A filled-in `CGRect` structure. The origin value is offset in the x-axis by the distance specified by the `dx` parameter and in the y-axis by the distance specified by the `dy` parameter, and its size adjusted by `(2*dx,2*dy)`, relative to the source rectangle. If `dx` and `dy` are positive values, then the rectangle's size is decreased. If `dx` and `dy` are negative values, the rectangle's size is increased.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGGeometry.h`

CGRectIntegral

Returns the smallest rectangle that results from converting the source rectangle values to integers.

```
CGRect CGRectIntegral (
   CGRect rect
);
```

Parameters

rect
The source rectangle.

Return Value
A filled-in `CGRect` structure whose values represent the rectangle with the smallest integer values for its origin and size that contains the source rectangle. That is, given a rectangle with fractional origin or size values, `CGRectIntegral` rounds the rectangle's origin downward and its size upward to the nearest whole integers, such that the result contains the original rectangle.

Availability
Available in Mac OS X version 10.0 and later.

See Also
`CGRectIsIntegral` (page 619)

Related Sample Code
WhackedTV

Declared In
`CGGeometry.h`

CGRectIntersection

Returns the intersection of two rectangles.

```
CGRect CGRectIntersection (
   CGRect r1,
   CGRect r2
);
```

Parameters

rect1
The first source rectangle.

rect2
The second source rectangle.

Return Value
A filled-in `CGRect` structure that represents the intersection of the two specified rectangles. If the two rectangles do not intersect, returns the null rectangle. To check for this condition, use `CGRectIsNull` (page 620).

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
WhackedTV

Declared In
`CGGeometry.h`

CGRectIntersectsRect

Returns whether two rectangles intersect.

```
bool CGRectIntersectsRect (
   CGRect rect1,
   CGRect rect2
);
```

Parameters

rect1
The first rectangle to examine.

rect2
The second rectangle to examine.

Return Value
Returns `1` if the two specified rectangles intersect; otherwise, `0`. The first rectangle intersects the second if the intersection of the rectangles is not equal to the null rectangle.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGGeometry.h`

CGRectIsEmpty

Returns whether a rectangle has zero width or height, or is a null rectangle.

```
bool CGRectIsEmpty (
   CGRect rect
);
```

Parameters

rect
The rectangle to examine.

Return Value
Returns `1` if the specified rectangle is empty; otherwise, `0`.

Discussion
An empty rectangle is either a null rectangle or a valid rectangle with zero height or width. See also `CGRectIsNull` (page 620).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGGeometry.h`

CGRectIsInfinite

Returns whether a rectangle is infinite.

```
bool CGRectIsInfinite (
    CGRect rect
);
```

Parameters

rect
The rectangle to examine.

Return Value
Returns `true` if the specified rectangle is infinite, `false` otherwise.

Discussion
An infinite rectangle is one that has no defined bounds. Infinite rectangles can be created as output from a tiling filter. For example, the Core Image framework perspective tile filter creates an image whose extent is described by an infinite rectangle.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
WhackedTV

Declared In
`CGGeometry.h`

CGRectIsIntegral

Returns whether the origin and size of the rectangle can be represented exactly as integers.

```
bool CGRectIsIntegral (
   CGRect rect
);
```

Parameters

rect
The rectangle to examine.

Return Value
Returns `true` if the origin and size of the rectangle can be represented exactly as integers; `false` otherwise.

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

Declared In
`CGGeometry.h`

CGRectIsNull

Returns whether a rectangle is invalid.

```
bool CGRectIsNull (
   CGRect rect
);
```

Parameters

rect
The rectangle to examine.

Return Value
Returns `1` if the specified rectangle is null; otherwise, `0`.

Discussion
A null rectangle is one that is not valid (you cannot draw a null rectangle). For example, the result of intersecting two disjoint rectangles is a null rectangle. See also `CGRectIsEmpty` (page 618).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGGeometry.h`

CGRectMake

Returns a `CGRect` structure filled in with the coordinate and dimension values you provide.

```
CGRect CGRectMake (
   CGFloat x,
   CGFloat y,
   CGFloat width,
   CGFloat height
);
```

Parameters

x
The x-coordinate of the rectangle's origin point.

y
The y-coordinate of the rectangle's origin point.

width
The width of the rectangle.

height
The height of the rectangle.

Return Value
Returns a rectangle with the specified location and dimensions.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CALayerEssentials
CarbonSketch
HID Calibrator
HID Explorer
QTCarbonShell

Declared In
`CGGeometry.h`

CGRectMakeWithDictionaryRepresentation

Fills in a `CGRect` structure using the contents of the provided dictionary.

```
bool CGRectMakeWithDictionaryRepresentation(
    CFDictionaryRef dict,
    CGRect *rect
);
```

Parameters

dict
A dictionary that was previously returned from the function `CGRectCreateDictionaryRepresentation` (page 610).

rect
On return, the rectangle created from the provided dictionary.

Return Value
`true` if successful; `false` otherwise.

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

Declared In
`CGGeometry.h`

CGRectOffset

Returns a rectangle with an origin that is offset from that of the source rectangle.

```
CGRect CGRectOffset (
   CGRect rect,
   CGFloat dx,
   CGFloat dy
);
```

Parameters

rect

The source rectangle.

dx

The offset value for the x-coordinate.

dy

The offset value for the y-coordinate.

Return Value
A filled-in `CGRect` structure that is the same size as the source, but with its origin offset by `dx` units along the x-axis and `dy` units along the y-axis with respect to the source.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGGeometry.h`

CGRectStandardize

Returns a rectangle with a positive width and height.

```
CGRect CGRectStandardize (
   CGRect rect
);
```

Parameters

rect

The source rectangle.

Return Value
A filled-in `CGRect` structure that represents the source rectangle, but with positive width and height values.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGGeometry.h`

CGRectUnion

Returns the smallest rectangle that contains the two provided rectangles.

```
CGRect CGRectUnion (
   CGRect r1,
   CGRect r2
);
```

Parameters

r1

The first source rectangle.

r2

The second source rectangle.

Return Value

A filled-in `CGRect` structure that represents the smallest rectangle that completely contains both of the source rectangles.

Discussion

If one of the rectangles has 0 (or negative) width or height, a copy of the other rectangle is returned; but if both have 0 (or negative) width or height, the returned rectangle has its origin at (0.0, 0.0) and has 0 width and height.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`CGGeometry.h`

CGSizeCreateDictionaryRepresentation

Returns a dictionary representation of the provided size.

```
CFDictionaryRef CGSizeCreateDictionaryRepresentation(
    CGSize size
);
```

Parameters

size

A size.

Return Value

The dictionary representation of the size.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CGGeometry.h`

CGSizeEqualToSize

Returns whether two sizes are equal.

```
bool CGSizeEqualToSize (
   CGSize size1,
   CGSize size2
);
```

Parameters

size1

The first size to examine.

size2

The second size to examine.

Return Value
Returns `1` if the two specified sizes are equal; otherwise, `0`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`CGGeometry.h`

CGSizeMake

Returns a `CGSize` structure filled in with dimension values you provide.

```
CGSize CGSizeMake (
   CGFloat width,
   CGFloat height
);
```

Parameters

width

A width value.

height

A height value.

Return Value
Returns a `CGSize` structure with the specified width and height.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`CGGeometry.h`

CGSizeMakeWithDictionaryRepresentation

Fills in a `CGSize` structure using the contents of the provided dictionary.

```
bool CGSizeMakeWithDictionaryRepresentation(
    CFDictionaryRef dict,
    CGSize *size
);
```

Parameters

dict

A dictionary that was previously returned from the function `CGSizeCreateDictionaryRepresentation` (page 623).

size

On return, the size created from the provided dictionary.

Return Value

`true` if successful; `false` otherwise.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CGGeometry.h`

Data Types

CGPoint

A structure that contains a point in a two-dimensional coordinate system.

```
struct CGPoint {
     CGFloat x;
    CGFloat y;
};
typedef struct CGPoint CGPoint;
```

Fields

`x`

The x-coordinate of the point.

`y`

The y-coordinate of the point.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CGGeometry.h`

CGRect

A structure that contains the location and dimensions of a rectangle.

```
struct CGRect {
    CGPoint origin;
    CGSize size;
};
typedef struct CGRect CGRect;
```

Fields

`origin`
A `CGPoint` (page 625) structure that specifies the coordinates of the rectangle's origin. The origin is located in the lower-left of the rectangle.

`size`
A `CGSize` (page 626) structure that specifies the height and width of the rectangle.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGGeometry.h`

CGSize

A structure that contains width and height values.

```
struct CGSize {
    CGFloat width;
    CGFloat height;
};
typedef struct CGSize CGSize;
```

Fields

`width`
A width value.

`height`
A height value.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CGGeometry.h`

Constants

CGRectInfinite

A rectangle that has infinite extent.

```
const CGRect CGRectInfinite;
```

Constants

`CGRectInfinite`

A rectangle that has infinite extent.

Available in Mac OS X v10.4 and later.

Declared in `CGGeometry.h`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CGGeometry.h`

Geometric Zeroes

A zero point, zero rectangle, or zero size.

```
const CGPoint CGPointZero;
const CGRect CGRectZero;
const CGSize CGSizeZero;
```

Constants

`CGPointZero`

A point constant with location `(0, 0)`. The zero point is equivalent to `CGPointMake(0,0)`.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

`CGRectZero`

A rectangle constant with location `(0,0)`, and width and height of 0. The zero rectangle is equivalent to `CGRectMake(0,0,0,0)`.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

`CGSizeZero`

A size constant with width and height of *0*. The zero size is equivalent to `CGSizeMake(0,0)`.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

Declared In

`CGGeometry.h`

Geometrical Null

The null or empty rectangle.

```
const CGRect CGRectNull;
```

Constants

`CGRectNull`

The null rectangle. This is the rectangle returned when, for example, you intersect two disjoint rectangles. Note that the null rectangle is not the same as the zero rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

Declared In

`CGGeometry.h`

CGRectEdge

Coordinates that establish the edges of a rectangle.

```
enum CGRectEdge {
    CGRectMinXEdge,
    CGRectMinYEdge,
    CGRectMaxXEdge,
    CGRectMaxYEdge
};
typedef enum CGRectEdge CGRectEdge;
```

Constants

`CGRectMinXEdge`

The x-coordinate that establishes the left edge of a rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

`CGRectMinYEdge`

The y-coordinate that establishes the minimum edge of a rectangle. In Mac OS X, this is typically the bottom edge of the rectangle. If the coordinate system is flipped (or if you are using the default coordinate system in iPhone OS), this constant refers to the top edge of the rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

`CGRectMaxXEdge`

The x-coordinate that establishes the right edge of a rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

`CGRectMaxYEdge`

The y-coordinate that establishes the maximum edge of a rectangle. In Mac OS X, this is typically the top edge of the rectangle. If the coordinate system is flipped (or if you are using the default coordinate system in iPhone OS), this constant refers to the bottom edge of the rectangle.

Available in Mac OS X v10.0 and later.

Declared in `CGGeometry.h`.

Declared In

`CGGeometry.h`

CGFloat Informational Macros

Informational macros for the `CGFloat` type.

```
#define CGFLOAT_MIN FLT_MIN                                     // 32-bit
#define CGFLOAT_MAX FLT_MAX
#define CGFLOAT_IS_DOUBLE 0

#define CGFLOAT_MIN DBL_MIN                                     // 64-bit
#define CGFLOAT_MAX DBL_MAX
#define CGFLOAT_IS_DOUBLE 1
```

Constants

`CGFLOAT_MIN`

The minimum allowable value for a `CGFloat` type. For 32-bit code, this value is `1.17549435e-38F`. For 64-bit code, it is `2.2250738585072014e-308`.

Available in Mac OS X v10.5 and later.

Declared in `CABase.h`.

`CGFLOAT_MAX`

The maximum allowable value for a `CGFloat` type. For 32-bit code, this value is `3.40282347e+38F`. For 64-bit code, it is `1.7976931348623157e+308`.

Available in Mac OS X v10.5 and later.

Declared in `CABase.h`.

`CGFLOAT_IS_DOUBLE`

Indicates whether `CGFloat` is defined as a `float` or `double` type.

Available in Mac OS X v10.5 and later.

Declared in `CABase.h`.

Document Revision History

This table describes the changes to *Core Graphics Reference Collection*.

Date	Notes
2006-12-11	Updated with new documents for Mac OS X v10.5.
Date	**Notes**
$RevisionDate_1.0	$RevisionSummary_1.0

Document Revision History

Index

Numerals

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

R

S

T

W