# Font Manager Reference

Carbon > Text & Fonts

2007-12-11

# Contents

# Font Manager Reference

| | |
|---|---|
| **Framework:** | ApplicationServices/ApplicationServices.h |
| **Declared in** | Fonts.h |

## Overview

As of Mac OS X version 10.5, all Font Manager functions but three (`FMFontGetCGFontRefFromFontFamilyInstance` (page 11), `FMGetATSFontRefFromFont` (page 12), and `FMGetFontFromATSFontRef` (page 13)) are deprecated. Most were deprecated in Mac OS X v10.4. The Font Manager was the font management API for the QuickDraw framework, which is now deprecated.

There are several alternatives that provide better compatibility with the rest of Mac OS X than using QuickDraw font functions. You should consider the following:

- For drawing and measuring text, use Core Text on Mac OS X v10.5 and later to render text directly through a Quartz (Core Image) graphics context. See *Core Text Programming Guide* and *Core Text Reference Collection*. On Mac OS X v10.4 and earlier, you can use the Appearance Manager API or the ATSUI API. See *Appearance Manager Reference*, *ATSUI Programming Guide*, and *ATSUI Reference*.
- For accessing information on fonts tracked by the operating system, use Core Text on Mac OS X v10.5 and later. See *Core Text Programming Guide* and *Core Text Reference Collection*. On Mac OS X v10.4 and earlier, use the ATS for Fonts API. See *Apple Type Services for Fonts Programming Guide* and *Apple Type Services for Fonts Reference*.
- For accessing and modifying information on fonts in a Quartz graphics context, use the Quartz API. See *Quartz 2D Programming Guide* and *Quartz 2D Reference Collection*.

The Font Manager API was used to manage the fonts your application uses to display and print text. The Font Manager was used to determine the characteristics of a font, change certain font settings, favor outline fonts over bitmapped fonts, and manipulate fonts in memory.

## Functions by Task

### Activating and Deactivating Fonts

`FMGetGeneration` (page 71) Deprecated in Mac OS X v10.5
    Retrieves the value of the generation count. (Deprecated. Use `ATSGetGeneration` instead.)

`FMActivateFonts` (page 37) Deprecated in Mac OS X v10.4
    Activates one or more fonts. (Deprecated. Use `ATSFontActivateFromFileReference` instead.)

`FMDeactivateFonts` (page 41) Deprecated in Mac OS X v10.4
: Deactivates one or more fonts. (Deprecated. Use `ATSFontDeactivate` instead.)

## Accessing Font Objects

`FMGetFontContainer` (page 44) Deprecated in Mac OS X v10.4
: Obtains the file that contains data for a font. (Deprecated. Use `ATSFontGetContainer` instead.)

`FMGetFontFormat` (page 50) Deprecated in Mac OS X v10.4
: Obtains the format identifier of a font.

`FMGetFontGeneration` (page 51) Deprecated in Mac OS X v10.4
: Obtains the generation count of a font. (Deprecated. Use `ATSFontGetGeneration` instead.)

`FMGetFontTable` (page 52) Deprecated in Mac OS X v10.4
: Retrieves all or part of a data table for a font. (Deprecated. Use `ATSFontGetTable` or `CTFontCopyTable` instead.)

`FMGetFontTableDirectory` (page 53) Deprecated in Mac OS X v10.4
: Obtains the table directory for a font.

## Accessing Font Containers

`FMGetFontContainerFromFontFamilyInstance` (page 45) Deprecated in Mac OS X v10.4
: Obtains the font container associated with a font family instance. (Deprecated. Use `ATSFontGetContainer` instead.)

`FMGetFontFamilyResource` (page 48) Deprecated in Mac OS X v10.4
: Obtains the font family resource for a font family. (Deprecated. Use `ATSFontGetFontFamilyResource` instead.)

## Accessing Font Family Objects

`FMGetFontFamilyFromName` (page 46) Deprecated in Mac OS X v10.4
: Returns the font family reference associated with a standard QuickDraw name. (Deprecated. Use `ATSFontFamilyFindFromName` instead.)

`FMGetFontFamilyGeneration` (page 46) Deprecated in Mac OS X v10.4
: Obtains the generation count of a font family. (Deprecated. Use `ATSFontFamilyGetGeneration` instead.)

`FMGetFontFamilyName` (page 47) Deprecated in Mac OS X v10.4
: Obtains the font family name associated with a font family reference. (Deprecated. Use `ATSFontFamilyGetName` instead.)

`FMGetFontFamilyTextEncoding` (page 49) Deprecated in Mac OS X v10.4
: Obtains the text encoding used by a font family. (Deprecated. Use `ATSFontFamilyGetEncoding` instead.)

## Enumerating Font Data

`FMCreateFontFamilyInstanceIterator` (page 38) Deprecated in Mac OS X v10.4
: Creates a font family instance iterator that your application can use to access the member fonts associated with a font family.

`FMCreateFontFamilyIterator` (page 39) Deprecated in Mac OS X v10.4
: Creates a font family iterator that your application can use to access font family objects. (Deprecated. Use `ATSFontFamilyIteratorCreate` instead.)

`FMCreateFontIterator` (page 40) Deprecated in Mac OS X v10.4
: Creates an iterator that your application can use to access fonts. (Deprecated. Use `ATSFontIteratorCreate` instead.)

`FMDisposeFontFamilyInstanceIterator` (page 42) Deprecated in Mac OS X v10.4
: Disposes of a font family instance iterator.

`FMDisposeFontFamilyIterator` (page 42) Deprecated in Mac OS X v10.4
: Disposes of the contents of a font family iterator. (Deprecated. Use `ATSFontFamilyIteratorRelease` instead.)

`FMDisposeFontIterator` (page 43) Deprecated in Mac OS X v10.4
: Disposes of a font iterator. (Deprecated. Use `ATSFontIteratorRelease` instead.)

`FMGetNextFont` (page 53) Deprecated in Mac OS X v10.4
: Obtains the next font reference. (Deprecated. Use `ATSFontIteratorNext` instead.)

`FMGetNextFontFamily` (page 54) Deprecated in Mac OS X v10.4
: Obtains the next font family reference. (Deprecated. Use `ATSFontFamilyIteratorNext` instead.)

`FMGetNextFontFamilyInstance` (page 55) Deprecated in Mac OS X v10.4
: Obtains the next instance associated with a font family reference.

`FMResetFontFamilyInstanceIterator` (page 56) Deprecated in Mac OS X v10.4
: Resets the a font family instance iterator to the beginning of the iteration for the specified font family.

`FMResetFontFamilyIterator` (page 56) Deprecated in Mac OS X v10.4
: Resets a font family iterator to the beginning of the iteration. (Deprecated. Use `ATSFontFamilyIteratorReset` instead.)

`FMResetFontIterator` (page 57) Deprecated in Mac OS X v10.4
: Resets a font iterator to the beginning of the iteration. (Deprecated. Use `ATSFontIteratorReset` instead.)

## Converting Font Data

`FMFontGetCGFontRefFromFontFamilyInstance` (page 11)
: Obtains the Quartz font associated with a typeface from a QuickDraw font family.

`FMGetATSFontRefFromFont` (page 12)
: Obtains the ATS font reference associated with a font object.

`FMGetFontFromATSFontRef` (page 13)
: Obtains the font object associated with an ATS font reference.

`FMGetATSFontFamilyRefFromFontFamily` (page 43) Deprecated in Mac OS X v10.4
: Obtains the ATS font family reference associated with a font family object.

`FMGetFontFamilyFromATSFontFamilyRef` (page 45) Deprecated in Mac OS X v10.4
: Obtains the font family associated with an ATS font family reference.

`FMGetFontFamilyInstanceFromFont` (page 47) Deprecated in Mac OS X v10.4
: Finds the font family reference and standard QuickDraw style associated with a font.

`FMGetFontFromFontFamilyInstance` (page 51) Deprecated in Mac OS X v10.4
: Obtains the font reference associated with a standard QuickDraw style and font family. (Deprecated. Use `CTFontCreateWithQuickdrawInstance` instead.)

## Getting Font Information

`FetchFontInfo` (page 37) Deprecated in Mac OS X v10.4
: Obtains the information for a specific font. (Deprecated. There is no replacement function.)

`FMSwapFont` (page 58) Deprecated in Mac OS X v10.4
: Returns a pointer to the font output structure for a specified font. (Deprecated. There is no replacement function.)

`FontMetrics` (page 59) Deprecated in Mac OS X v10.4
: Obtains fractional measurements for the font, size, and style specified in the current graphics port. (Deprecated. There is no replacement function.)

`GetFNum` (page 60) Deprecated in Mac OS X v10.4
: Obtains the font family ID for a specified font family name. (Deprecated. Use `ATSFontFamilyFindFromName` instead.)

`GetFontName` (page 61) Deprecated in Mac OS X v10.4
: Obtains the name of a font family that has a specified family ID number. (Deprecated. Use `ATSFontFamilyGetName` instead.)

`OutlineMetrics` (page 64) Deprecated in Mac OS X v10.4
: Obtains font measurements for a block of text to be drawn in a specified outline font. (Deprecated. There is no replacement function.)

`RealFont` (page 67) Deprecated in Mac OS X v10.4
: Determines whether a font is available or is intended for use in a specified size. (Deprecated. There is no replacement function.)

## Working With Outline Fonts

`GetOutlinePreferred` (page 62) Deprecated in Mac OS X v10.4
: Obtains the current preference for whether outline or bitmapped fonts are returned when the Font Manager receives a font request. (Deprecated. There is no replacement function.)

`GetPreserveGlyph` (page 62) Deprecated in Mac OS X v10.4
: Determines whether the Font Manager preserves the shapes of glyphs from outline fonts. (Deprecated. There is no replacement function.)

`IsOutline` (page 63) Deprecated in Mac OS X v10.4
: Determines whether the specified scaling factors will cause the Font Manager to choose an outline font for the current graphics port. (Deprecated. There is no replacement function.)

`SetOutlinePreferred` (page 69) Deprecated in Mac OS X v10.4
: Sets the preference for whether to use bitmapped or outline fonts when both kinds of fonts are available. (Deprecated. There is no replacement function.)

`SetPreserveGlyph` (page 70) Deprecated in Mac OS X v10.4
: Temporarily changes the default behavior of the Font Manager, so that it does not scale oversized glyphs. (Deprecated. There is no replacement function.)

### Working with Antialiased Text

`IsAntiAliasedTextEnabled` (page 63) Deprecated in Mac OS X v10.4
: Checks whether antialiased text is enabled. (Deprecated. There is no replacement function.)

`SetAntiAliasedTextEnabled` (page 67) Deprecated in Mac OS X v10.4
: Enables or disables antialiased text for an application. (Deprecated. There is no replacement function.)

### Working With Font Measurements and Scaling

`QDTextBounds` (page 66) Deprecated in Mac OS X v10.4
: Obtains a rectangle that specifies the bounds of QuickDraw text. (Deprecated. There is no replacement function.)

`SetFractEnable` (page 68) Deprecated in Mac OS X v10.4
: Enables or disables fractional glyph widths. (Deprecated. There is no replacement function.)

`SetFScaleDisable` (page 68) Deprecated in Mac OS X v10.4
: Enables or disables the computation of font scaling factors by the Font Manager for bitmapped glyphs. (Deprecated. There is no replacement function.)

### Using the Current, System, and Application Fonts

`GetAppFont` (page 60) Deprecated in Mac OS X v10.4
: Returns the font family ID of the current application font. (Deprecated. There is no replacement function.)

`GetDefFontSize` (page 60) Deprecated in Mac OS X v10.4
: Determines the default size of the system font. (Deprecated. There is no replacement function.)

`GetSysFont` (page 62) Deprecated in Mac OS X v10.4
: Obtains the font family ID of the current system font. (Deprecated. There is no replacement function.)

## Functions

### FMFontGetCGFontRefFromFontFamilyInstance

Obtains the Quartz font associated with a typeface from a QuickDraw font family.

```
OSStatus FMFontGetCGFontRefFromFontFamilyInstance (
   FMFontFamily iFontFamily,
   FMFontStyle iStyle,
   CGFontRef *oFont,
   FMFontStyle *oStyle
);
```

**Parameters**

*`iFontFamily`*
: A QuickDraw font family.

*`iStyle`*
: A QuickDraw font style.

*`oFont`*
: A pointer to a Quartz font reference. On output, points to the Quartz font reference for the specified font family and style. You are responsible for allocating the memory for the Quartz font reference.

*`oStyle`*
: On output, a pointer to an intrinsic font style. If a font object isn't found that matches the font family reference and font style you specify, the function returns the QuickDraw style that matches most closely.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35). If a font reference and intrinsic style are not found, the function returns a value of `kFMInvalidFontErr`.

**Availability**

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMGetATSFontRefFromFont

Obtains the ATS font reference associated with a font object.

```
ATSFontRef FMGetATSFontRefFromFont (
   FMFont iFont
);
```

**Parameters**

*`iFont`*
: A font reference.

**Return Value**

The `ATSFontRef` associated with the font object.

**Availability**

Available in Mac OS X v10.1 and later.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

### FMGetFontFromATSFontRef

Obtains the font object associated with an ATS font reference.

```
FMFont FMGetFontFromATSFontRef (
   ATSFontRef iFont
);
```

#### Parameters

*iFont*

An ATS font reference.

#### Return Value

The font reference associated with the specified ATS font reference.

#### Availability

Available in Mac OS X v10.1 and later.

Not available to 64-bit applications.

#### Declared In

`Fonts.h`

## Data Types

### Font and Font Family Data Structures

#### FMFontContainer

Represents a font container.

```
typedef UInt32 FMFontContainer;
```

#### FMFontInstance

Contains information for a font instance.

```
struct FMFontInstance {
    FMFont font;
    UInt32 fontInstanceIndex;
};
```

**Fields**

`font`

A font reference.

`fontInstanceIndex`

The index associated with the font.

### FMFontSpecification

Contains a font family and style.

```
struct FMFontSpecification {
    FMFontFamily fontFamily;
    SInt16 style;
};
```

**Fields**

`fontFamily`

A font family reference.

`style`

A font style.

### FontFamilyID

Represents the ID of a font family.

```
typedef FMFontFamily FontFamilyID;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`Fonts.h`

## Font Input and Output Structures

### FMInput

Contains information about a specific font.

```
struct FMInput {
    SInt16 family;
    SInt16 size;
    Style face;
    Boolean needBits;
    SInt16 device;
    Point numer;
    Point denom;
    float x;
    float y;
    float width;
    float height;
    CGPoint origin;
    CGSize size;
};
```

**Fields**

`family`

The font family ID.

`size`

The point size of the font.

`face`

The font style. The defined QuickDraw styles are bold, italic, underline, outline, shadow, condense, and extend.

`needBits`

Indicates whether QuickDraw draws the glyphs. If QuickDraw does not draw the glyphs, as is the case for measurement functions such as `MeasureText`, then the glyph bitmaps do not have to be read or constructed. If QuickDraw draws the glyphs and the font is contained in a bitmapped font resource, all of the information describing the font, including the bit image, is read into memory.

`device`

This is no longer used. The high-order byte contains the printer driver reference number as defined in the old Printing Manager. The low-order byte is reserved.

`numer`

The numerators of the vertical and horizontal scaling factors. The `numer` field is of type `Point` and contains two fields: `h` (the numerator of the ratio for horizontal scaling) and `v` (the numerator of the ratio for vertical scaling).

`denom`

The denominators of the vertical and horizontal scaling factors. The `denom` field is of type `Point` and contains two fields: `h` (the denominator of the ratio for horizontal scaling) and `v` (the denominator of the ratio for vertical scaling).

**Discussion**

The font input structure, of data type `FMInput`, is used by QuickDraw when it requests a font from the Font Manager. You can also use this data type when you request a font with the `FMSwapFont` (page 58) function.

## FMOutPtr

Defines a reference to a font output structure.

```
typedef FMOutputPtr FMOutPtr;
```

**Discussion**

See FMOutput (page 15).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`Fonts.h`

## FMOutput

Contains a handle to a font resource and measurement and display information about a specific font.

```
struct FMOutput {
    SInt16 errNum;
    Handle fontHandle;
    UInt8 boldPixels;
    UInt8 italicPixels;
    UInt8 ulOffset;
    UInt8 ulShadow;
    UInt8 ulThick;
    UInt8 shadowPixels;
    SInt8 extra;
    UInt8 ascent;
    UInt8 descent;
    UInt8 widMax;
    SInt8 leading;
    SInt8 curStyle;
    Point numer;
    Point denom;
};
typedef struct FMOutput FMOutput;
```

**Fields**

`errNum`

Reserved for use by Apple Computer, Inc.

`fontHandle`

A handle to the font resource. The font resource can be for either a bitmapped font or outline font resource.

`boldPixels`

A value used by QuickDraw to modify how it applies the bold style on the screen and on raster printers. Other display devices may handle styles differently.

`italicPixels`

A value used by QuickDraw to modify how it applies the italic style on the screen and on raster printers. Other display devices may handle styles differently.

`ulOffset`

A value used by QuickDraw to modify how it applies the underline style on the screen and on raster printers. Other display devices may handle styles differently.

`ulShadow`

A value used by QuickDraw to modify how it applies the underline shadow style on the screen and on raster printers. Other display devices may handle styles differently.

`ulThick`

A value used by QuickDraw to modify how it applies the thickness of the underline style on the screen and on raster printers. Other display devices may handle styles differently.

`shadowPixels`

A value used by QuickDraw to modify how it applies the shadow style on the screen and on raster printers. Other display devices may handle styles differently.

`extra`

The number of pixels by which the styles have widened each glyph.

`ascent`

The ascent measurement of the font. Any algorithmic styles or stretching that may be applied to the font are not taken into account for this value.

`descent`
: The descent measurement of the font. Any algorithmic styles or stretching that may be applied to the font are not taken into account for this value.

`widMax`
: The maximum width of the font. Any algorithmic styles or stretching that may be applied to the font are not taken into account for this value.

`leading`
: The leading assigned to the font. Any algorithmic styles or stretching that may be applied to the font are not taken into account for this value.

`curStyle`
: The actual style being made available for QuickDraw's text drawing, as opposed to the requested style.

`numer`
: The numerators of the vertical and horizontal scaling factors. The `numer` parameter is of type `Point`, and contains two fields: `h` (the numerator of the ratio for horizontal scaling) and `v` (the numerator of the ratio for vertical scaling).

`denom`
: The denominators of the vertical and horizontal scaling factors. The `demon` parameter is of type `Point`, and contains two fields: `h` (the denominator of the ratio for horizontal scaling) and `v` (the denominator of the ratio for vertical scaling).

**Discussion**

The font output structure, of data type `FMOutput`, contains a handle to a font and information about font measurements. It is filled in by the Font Manager upon responding to a font request. You can request a font using the `FMSwapFont` (page 58) function.

The `bold`, `italic`, `ulOffset`, `ulShadow`, `ulThick`, and `shadow` values are all used to communicate to QuickDraw how to modify the way it renders each stylistic variation. Each byte value is taken from the font characterization table of the printer driver and is used by QuickDraw when it draws to a screen or raster printer.

The `ascent`, `descent`, `widMax`, and `leading` values can all be different in this structure than the corresponding values in the `FontInfo` structure that is produced by the `GetFontInfo` function in QuickDraw. This is because `GetFontInfo` takes into account any algorithmic styles or stretching that QuickDraw performs, while the Font Manager functions do not.

The `numer` and `denom` values are used to designate how font scaling is to be done. The values for these fields in the font output structure can be different than the values specified in the font input structure.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`Fonts.h`

## FMOutputPtr

Defines a pointer to a font output structure.

```
typedef FMOutput* FMOutputPtr;
```

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

**Declared In**
`Fonts.h`

## Font Measurements

### FMetricRec

Contains font measurements.

```
struct FMetricRec {
    Fixed ascent;
    Fixed descent;
    Fixed leading;
    Fixed widMax;
    Handle wTabHandle;
};
typedef struct FMetricRec FMetricRec;
```

**Fields**

`ascent`
: The measurement, in pixels, from the baseline to the ascent line of the font. You can determine the line height, in pixels, by adding the values of the `ascent`, `descent`, and `leading` fields of the font metrics structure.

`descent`
: The measurement, in pixels, from the baseline to the descent line of the font.

`leading`
: The measurement, in pixels, from the descent line to the ascent line below it.

`widMax`
: The width, in pixels, of the largest glyph in the font.

`wTabHandle`
: A handle to the global font width table.

**Discussion**
The font metrics structure (of data type `FMetricRec`) contains a handle to the global width table, which in turn contains a handle to the associated font family resource for the current font (the font in the current graphics port). It also contains the values of four measurements for the current font.

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

**Declared In**
`Fonts.h`

### FMetricRecHandle

Defines a handle to a font metrics structure.

```
typedef FMetricRecPtr* FMetricRecHandle;
```

**Discussion**
See FMetricRec (page 18).

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

**Declared In**
`Fonts.h`

### FMetricRecPtr

Defines a pointer to a font metrics structure.

```
typedef FMetricRec* FMetricRecPtr;
```

**Discussion**
See FMetricRec (page 18).

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

**Declared In**
`Fonts.h`

### FontPointSize

Represents the point size of a font.

```
typedef FMFontSize FontPointSize;
```

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

**Declared In**
`Fonts.h`

## Deprecated Data Types

The following data structures referenced by the low memory global variables of the Font Manager are deprecated in Mac OS X and CarbonLib 1.1. The low memory global variables are not shared between processes and may result in inconsistencies compared to previous releases of the system software. Changes made to the information contained in the low memory global variables, including any indirectly reference width tables, font family records, and font records, are not reflected in the global state of the Font Manager and may only be accessed through the font access and data management function of the Font Manager or ATS.

### AsscEntry

Contains the size and style for a specific font.

```
struct AsscEntry {
    SInt16 fontSize;
    SInt16 fontStyle;
    SInt16 fontID;
};
```

**Fields**

`fontSize`
: A font point size.

`fontStyle`
: A font style.

`fontID`
: A font Resource ID.

**Discussion**

The font association entry structure is used in `FontAssoc` (page 21).

### FamRec

Contains format information for a font family resource.

```
struct FamRec {
    SInt16 ffFlags;
    SInt16 ffFamID;
    SInt16 ffFirstChar;
    SInt16 ffLastChar;
    SInt16 ffAscent;
    SInt16 ffDescent;
    SInt16 ffLeading;
    SInt16 ffWidMax;
    SInt32 ffWTabOff;
    SInt32 ffKernOff;
    SInt32 ffStylOff;
    SInt16 ffProperty[9];
    SInt16 ffIntl[2];
    SInt16 ffVersion;
};
```

**Fields**

`ffFlags`
: Flags for family.

`ffFamID`
: Family ID number.

`ffFirstChar`
: ASCII code of first character.

`ffLastChar`
: ASCII code of last character.

`ffAscent`
: Maximum ascent for 1-point font.

`ffDescent`
: Maximum descent for 1-point font.

`ffLeading`
: Maximum leading for 1-point font.

`ffWidMax`
: Maximum glyph width for 1-point font.

`ffWTabOff`
: Offset to family glyph-width table.

`ffKernOff`
: Offset to kerning table.

`ffStylOff`
: Offset to style-mapping table.

`ffProperty`
: Style properties info.

`ffIntl`
: For international use.

`ffVersion`
: Version number.

**Discussion**

The font family structure, of data type `FamRec`, describes the format of the font family (`'FOND'`) resource. It is shown here as a guide to the format of the resource. The font family structure is not used directly by any Font Manager functions.

## FontAssoc

Contains the number of entries in a font association table.

```
struct FontAssoc {
    SInt16 numAssoc;
};
```

**Fields**

`numAssoc`
: Number of entries - 1.

**Discussion**

Each entry in the font association table is a font association entry structure, of data type `AsscEntry` (page 20).

The font association table structure, which is part of the font family resource, maps a point size and style to a specific font that is part of the family. The table structure, of data type `FontAssoc`, consists of a count of the entries in the table and is followed by the entry structures.

## FontRec

Contains information for a format of ‘`NFNT`' and, likewise, the `'FONT'` resource

```
struct FontRec {
    SInt16 fontType;
    SInt16 firstChar;
    SInt16 lastChar;
    SInt16 widMax;
    SInt16 kernMax;
    SInt16 nDescent;
    SInt16 fRectWidth;
    SInt16 fRectHeight;
    UInt16 owTLoc;
    SInt16 ascent;
    SInt16 descent;
    SInt16 leading;
    SInt16 rowWords;
};
typedef struct FontRec FontRec;
```

**Fields**

`fontType`
: Font type.

`firstChar`
: Character code of first glyph.

`lastChar`
: Character code of last glyph.

`widMax`
: Maximum glyph width.

`kernMax`
: Negative of maximum glyph kern.

`nDescent`
: Negative of descent.

`fRectWidth`
: Width of font rectangle.

`fRectHeight`
: Height of font rectangle.

`owTLoc`
: Location of width/offset table.

`ascent`
: Ascent.

`descent`
: Descent.

`leading`
: Leading.

`rowWords`
: Row width of bit image / 2.

**Discussion**

The font structure, of data type `FontRec`, describes the format of ‘`NFNT`' and, likewise, the '`FONT`' resource. It is shown here as a guide to the format of the resource. The font structure is not used directly by any Font Manager functions.

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

**Declared In**
`Fonts.h`

## FontRecHdl

Defines a handle to a font record.

```
typedef FontRecPtr* FontRecHdl;
```

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

**Declared In**
`Fonts.h`

## FontRecPtr

Defines a pointer to a font record.

```
typedef FontRec* FontRecPtr;
```

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

**Declared In**
`Fonts.h`

## KernEntry

Contains kerning information for a specific stylistic variation of the font family.

```
struct KernEntry {
    SInt16 kernStyle;
    SInt16 kernLength;
};
```

**Fields**

`kernStyle`
    Length of this entry.

`kernLength`
    Style to which this entry applies.

## KernPair

Specifies a kerning value for a pair of glyphs.

```
struct KernPair {
    char kernFirst;
    char kernSecond;
    SInt16 kernWidth;
};
```

**Fields**

`kernFirst`

ASCII character code of the first character of a kerned pair.

`kernSecond`

ASCII character code of the second character of a kerned pair.

`kernWidth`

Kerning value in 1pt fixed format.

### KernTable

Contains the number of entries in a kerning table.

```
struct KernTable {
    SInt16 numKerns;
};
```

**Fields**

`numKerns`

Number of subtable entries.

**Discussion**

The font family kerning table structure, which is part of the font family resource, contains a number of kerning subtable entries, with different subtables for different stylistic variations. The table structure, of data type `KernTable`, consists of a count of the entries in the table and is followed by the entry structures.

### NameTable

Contains the base name and suffixes for a font family.

```
struct NameTable {
    SInt16 stringCount;
    Str255 baseFontName;
};
```

**Fields**

`stringCount`

The number of entries in the name table.

`baseFontName`

A string that specifies the base name and suffixes for a font family name.

### StyleTable

Contains font style information

```
struct StyleTable {
    SInt16 fontClass;
    SInt32 offset;
    SInt32 reserved;
    char indexes[48];
};
```

**Fields**

`fontClass`

The font class of this table.

`offset`

Offset to glyph-encoding subtable.

`reserved`

Reserved.

`indexes`

Indexes into the font suffix name table. The font suffix name subtable structure, of data type `NameTable` (page 24), contains the base name and suffixes for a font family.

**Discussion**

The style-mapping table structure, which is part of the font family resource, provides information that is used by printer drivers to implement font styles. Each font family can have its own character encoding and its own set of font suffix names for style designations. Each style of a font has its own name, typically created by adding a style suffix to the base name of the font.

## WidEntry

Specifies a style for a glyph width.

```
struct WidEntry {
    SInt16 widStyle;
};
```

**Fields**

`widStyle`

The style to which the entry applies.

## WidTable

Specifies the number of entries in a font family glyph-width table.

```
struct WidTable {
    SInt16 numWidths;
};
```

**Fields**

`numWidths`

The number of entries minus one.

**Discussion**

The font family glyph-width table structure, which is part of the font family resource, is used to specify glyph widths for the font family on a per-style basis.

### WidthTable

Contains the widths of all the glyphs of a specific font.

```
struct WidthTable {
    Fixed tabData[256];
    Handle tabFont;
    SInt32 sExtra;
    SInt32 style;
    SInt16 fID;
    SInt16 fSize;
    SInt16 face;
    SInt16 device;
    Point inNumer;
    Point inDenom;
    SInt16 aFID;
    Handle fHand;
    Boolean usedFam;
    UInt8 aFace;
    SInt16 vOutput;
    SInt16 hOutput;
    SInt16 vFactor;
    SInt16 hFactor;
    SInt16 aSize;
    SInt16 tabSize;
};
typedef struct WidthTable WidthTable;
```

**Fields**

`tabData`
: The widths for the glyphs in the font, in standard 32-bit fixed-point format. If a glyph is missing in the font, its entry contains the width of the missing-character glyph.

`tabFont`
: A handle to the font resource used to build this table.

`sExtra`
: The average number of pixels by which QuickDraw widens each space in a line of text.

`style`
: The average number of pixels by which QuickDraw widens a line of text after applying a style.

`fID`
: The font family ID of the font represented by this table. This is the ID that was used in the request to build the table. It may be different from the ID of the font family that was used, which is indicated by the `aFID` field.

`fSize`
: The point size that was originally requested for the font represented by this table. The actual size used is specified in the `aSize` field.

`face`
: The font style that was originally requested for the font represented by this table. The actual style used is specified in the `aFace` field.

`device`
: The device ID of the device on which these widths may be used.

`inNumer`
: The numerators of the vertical and horizontal scaling factors. The `numer` parameter is of type `Point`, and contains two fields: `h` (the numerator of the ratio for horizontal scaling) and `v` (the numerator of the ratio for vertical scaling).

`inDenom`
: The denominators of the vertical and horizontal scaling factors. The `denom` parameter is of type `Point`, and contains two fields: `h` (the denominator of the ratio for horizontal scaling) and `v` (the denominator of the ratio for vertical scaling).

`aFID`
: The font family ID of the font family actually used to build this table. If the Font Manager could not find the font requested, this value may be different from the value of the `fID` field.

`fHand`
: The handle to the font family resource used to build this table.

`usedFam`
: Set to `TRUE` if the fixed-point family glyph widths were used rather than integer glyph widths.

`aFace`
: The font style of the font whose widths are contained in this table.

`vOutput`
: The factor by which glyphs are to be expanded vertically in the current graphics port. This is a 16-bit fixed-point number, with the integer part in the high-order byte and a fractional part in the low-order byte.

`hOutput`
: The factor by which glyphs are to be expanded horizontally in the current graphics port. This is a 16-bit fixed-point number, with the integer part in the high-order byte and a fractional part in the low-order byte.

`vFactor`
: The factor by which widths of the chosen font, after a style has been applied, have been increased vertically in the current graphics port. This is a 16-bit fixed-point number, with the integer part in the high-order byte and a fractional part in the low-order byte. The value of the `vFactor` field is not used by the Font Manager.

`hFactor`
: The factor by which widths of the chosen font, after a style has been applied, have been increased horizontally in the current graphics port. This is a 16-bit fixed-point number, with the integer part in the high-order byte and a fractional part in the low-order byte.

`aSize`
: The size of the font actually used to build this table. Both the point size and the font used to build this table may be different from the requested point size and font. If font scaling is disabled, the Font Manager may use a size different from the size requested and add more or less space to approximate the appearance of the font requested.

`tabSize`
: The total size of the global width table.

**Discussion**

The global width table structure, of data type `WidthTable`, contains the widths of all the glyphs of one font. The font family, point size, and style of this font are specified in this table. Your application should use the widths found in the global width table for placement of glyphs and words both on the screen and on the printed page. You can use the `FontMetrics` (page 59) function to get a handle to the global width table. However, you should not assume that the table is the same size as shown in the structure declaration; it may be larger because of some private system-specific information that is attached to it.

Multiplying the values of the `hOutput` and `vOutput` fields by the values of the `hFactor` and `vFactor` fields, respectively, gives the font scaling. (Because the value of the `vFactor` field is ignored, the Font Manager multiplies the value of the `vOutput` field by 1.) The product of the value of the `hOutput` field and an entry in the global width table is the scaled width for that glyph.

The Font Manager gathers data for the global width table from one of three data structures:

1. The Font Manager looks in the font resource for a table that stores fractional glyph widths. For bitmapped fonts, the Font Manager uses the glyph-width table of the bitmapped font resource. For outline fonts, the Font Manager uses the advance width and left-side bearing values in the horizontal metrics table of the outline font. In both cases, the values are stored in 16-bit fixed format, with the integer part in the high-order byte and the fractional part in the low-order byte.

2. If there is no glyph-width table in the font resource, the Font Manager looks for the font family's glyph-width table in the font family resource, which contains fractional widths for a hypothetical 1-point font. The Font Manager calculates the actual values by multiplying these widths by the requested font size.

3. If there is no glyph-width table in the font family resource, and if the font is contained in a bitmapped font resource, the Font Manager derives the glyph widths from the integer widths contained in the glyph-width table of the bitmapped font resource. There is no corresponding table for the outline font resource.

Your application should obtain glyph widths either from the global width table or from the QuickDraw function `MeasureText`. The `MeasureText` function works only with text to be displayed on the screen, not with text to be printed. You can get the individual widths of glyphs of an outline font using the `OutlineMetrics` function. The `FontMetrics` function returns only the width of the largest glyph in a font contained in a bitmapped font resource.

Do not use the values from the global width table if your application is running on a computer on which non-Roman script systems are installed. You can check to see if a non-Roman script system is present by calling the `GetScriptManagerVariable` function with a selector of `smEnabled`; if the function returns a value greater than 0, at least one non-Roman script system is present and you need to call `MeasureText` to measure text that is displayed on the screen. Measuring text from a non-Roman script system for printing is handled by the printer driver.

For more information about the `MeasureText` function, see the documentation on "QuickDraw Text". See also the `FontMetrics` (page 59) and `OutlineMetrics` (page 64) functions.

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

**Declared In**
`Fonts.h`

### WidthTableHdl

Defines a handle to a glyph width table.

```
typedef WidthTablePtr* WidthTableHdl;
```

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

**Declared In**
`Fonts.h`

### WidthTablePtr

Defines a pointer to a glyph width table.

```
typedef WidthTable* WidthTablePtr;
```

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

**Declared In**
`Fonts.h`

## Constants

### Activation Contexts

Specify the scope of available fonts.

```
enum{
kFMDefaultActivationContext = kFMDefaultOptions,
kFMGlobalActivationContext = 0x00000001,
kFMLocalActivationContext = kFMDefaultActivationContext
}
```

**Constants**

`kFMDefaultActivationContext`
: Specifies to use the default scope, which is local.

  Available in Mac OS X v10.1 and later.

  Declared in `Fonts.h`.

`kFMGlobalActivationContext`
: Specifies the scope is global; fonts are available to all applications.

  Available in Mac OS X v10.1 and later.

  Declared in `Fonts.h`.

`kFMLocalActivationContext`
: Specifies the scope is local; fonts are available only to the application.

  Available in Mac OS X v10.1 and later.

  Declared in `Fonts.h`.

### Default Options

Specify the scope of fonts for an application.

```
enum {
    kFMDefaultOptions = kNilOptions
};
```

**Constants**

`kFMDefaultOptions`

Restricts the scope only to the fonts accessible to your application. This flag is also used when Apple has not yet defined options for a function that has an options parameter. When no options are defined yet, you can use `kFMDefaultOptions` as a neutral value to assure future compatibility.

Available in Mac OS X v10.0 and later.

Declared in `Fonts.h`.

## Font ID Constants

Specify a font. These constants are deprecated.

```
enum {
    kFontIDNewYork = 2,
    kFontIDGeneva = 3,
    kFontIDMonaco = 4,
    kFontIDVenice = 5,
    kFontIDLondon = 6,
    kFontIDAthens = 7,
    kFontIDSanFrancisco = 8,
    kFontIDToronto = 9,
    kFontIDCairo = 11,
    kFontIDLosAngeles = 12,
    kFontIDTimes = 20,
    kFontIDHelvetica = 21,
    kFontIDCourier = 22,
    kFontIDSymbol = 23,
    kFontIDMobile = 24
};
```

## Font Constants

Specify a font. These constants are deprecated.

```
enum {
    newYork = 2,
    geneva = 3,
    monaco = 4,
    venice = 5,
    london = 6,
    athens = 7,
    sanFran = 8,
    toronto = 9,
    cairo = 11,
    losAngeles = 12,
    times = 20,
    helvetica = 21,
    courier = 22,
    symbol = 23,
    mobile = 24
};
```

**Discussion**
You should use the functions `GetFNum` or `FMGetFontFamilyFromName` to find a font family from a standard QuickDraw name.

## Global Scope Option

```
enum {
kFMUseGlobalScopeOption
};
```

**Discussion**
Use the constant `kFMGlobalIterationScope` instead; `kFMUseGlobalScopeOption` is deprecated.

## Height and Width Constants

Specify proportional or fixed font heights and widths.

```
enum {
    propFont = 36864,
    prpFntH = 36865,
    prpFntW = 36866,
    prpFntHW = 36867,
    fixedFont = 45056,
    fxdFntH = 45057,
    fxdFntW = 45058,
    fxdFntHW = 45059,
    fontWid = 44208
};
```

**Constants**
`propFont`
Specifies a proportional font.

Available in Mac OS X v10.0 and later.

Declared in `Fonts.h`.

`prpFntH`
: Specifies a proportional height font.

  Available in Mac OS X v10.0 and later.

  Declared in `Fonts.h`.

`prpFntW`
: Specifies a proportional width font.

  Available in Mac OS X v10.0 and later.

  Declared in `Fonts.h`.

`prpFntHW`
: Specifies a proportional width and height font.

  Available in Mac OS X v10.0 and later.

  Declared in `Fonts.h`.

`fixedFont`
: Specifies a fixed font.

  Available in Mac OS X v10.0 and later.

  Declared in `Fonts.h`.

`fxdFntH`
: Specifies a fixed height font.

  Available in Mac OS X v10.0 and later.

  Declared in `Fonts.h`.

`fxdFntW`
: Specifies a fixed width font.

  Available in Mac OS X v10.0 and later.

  Declared in `Fonts.h`.

`fxdFntHW`
: Specifies a fixed height and width font.

  Available in Mac OS X v10.0 and later.

  Declared in `Fonts.h`.

`fontWid`
: Specifies a font width.

  Available in Mac OS X v10.0 and later.

  Declared in `Fonts.h`.

## Iteration Scopes

Specify a scope over which to iterate.

```
enum{
kFMDefaultIterationScope = kFMDefaultOptions,
kFMGlobalIterationScope = 0x00000001,
kFMLocalIterationScope = kFMDefaultIterationScope
}
```

**Constants**

`kFMDefaultIterationScope`

Specifies to use the default.

Available in Mac OS X v10.1 and later.

Declared in `Fonts.h`.

`kFMGlobalIterationScope`

Specifies the scope is global, iterate over all applications.

Available in Mac OS X v10.1 and later.

Declared in `Fonts.h`.

`kFMLocalIterationScope`

Specifies the scope is local, restrict the iteration to the application.

Available in Mac OS X v10.1 and later.

Declared in `Fonts.h`.

## Marking Character Constants

Specify a character to use for an active menu or submenu item.

```
enum {
    commandMark = 17,
    checkMark = 18,
    diamondMark = 19,
    appleMark = 20
};
```

**Constants**

`commandMark`

Specifies to use a command mark next to an active menu or submenu item.

Available in Mac OS X v10.0 and later.

Declared in `Fonts.h`.

`checkMark`

Specifies to use a check mark next to an active menu or submenu item.

Available in Mac OS X v10.0 and later.

Declared in `Fonts.h`.

`diamondMark`

Specifies to use a diamond mark next to an active menu or submenu item.

Available in Mac OS X v10.0 and later.

Declared in `Fonts.h`.

`appleMark`

Specifies to use an Apple character next to an active menu or submenu item.

Available in Mac OS X v10.0 and later.

Declared in `Fonts.h`.

**Discussion**

You can pass these constants in the `markChar` parameter of the Menu Manager function `GetItemMark` and the marking character field of the menu resource (of type `'MENU'`) and return these constants in the `markChar` parameter of the Menu Manager function `SetItemMark` to specify the mark of a specific menu item or the menu ID of the submenu associated with the menu item.

## QuickTime User Interface Default Font

Defines the default font for the QuickTime user interface.

```
enum {
    kPlatformDefaultGuiFontID = applFont;
};
```

**Constants**

`kPlatformDefaultGuiFontID`

Specifies that the default font ID for the graphical user interface in QuickTime 3.0 should be the application font ID.

Available in Mac OS X v10.0 and later.

Declared in `Fonts.h`.

**Discussion**

This constant is used in QuickTime 3.0.

## System and Application Fonts

Specify the current system and application fonts.

```
enum {
    systemFont = 0,
    applFont = 1
};
```

**Constants**

`systemFont`

Specifies the current System font.

Available in Mac OS X v10.0 and later.

Declared in `Fonts.h`.

`applFont`

Specifies the current application font.

Available in Mac OS X v10.0 and later.

Declared in `Fonts.h`.

## Result Codes

The most common result codes returned by Font Manager are listed below.

| Result Code | Value | Description |
|---|---|---|
| `kFMIterationCompleted` | -980 | Iteration successfully completed.<br>Available in Mac OS X v10.0 and later. |
| `kFMInvalidFontFamilyErr` | -981 | The specified font family is invalid.<br>Available in Mac OS X v10.0 and later. |
| `kFMInvalidFontErr` | -982 | The specified font is invalid.<br>Available in Mac OS X v10.0 and later. |
| `kFMIterationScopeModifiedErr` | -983 | Iteration scope modified.<br>Available in Mac OS X v10.0 and later. |
| `kFMTableAccessErr` | -984 | The table specified is invalid or doesn't exist. |
| `kFMFontContainerAccessErr` | -985 | Not able to access font container.<br>Available in Mac OS X v10.0 and later. |

# Deprecated Font Manager Functions

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

## Deprecated in Mac OS X v10.4

### FetchFontInfo

Obtains the information for a specific font. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
OSErr FetchFontInfo (
   SInt16 fontID,
   SInt16 fontSize,
   SInt16 fontStyle,
   FontInfo *info
);
```

**Parameters**

*fontID*
A signed, 16-bit integer that specifies the font ID of the font whose information you want to obtain.

*fontSize*
A signed, 16-bit integer that specifies the font size of the font whose information you want to obtain.

*fontStyle*
A signed, 16-bit integer that specifies the font style of the font whose information you want to obtain.

*info*
On output, points to a font information structure that contains measurement information (ascent, descent, width, and leading) for the specified font.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

**Declared In**
`Fonts.h`

### FMActivateFonts

Activates one or more fonts. (Deprecated in Mac OS X v10.4. Use `ATSFontActivateFromFileReference` instead.)

```
OSStatus FMActivateFonts (
   const FSSpec *iFontContainer,
   const FMFilter *iFilter,
   void *iRefCon,
   OptionBits iOptions
);
```

**Parameters**

*iFontContainer*

A pointer to the file specification of the file that contains the font data you want to activate. You can specify a directory or an individual font file.

*iFilter*

A pointer to a filter specification. This parameter is currently reserved for future use, so you should pass `NULL`.

*iRefCon*

An arbitrary 32-bit value specified by your application. This parameter is currently reserved for future use, so you should pass `NULL`.

*iOptions*

A value that specifies the scope to which the function applies. If you want the Font Manager to make the fonts visible only to your application, use the constant `kFMLocalActivationContext`. If you want the Font Manager to make fonts visible to all applications installed on the system, use the constant `kFMGlobalActivationContext`. See Activation Contexts (page 29) for more information on these constants.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMCreateFontFamilyInstanceIterator

Creates a font family instance iterator that your application can use to access the member fonts associated with a font family. (Deprecated in Mac OS X v10.4.)

```
OSStatus FMCreateFontFamilyInstanceIterator (
   FMFontFamily iFontFamily,
   FMFontFamilyInstanceIterator *ioIterator
);
```

**Parameters**

*iFontFamily*

A reference to the font family you want to access.

*ioIterator*
: A pointer to a structure of type `FMFontFamilyInstanceIterator`. On input, pass a pointer to an uninitialized structure. On output, its contents may have been changed and may include references to other data structures allocated by the system to maintain the structure's state. The iterator is positioned before the first member font of the font family. When you no longer need the font family instance iterator, you should call the function `FMDisposeFontFamilyInstanceIterator` to release the auxiliary data and memory allocated by the system.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Discussion**
A font family instance iterator is an opaque data structure used by the Font Manager to keep track of an iteration over currently active font family instances. A font family instance is a typeface and a size—an entry from the font association table.

When the font family iterator is initialized, it does not yet reference a font family instance. Do not attempt to modify the contents of a font family instance iterator.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMCreateFontFamilyIterator

Creates a font family iterator that your application can use to access font family objects. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyIteratorCreate` instead.)

```
OSStatus FMCreateFontFamilyIterator (
   const FMFilter *iFilter,
   void *iRefCon,
   OptionBits iOptions,
   FMFontFamilyIterator *ioIterator
);
```

**Parameters**

*iFilter*
: A pointer to a filter specification. Pass `NULL` if you want to access all font family objects within the scope of your iteration. Otherwise, you can use this parameter to restrict the scope of the iteration to the font families that match a generation count or criteria you specify in a custom filter function. Pass the filter selector constant `kFMGenerationFilterSelector` to select a generation filter or the constant `kFMFontFamilyCallbackFilterSelector` to select a custom filter. See `FMFilterSelector` in the ATS Types Reference for more information on these constants.

*iRefCon*
: An arbitrary 32-bit value specified by your application. If you are using a custom filter function, you can use this parameter to pass data to the custom filter function. If you are not using a custom filter function, pass `NULL`.

*iOptions*
A value that specifies the scope to which the font family iterator applies. If you want the Font Manager to apply the font family iterator only to the fonts accessible from your application use the `kFMLocalIterationScope` constant. If you want the Font Manager to apply the font family iterator to all fonts registered with the system use the constant `kFMGlobalIterationScope`. See Activation Contexts (page 29) for more information on these constants.

*ioIterator*
A pointer to a structure of type `FMFontFamilyIterator`. On input, pass a pointer to an uninitialized structure. On output, the structure's contents may have been changed and may include references to other data structures allocated by the system to maintain the structure's state. When you no longer need the font family iterator, you should call the function `FMDisposeFontFamilyIterator` to release the auxiliary data and memory allocated by the system.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Discussion**
A font family iterator is an opaque data structure used by the Font Manager to keep track of an iteration over currently active font families. When the font family iterator is initialized, it does not yet reference a font family. Do not attempt to modify the contents of a font family iterator.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMCreateFontIterator

Creates an iterator that your application can use to access fonts. (Deprecated in Mac OS X v10.4. Use `ATSFontIteratorCreate` instead.)

```
OSStatus FMCreateFontIterator (
   const FMFilter *iFilter,
   void *iRefCon,
   OptionBits iOptions,
   FMFontIterator *ioIterator
);
```

**Parameters**

*iFilter*
A pointer to font filter specification. Pass `NULL` if you want to access all font objects within the scope of your iteration. Otherwise, you can use this parameter to restrict the scope of the iteration to font information that matches a technology, font container, or criteria you specify in a custom filter function. Pass the filter selector constant `kFMFontTechnologyFilterSelector` to select a font technology filter, the constant `kFMFontContainerFilterSelector` to select a font container filter, or the constant `kFMFontCallbackFilterSelector` to select a custom filter. See `FMFilterSelector` in the ATS Types Reference for more information on these constants.

*iRefCon*

An arbitrary 32-bit value specified by your application. If you are using a custom filter function, you can use this parameter to pass data to the custom filter function. If are not using a custom filter function, pass `NULL`.

*iOptions*

A value that specifies the scope to which the font iterator applies. If you want the Font Manager to apply the font iterator only to the fonts accessible from your application use the `kFMLocalIterationScope` constant. If you want the Font Manager to apply the font iterator to all fonts registered with the system use the constants `kFMGlobalIterationScope`. See Activation Contexts (page 29) for more information on these constants.

*ioIterator*

A pointer to a structure of type `FMFontIterator`. On input, pass a pointer to an uninitialized structure. On output, the structure's contents may have been changed and may include references to other data structures allocated by the system to maintain the structure's state. When you no longer need the font iterator, you should call the function `FMDisposeFontIterator` to release the auxiliary data and memory allocated by the system.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Discussion**

A font iterator is an opaque structure used by the Font Manager to maintain font information in the context of the current application process. When the font iterator is initialized, it is not yet positioned on a font object. You should not attempt to modify the contents of a font iterator.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMDeactivateFonts

Deactivates one or more fonts. (Deprecated in Mac OS X v10.4. Use `ATSFontDeactivate` instead.)

```
OSStatus FMDeactivateFonts (
   const FSSpec *iFontContainer,
   const FMFilter *iFilter,
   void *iRefCon,
   OptionBits iOptions
);
```

**Parameters**

*iFontContainer*

A pointer to the file specification of the file that contains the font data you want to deactivate. You can specify a directory or an individual font file.

*iFilter*

A pointer to a filter specification. This parameter is currently reserved for future use, so you should pass `NULL`.

*iRefCon*
: An arbitrary 32-bit value specified by your application. This parameter is currently reserved for future use, so you should pass `NULL`.

*iOptions*
: A value that specifies the scope to which the function applies. This parameter is currently reserved for future use, so you should pass `NULL`.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMDisposeFontFamilyInstanceIterator

Disposes of a font family instance iterator. (Deprecated in Mac OS X v10.4.)

```
OSStatus FMDisposeFontFamilyInstanceIterator (
   FMFontFamilyInstanceIterator *ioIterator
);
```

**Parameters**

*ioIterator*
: A pointer to a font family instance iterator you created with the function `FMCreateFontFamilyInstanceIterator` (page 38). If you try to use the font family instance iterator after disposing of its contents through this function, the Font Manager returns an error code to your application.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMDisposeFontFamilyIterator

Disposes of the contents of a font family iterator. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyIteratorRelease` instead.)

```
OSStatus FMDisposeFontFamilyIterator (
   FMFontFamilyIterator *ioIterator
);
```

**Parameters**

*ioIterator*

A pointer to a font family iterator you created with the function `FMCreateFontFamilyIterator` (page 39). If you try to use the font family iterator after disposing of its contents through this function, the Font Manager returns an error code to your application.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMDisposeFontIterator

Disposes of a font iterator. (Deprecated in Mac OS X v10.4. Use `ATSFontIteratorRelease` instead.)

```
OSStatus FMDisposeFontIterator (
   FMFontIterator *ioIterator
);
```

**Parameters**

*ioIterator*

A pointer to a font iterator you created with the function `FMCreateFontIterator` (page 40). If you try to use the font iterator after disposing of its contents through this function, the Font Manager returns an error code to your application.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMGetATSFontFamilyRefFromFontFamily

Obtains the ATS font family reference associated with a font family object. (Deprecated in Mac OS X v10.4.)

```
ATSFontFamilyRef FMGetATSFontFamilyRefFromFontFamily (
   FMFontFamily iFamily
);
```

**Parameters**

*iFamily*

A font family reference.

**Return Value**

The `ATSFontFamilyRef` associated with the font family object.

**Availability**

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMGetFontContainer

Obtains the file that contains data for a font. (Deprecated in Mac OS X v10.4. Use `ATSFontGetContainer` instead.)

```
OSStatus FMGetFontContainer (
   FMFont iFont,
   FSSpec *oFontContainer
);
```

**Parameters**

*iFont*

A font reference.

*oFontContainer*

On output, a pointer to the file specification of the file that contains the font data.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Discussion**

You can pass the file specification returned by this function to the Resource Manager or File Manager to obtain the actual font data. However, if the font is an LWFN-class font, the outline data is located in a separate file from the font suitcase. The function `FMGetFontContainer` obtains the font suitcase. Your application is responsible for finding the individual outline files.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMGetFontContainerFromFontFamilyInstance

Obtains the font container associated with a font family instance. (Deprecated in Mac OS X v10.4. Use `ATSFontGetContainer` instead.)

```
OSStatus FMGetFontContainerFromFontFamilyInstance (
   FMFontFamily iFontFamily,
   FMFontStyle iStyle,
   FMFontSize iFontSize,
   FSSpec *oFontContainer
);
```

**Parameters**

*`iFontFamily`*

A font family reference for the font family whose container you want to obtain. You must pass a valid font family.

*`iStyle`*

The font style of the font family whose container you want to obtain. You must pass a valid font style.

*`iFontSize`*

The font size of the font family whose container you want to obtain. You must pass a valid font size.

*`oFontContainer`*

On output, a pointer to a file specification that specifies the name and location of the font container.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Availability**

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMGetFontFamilyFromATSFontFamilyRef

Obtains the font family associated with an ATS font family reference. (Deprecated in Mac OS X v10.4.)

```
FMFontFamily FMGetFontFamilyFromATSFontFamilyRef (
   ATSFontFamilyRef iFamily
);
```

**Parameters**

*`iFamily`*

An ATS font family reference.

**Return Value**

The font family reference associated with the specified ATS font family reference.

**Availability**

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetFontFamilyFromName

Returns the font family reference associated with a standard QuickDraw name. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyFindFromName` instead.)

```
FMFontFamily FMGetFontFamilyFromName (
   ConstStr255Param iName
);
```

**Parameters**

*`iName`*

A QuickDraw font family name.

**Return Value**
A font family reference. The function returns `kInvalidFontFamily` if it cannot find a matching font family. See the ATS Types documentation for a description of the `FMFontFamily` data type.

**Discussion**
This function is a replacement for the `GetFNum` (page 60) function. You should use the function `FMGetFontFamilyFromName` instead of the function `GetFNum` to assure your application supports font formats other than the resource fork TrueType and PostScript Type 1 fonts.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetFontFamilyGeneration

Obtains the generation count of a font family. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyGetGeneration` instead.)

```
OSStatus FMGetFontFamilyGeneration (
   FMFontFamily iFontFamily,
   FMGeneration *oGeneration
);
```

**Parameters**

*`iFontFamily`*

A font family reference.

*`oGeneration`*

On output, a pointer to the generation count for the font family associated with the font family reference.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetFontFamilyInstanceFromFont

Finds the font family reference and standard QuickDraw style associated with a font. (Deprecated in Mac OS X v10.4.)

```
OSStatus FMGetFontFamilyInstanceFromFont (
   FMFont iFont,
   FMFontFamily *oFontFamily,
   FMFontStyle *oStyle
);
```

**Parameters**

*`iFont`*

A font reference.

*`oFontFamily`*

A pointer to a font family reference. On output, points to the font family reference associated with the specified font. You are responsible for allocating the memory for the font family reference.

*`oStyle`*

A pointer to a font style. On output, points to the font style associated with the specified font. You are responsible for allocating the memory for the font style.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Discussion**
A font can be a member of more than one font family. This means if you call the function `FMGetFontFromFontFamilyInstance` (page 51) and then call the function `FMGetFontFamilyInstanceFromFont`, you will not necessarily get the font family reference you supplied when you called `FMGetFontFromFontFamilyInstance`.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetFontFamilyName

Obtains the font family name associated with a font family reference. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyGetName` instead.)

```
OSStatus FMGetFontFamilyName (
   FMFontFamily iFontFamily,
   Str255 oName
);
```

**Parameters**

*iFontFamily*

A font family reference.

*oName*

On output, the string contains the QuickDraw font family name. If the function does not find a name, it returns an empty string and a result code of `kFMInvalidFontFamilyErr`.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Discussion**

This function is a replacement for the `GetFontName` (page 61) function. You should use the function `FMGetFontFamilyName` instead of the function `GetFontName` to assure your application supports font formats other than the resource fork TrueType and PostScript Type 1 fonts.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMGetFontFamilyResource

Obtains the font family resource for a font family. (Deprecated in Mac OS X v10.4. Use `ATSFontGetFontFamilyResource` instead.)

```
OSStatus FMGetFontFamilyResource (
   FMFontFamily iFontFamily,
   FMFontStyle iFontStyle,
   FMFontSize iFontSize,
   ByteCount iBufferSize,
   void *ioBuffer,
   ByteCount *oSize
);
```

**Parameters**

*iFontFamily*

A value of type `FMFontFamily` that specifies the font family whose resource you want to obtain. You must pass a valid font family.

*iFontStyle*

A value of type `FMFontStyle` that specifies the font style of the font family whose resource you want to obtain. You must pass a valid font style.

*iFontSize*

A value of type `FMFontSize` that specifies the font size of the font family whose resource you want to obtain. You must pass a valid font size.

*iBufferSize*
: The size of the buffer (`ioBuffer`).

*ioBuffer*
: A pointer to the buffer used to store a copy of the font family resource. On input, pass `NULL` if you want to obtain only the length of the font family resource, not its contents.

*oSize*
: On output, the actual size of the buffer.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Discussion**
You should call the function `FMGetFontFamilyResource` twice. First, to get the length of the font family resource. Then after you allocate a buffer (`ioBuffer`) of the appropriate size, call the function a second time to obtain the contents of the font family resource.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetFontFamilyTextEncoding

Obtains the text encoding used by a font family. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyGetEncoding` instead.)

```
OSStatus FMGetFontFamilyTextEncoding (
   FMFontFamily iFontFamily,
   TextEncoding *oTextEncoding
);
```

**Parameters**

*iFontFamily*
: A font family reference.

*oTextEncoding*
: On output, a pointer to the text encoding used by the font family associated with the font family reference.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Discussion**
This function is a replacement for the Script Manager function `FontToScript`. You should use the function `FMGetFontFamilyTextEncoding` instead of the function `FontToScript` to ensure your application supports font formats other than the resource fork TrueType and PostScript Type 1 fonts. Unlike the `FontToScript` function, the state of the font force flag is ignored and the script system of the font family is not mapped to zero even if the script system is disabled in the current application process.

Once you have obtained the text encoding, you can use Text Encoding Converter Manager function `RevertTextEncodingToScriptInfo` to extract the script as follows:

```
status = FMGetFontFamilyTextEncoding (myFontFamily, &myTextEncoding)

status = RevertTextEncodingToScriptInfo (myTextEncoding, &myScriptCode);
```

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetFontFormat

Obtains the format identifier of a font. (Deprecated in Mac OS X v10.4.)

```
OSStatus FMGetFontFormat (
   FMFont iFont,
   FourCharCode *oFormat
);
```

**Parameters**

*`iFont`*

A font reference.

*`oFormat`*

On output, a pointer to a four-character-code that represents the format identifier of the font. See the discussion that follows for information on format identifiers.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Discussion**
A format identifier is a four-character-code, assigned to a font by a font vendor, that identifies the format of a font. Some of the identifiers currently supported in the Mac OS are:

- `'true'` TrueType fonts use the 32-bit hexadecimal value 0x00010000.
- `'LWFN'` PostScript Type 1 fonts ("LaserWriter Font") consist of two parts: a `'FOND'` resource (contained in a font or font suitcase resource file) whose style mapping table references PostScript font data for each typeface (style), stored in separate file. The separate data files have names derived from the PostScript name of the typeface.
- `'typ1'` PostScript Type 1 fonts are housed in packages that have an `'sfnt'` format (OpenType).
- `'OTTO'` PostScript compact font format (CFF) font data is housed in a package that has an `'sfnt'` format (OpenType).

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetFontFromFontFamilyInstance

Obtains the font reference associated with a standard QuickDraw style and font family. (Deprecated in Mac OS X v10.4. Use `CTFontCreateWithQuickdrawInstance` instead.)

```
OSStatus FMGetFontFromFontFamilyInstance (
   FMFontFamily iFontFamily,
   FMFontStyle iStyle,
   FMFont *oFont,
   FMFontStyle *oIntrinsicStyle
);
```

### Parameters

*iFontFamily*
A font family reference.

*iStyle*
A font style.

*oFont*
A pointer to a font reference. On output, points to the font reference for the specified font family and style. You are responsible for allocating the memory for the font reference.

*oIntrinsicStyle*
On output, a pointer to an intrinsic font style. If a font object isn't found that matches the font family reference and font style you specify, the function returns the QuickDraw style that matches most closely.

### Return Value

A result code. See "Font Manager Result Codes" (page 35). If a font reference and intrinsic style are not found, the function returns a value of `kFMInvalidFontErr`.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

### Declared In

`Fonts.h`

## FMGetFontGeneration

Obtains the generation count of a font. (Deprecated in Mac OS X v10.4. Use `ATSFontGetGeneration` instead.)

```
OSStatus FMGetFontGeneration (
   FMFont iFont,
   FMGeneration *oGeneration
);
```

### Parameters

*iFont*
A font reference.

*oGeneration*
On output, a pointer to a value that specifies the generation count of the font.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetFontTable

Retrieves all or part of a data table for a font. (Deprecated in Mac OS X v10.4. Use `ATSFontGetTable` or `CTFontCopyTable` instead.)

```
OSStatus FMGetFontTable (
   FMFont iFont,
   FourCharCode iTag,
   ByteOffset iOffset,
   ByteCount iLength,
   void *iBuffer,
   ByteCount *oActualLength
);
```

**Parameters**

*`iFont`*
: A font reference.

*`iTag`*
: A tag that identifies the data table for a font.

*`iOffset`*
: An offset to font table data you want to retrieve. The offset is relative to the beginning of the data table and is zero-based.

*`iLength`*
: The size of the data buffer (`ioBuffer`) you allocate.

*`iBuffer`*
: A pointer to the buffer used to store a copy of the font table. On input, pass `NULL` if you want to obtain only the length of the table, not its contents.

*`oActualLength`*
: On output, the actual length of the font table.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Discussion**
You should call the function `FMGetFontTable` twice. First, call it to retrieve the length of the font table. Then, after you've allocated space for the `iBuffer` parameter, call the function a second time to obtain the contents of the font table.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetFontTableDirectory

Obtains the table directory for a font. (Deprecated in Mac OS X v10.4.)

```
OSStatus FMGetFontTableDirectory (
   FMFont iFont,
   ByteCount iLength,
   void *iBuffer,
   ByteCount *oActualLength
);
```

**Parameters**

*iFont*

A font reference.

*iLength*

The number of bytes in the buffer used to store a copy of the font table directory associated with the font.

*iBuffer*

A pointer to the buffer used to store a copy of the font table directory. On input, pass `NULL` if you want to obtain only the length of the table directory, not its contents.

*oActualLength*

On output, the length of the font table directory.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35).

**Discussion**
You should call the function `FMGetFontTableDirectory` twice. First, call it to retrieve the length of the font table directory. Then, after you've allocated space for the `iBuffer` parameter, call the function a second time to obtain the contents of the table directory.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetNextFont

Obtains the next font reference. (Deprecated in Mac OS X v10.4. Use `ATSFontIteratorNext` instead.)

```
OSStatus FMGetNextFont (
   FMFontIterator *ioIterator,
   FMFont *oFont
);
```

**Parameters**

*ioIterator*

A pointer to a font iterator you created using the function `FMCreateFontIterator`.

*oFont*

A pointer to a font reference. On output, points to the next font reference obtained by the font iterator. You are responsible for allocating the memory for the font reference.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35). If there are no more font objects to get, the function `FMGetNextFont` returns the result code `kFMIterationCompleted`.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMGetNextFontFamily

Obtains the next font family reference. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyIteratorNext` instead.)

```
OSStatus FMGetNextFontFamily (
   FMFontFamilyIterator *ioIterator,
   FMFontFamily *oFontFamily
);
```

**Parameters**

*ioIterator*

A pointer to a font family iterator you created using the function `FMCreateFontFamilyIterator`.

*oFontFamily*

A pointer to a font family reference. On output, points to the font family reference obtained by the iterator. You are responsible for allocating memory for the font family reference.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35). If there are no more font family references to get, the function `FMGetNextFontFamily` returns the result code `kFMIterationCompleted`.

**Discussion**

If any changes are made to the font database while you are using the font family iterator, the iterator is invalidated and the function `FMGetNextFontFamily` returns the error `kFMIteratorScopeModified`. To remedy this error, your application must either restart or cancel the enumeration by calling the `FMResetFontFamilyIterator` (page 56) or the `FMDisposeFontFamilyIterator` (page 42) functions.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FMGetNextFontFamilyInstance

Obtains the next instance associated with a font family reference. (Deprecated in Mac OS X v10.4.)

```
OSStatus FMGetNextFontFamilyInstance (
   FMFontFamilyInstanceIterator *ioIterator,
   FMFont *oFont,
   FMFontStyle *oStyle,
   FMFontSize *oSize
);
```

**Parameters**

*`ioIterator`*
A pointer to a font family instance iterator you created with the function `FMCreateFontFamilyInstanceIterator` (page 38).

*`oFont`*
A pointer to a font reference. On output, points to the font reference obtained by the iterator. You are responsible for allocating the memory for the font reference.

*`oStyle`*
A pointer to a font style. On output, points to the font style obtained by the iterator. You are responsible for allocating the memory for the font style.

*`oSize`*
A pointer to a font size. On output, points to the font size obtained by the iterator. You are responsible for allocating the memory for the font size.

**Return Value**
A result code. See "Font Manager Result Codes" (page 35). If there is no more font family information to retrieve, the function `FMGetNextFontFamlyInstance` returns the status code `kFMIterationCompleted`.

**Discussion**
Instances are not necessarily retrieved in the order they are listed in a Font Association Table.

If any changes are made to the font database while your application is using the font family instance iterator, the iterator is invalidated and the function `FMGetNextFontFamilyInstance` (page 55) returns the error `kFMIteratorScopeModified`. To remedy this error, your application must either call the `FMResetFontFamilyInstanceIterator` (page 56) or the `FMDisposeFontFamilyInstanceIterator` (page 42) functions.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

### FMResetFontFamilyInstanceIterator

Resets the a font family instance iterator to the beginning of the iteration for the specified font family. (Deprecated in Mac OS X v10.4.)

```
OSStatus FMResetFontFamilyInstanceIterator (
   FMFontFamily iFontFamily,
   FMFontFamilyInstanceIterator *ioIterator
);
```

#### Parameters

*iFontFamily*

A font family reference.

*ioIterator*

A pointer to a font family instance iterator you created with the function `FMCreateFontFamilyInstanceIterator`.

#### Return Value

A result code. See "Font Manager Result Codes" (page 35).

#### Discussion

Once you have created a font family instance iterator, you can reuse it by calling the function `FMResetFontFamilyInstanceIterator`. This function sets the `iFontFamily` parameter to the new font family object you specify, and repositions the iterator so it is ready to get the first font family instance when you call the function `FMGetNextFontFamilyInstance` (page 55).

#### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

#### Declared In

`Fonts.h`

### FMResetFontFamilyIterator

Resets a font family iterator to the beginning of the iteration. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyIteratorReset` instead.)

```
OSStatus FMResetFontFamilyIterator (
   const FMFilter *iFilter,
   void *iRefCon,
   OptionBits iOptions,
   FMFontFamilyIterator *ioIterator
);
```

#### Parameters

*iFilter*

A pointer to a filter specification. Pass `NULL` if you want to access all font family objects within the scope of your iteration. Otherwise, you can use this parameter to restrict the scope of the iteration to the font families that match a generation count or criteria you specify in a custom filter function.

*iRefCon*

An arbitrary 32-bit value specified by your application. If you are using a custom filter function, you can use this parameter to pass data to the custom filter function. If you are not using a custom filter function, pass `NULL`.

*iOptions*

A value that specifies the scope to which the font family iterator applies. If you want the Font Manager to apply the font family iterator only to the fonts accessible from your application use the `kFMLocalIterationScope` constant. If you want the Font Manager to apply the font family iterator to all fonts registered with the system use the constant `kFMGlobalIterationScope`.

*iOIterator*

A pointer to a font family iterator you created with the function `FMCreateFontFamilyIterator`. On output, the font family iterator is reset.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Discussion**

Once you have created a font family iterator, you can reuse it by calling the function `FMResetFontFamilyIterator` (page 56). This function sets the parameters to the new values you specify, and repositions the iterator so it is ready to get the first font family reference when you call the function `FMGetNextFontFamily` (page 54).

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMResetFontIterator

Resets a font iterator to the beginning of the iteration. (Deprecated in Mac OS X v10.4. Use `ATSFontIteratorReset` instead.)

```
OSStatus FMResetFontIterator (
   const FMFilter *iFilter,
   void *iRefCon,
   OptionBits iOptions,
   FMFontIterator *ioIterator
);
```

**Parameters**

*iFilter*

A pointer to font filter specification. Pass `NULL` if you want to access all font objects within the scope of your iteration. Otherwise, you can use this parameter to restrict the scope of the iteration to font information that matches a technology, font container, or criteria you specify in a custom filter function. Pass the filter selector constant `kFMFontTechnologyFilterSelector` to select a font technology filter, the constant `kFMFontContainerFilterSelector` to select a font container filter, or the constant `kFMFontCallbackFilterSelector` to select a custom filter. See `FMFilterSelector` in the ATS Types Reference for more information on these constants.

*iRefCon*

An arbitrary 32-bit value specified by your application. If you are using a custom filter function, you can use this parameter to pass data to the custom filter function. If are not using a custom filter function, pass `NULL`.

*iOptions*

A value that specifies the scope to which the font iterator applies. If you want the Font Manager to apply the font iterator only to the fonts accessible from your application use the `kFMLocalIterationScope` constant. If you want the Font Manager to apply the font iterator to all fonts registered with the system use the constant `kFMGlobalIterationScope`.

*ioIterator*

A pointer to a font iterator you created with the function `FMCreateFontIterator` (page 40). On output, the font iterator is not positioned on a font object, and any information about font objects that were returned previously in the font iterator is no longer available.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Discussion**

Once you have created a font iterator, you can reuse it by calling the function `FMResetFontIterator` (page 57). This function sets the parameters to the new values you specify, and repositions the iterator so it is ready to get the first font object when you call the function `FMGetNextFont` (page 53).

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## FMSwapFont

Returns a pointer to the font output structure for a specified font. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
FMOutPtr FMSwapFont (
   const FMInput *inRec
);
```

**Parameters**

*inRec*

A pointer to the font input structure for which you want to obtain font output information. A font input structure contains the font family ID, the style requested, scaling factors, and other information that specifies the characteristics of the font that is requested.

**Return Value**

A pointer to a font output structure (`FMOutput`). The font output structure contains a handle to the font resource for the specified input font, along with information about the font, such as the ascent, descent, and leading measurements.

**Discussion**

The function `FMSwapFont` is typically called by QuickDraw and other parts of the system software to access font handles. QuickDraw calls the `FMSwapFont` function every time a QuickDraw text function is used.

In most cases you don't need to call this function. If you want to call the `FMSwapFont` function to get a handle to a font resource or information about a font, you must first create a font input structure and fill it with the appropriate information. You can use the pointer returned by `FMSwapFont` to access the font output structure. You cannot assume that the font resource pointed to by the `fontHandle` field of the font output structure returned by this function is of any particular type, such as `'NFNT'` or `'sfnt'`. If you need to access specific information in the font resource, call the Resource Manager function `GetResInfo` with the handle returned in the font output structure to determine the font resource type.

The pointer to the font output structure returned by the function `FMSwapFont` points to a structure allocated in low memory by the Font Manager. The same structure is reused for each call made to `FMSwapFont`. Do not free the memory allocated for this structure.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## FontMetrics

Obtains fractional measurements for the font, size, and style specified in the current graphics port. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
void FontMetrics (
   FMetricRecPtr theMetrics
);
```

**Parameters**

*`theMetrics`*
A pointer to a font metrics structure. On output, the structure contains the font measurement information in fractional values.

**Discussion**
The `FontMetrics` function obtains measurements for the ascent, descent, leading, and width of the largest glyph in the font for the font, size, and style specified in the current graphics port.

The font metrics structure (of data type `FMetricRec`) contains a handle to the global width table, which in turn contains a handle to the associated font family resource for the current font (the font in the current graphics port). It also contains the values of four measurements for the current font.

The `FontMetrics` function is similar to the QuickDraw function `GetFontInfo` except that `FontMetrics` returns fractional values for greater accuracy in high-resolution printing. The `FontMetrics` function also does not take into account any additional widths that are added by QuickDraw when it applies styles to the glyphs in a font.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## GetAppFont

Returns the font family ID of the current application font. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
short GetAppFont (
   void
);
```

**Return Value**
The font family ID of the current application font. This is the font family ID that has been mapped to 1 by the system software.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## GetDefFontSize

Determines the default size of the system font. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
short GetDefFontSize (
   void
);
```

**Return Value**
The the default font size of the system font.

**Discussion**
You can determine the preferred size for either the system font or the application font of any enabled script system by calling the Script Manager function `GetScriptManagerVariable`.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## GetFNum

Obtains the font family ID for a specified font family name. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyFindFromName` instead.)

Not recommended.

```
void GetFNum (
   ConstStr255Param name,
   short *familyID
);
```

**Parameters**

*name*

The font family name.

*familyID*

On output, a pointer to the font family ID for the font family specified in `name`. If the font specified in the parameter `name` does not exist, the font family ID contains 0.

**Carbon Porting Notes**

You should use the function `FMGetFontFamilyFromName` instead of the function `GetFNum`.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Related Sample Code**

Simple DrawSprocket

**Declared In**

`Fonts.h`

## GetFontName

Obtains the name of a font family that has a specified family ID number. (Deprecated in Mac OS X v10.4. Use `ATSFontFamilyGetName` instead.)

Not recommended.

```
void GetFontName (
   short familyID,
   Str255 name
);
```

**Parameters**

*familyID*

The font family ID.

*name*

On output, this parameter contains the font family name for the font family specified in `familyID`. If the font specified in the `familyID` parameter does not exist, `name` contains an empty string.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## GetOutlinePreferred

Obtains the current preference for whether outline or bitmapped fonts are returned when the Font Manager receives a font request. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
Boolean GetOutlinePreferred (
   void
);
```

**Return Value**
The value of the Font Manager's current preference for outline or bitmapped fonts. If `GetOutlinePreferred` returns `TRUE`, then the Font Manager will return an outline font when both an outline font and a bitmapped font are available for a particular request. If `GetOutlinePreferred` returns `FALSE`, then the Font Manager will return the bitmapped font when both types are available. See the Debugger Services documentation for a description of the `Boolean` data type.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## GetPreserveGlyph

Determines whether the Font Manager preserves the shapes of glyphs from outline fonts. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
Boolean GetPreserveGlyph (
   void
);
```

**Return Value**
A `Boolean` value that indicates whether the Font Manager preserves the shapes of glyphs from outline fonts. Your application can set the value of this variable with the `SetPreserveGlyph` function. If `GetPreserveGlyph` returns `TRUE`, the Font Manager preserves glyph shapes; if `GetPreserveGlyph` returns `FALSE`, then the Font Manager scales glyphs to fit between the ascent and descent lines for the font in use.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## GetSysFont

Obtains the font family ID of the current system font. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
short GetSysFont (
   void
);
```

**Return Value**
The current value of the font family ID of the current system font. This is the font family ID that has been mapped to 0 by the system software.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## IsAntiAliasedTextEnabled

Checks whether antialiased text is enabled. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
Boolean IsAntiAliasedTextEnabled (
   SInt16 *oMinFontSize
);
```

**Parameters**

*`oMinFontSize`*
    On output, points to the minimum font size for which antialiasing is enabled.

**Return Value**
Returns `true` if antialiased text is enabled; `false` otherwise.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## IsOutline

Determines whether the specified scaling factors will cause the Font Manager to choose an outline font for the current graphics port. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
Boolean IsOutline (
   Point numer,
   Point denom
);
```

**Parameters**

*numer*

The numerators of the vertical and horizontal scaling factors. The `numer` parameter is of type `Point`, and contains two fields: `h` (the numerator of the ratio for horizontal scaling) and `v` (the numerator of the ratio for vertical scaling).

*denom*

The denominators of the vertical and horizontal scaling factors. The `denom` parameter is of type `Point`, and contains two fields: `h` (the denominator of the ratio for horizontal scaling) and `v` (the denominator of the ratio for vertical scaling).

**Return Value**

Returns `TRUE` if the Font Manager will choose an outline font for the current graphics port.

**Discussion**

The Font Manager uses the font scaling factors specified in the `numer` and `denom` parameters as well as the current preference (as set by the `SetOutlinePreferred` function) as criteria to choose which font to use.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## OutlineMetrics

Obtains font measurements for a block of text to be drawn in a specified outline font. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
OSErr OutlineMetrics (
   short byteCount,
   const void *textPtr,
   Point numer,
   Point denom,
   short *yMax,
   short *yMin,
   FixedPtr awArray,
   FixedPtr lsbArray,
   RectPtr boundsArray
);
```

**Parameters**

*byteCount*

The number of bytes in the block of text that you want measured.

*textPtr*

A pointer to the block of text that for which you want to obtain font measurements.

*numer*

The numerators of the vertical and horizontal scaling factors. The `numer` parameter is of type `Point`, and contains two fields: `h` (the numerator of the ratio for horizontal scaling) and `v` (the numerator of the ratio for vertical scaling). The Font Manager applies these scaling factors to the current font when calculating the measurements for glyphs in the block of text.

*denom*

The denominators of the vertical and horizontal scaling factors. The `denom` parameter is of type `Point`, and contains two fields: `h` (the denominator of the ratio for horizontal scaling) and `v` (the denominator of the ratio for vertical scaling). The Font Manager applies these scaling factors to the current font when calculating the measurements for glyphs in the block of text.

*yMax*

On output, a pointer to the maximum y-value for the text. Pass `NULL` in this parameter if you don't want this value returned.

*yMin*

On output, a pointer to the minimum y-value for the text. Pass `NULL` in this parameter if you don't want this value returned.

*awArray*

A pointer to an array. On output the array is filled with the advance width measurements for the glyphs being measured. These measurements are in pixels, based on the point size and font scaling factors of the current font. There is an entry in this array for each glyph that is being measured.

The `awArray` parameter is of type `FixedPtr`. The `FixedPtr` data type is a pointer to an array, and each entry in the array is of type `Fixed`, which is 4 bytes in length. Multiply `byteCount` by 4 to calculate the memory you need in bytes.

If the `FractEnable` global variable has been set to `TRUE` through the `SetFractEnable` function, the values in `awArray` have fractional character widths. If `FractEnable` has been set to `FALSE`, the Font Manager returns integer values for the advance widths, with 0 in the decimal part of the values.

*lsbArray*

A pointer to an array. On output the array is filled with the left-side bearing measurements for the glyphs being measured. The measurements are in pixels, based on the point size of the current font. There is an entry in this array for each glyph that is being measured.

The `lsbArray` parameter is of type `FixedPtr`. The `FixedPtr` data type is a pointer to an array, and each entry in the array is of type `Fixed`, which is 4 bytes in length. Multiply `byteCount` by 4 to calculate the memory you need in bytes.

The fractional portion of left-side bearing values are retained.

*boundsArray*

A pointer to an array. On output the array is filled with the bounding boxes for the glyphs being measured. Bounding boxes are the smallest rectangles that fit around the pixels of the glyph. There is an entry in this array for each glyph that is being measured.

The coordinate system used to describe the bounding boxes is in pixel units, centered at the glyph origin, and with a vertical positive direction upwards. This is the opposite of the QuickDraw vertical orientation.

The `boundsArray` parameter is of type `RectPtr`. The `RectPtr` data type is a pointer to QuickDraw's `Rect` data type, which is 8 bytes in length. Multiply `byteCount` by 8 to calculate the memory you need in bytes. Allocate the memory needed for the array and pass a pointer to the array in the `boundsArray` parameter.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Discussion**
The `OutlineMetrics` function computes the maximum y-value, minimum y-value, advance widths, left-side bearings, and bounding boxes for a block of text. It uses the font, size, and style specified in the current graphics port. You can use these measurements when laying out text. You may need to adjust line spacing to accommodate exceptionally large glyphs.

The `OutlineMetrics` function works for outline fonts only and is the preferred method for measuring text that is drawn with an outline font.

When you are using `OutlineMetrics` to compute advance width values, left-side bearing values, or bounding boxes, you need to bear in mind that when a text block contains 2-byte characters, not every byte in the `awArray`, `lsbArray`, and `boundsArray` structures is used. Each of these arrays is indexed by the glyph index; thus, if you have five characters in a string, only the first five entries in each array contains a value. Call the Script Manager function `CharByte` to determine how many characters there are in the text block, and ignore the unused array entries (which occur at the end of each array).

If you don't want `OutlineMetrics` to compute one of these three values, pass `NULL` in the applicable parameter. Otherwise, allocate the amount of memory needed for the array and pass a pointer to it in this parameter.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## QDTextBounds

Obtains a rectangle that specifies the bounds of QuickDraw text. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
void QDTextBounds (
   short byteCount,
   const void *textAddr,
   Rect *bounds
);
```

**Parameters**

*`byteCount`*
The number of bytes in the buffer that contains the text whose bounds you want to obtain.

*`textAddr`*
A pointer to a buffer that contains the text whose bounds you want to obtain. You must allocate this buffer.

*`bounds`*
On output, points to a rectangle that specifies the bounds of QuickDraw text.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## RealFont

Determines whether a font is available or is intended for use in a specified size. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
Boolean RealFont (
   short fontNum,
   short size
);
```

**Parameters**

*`fontNum`*

The font family ID.

*`size`*

The font size requested.

**Return Value**
Returns `TRUE` if the requested size of the font is available. The function `RealFont` first checks for a bitmapped font from the specified family. If one is not available, `RealFont` checks next for an outline font. If neither kind of font is available, `RealFont` returns `FALSE`.

**Discussion**
If an outline font exists for the requested font family, `RealFont` normally considers the font to be available in any requested size. However, the font designer can include instructions in the font that outlines should not be used at certain point sizes, in which case the `RealFont` function considers the font unavailable and returns `FALSE`. The Font Manager determines whether the size is valid by testing the value of the smallest readable size element of the font family header table.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## SetAntiAliasedTextEnabled

Enables or disables antialiased text for an application. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
OSStatus SetAntiAliasedTextEnabled (
   Boolean iEnable,
   SInt16 iMinFontSize
);
```

**Parameters**

*`iEnable`*

A `Boolean` value. Pass `true` to enable antialiased text or `false` to disable it.

*iMinFontSize*

A integer of type `SInt16` that specifies the minimum font size to which antialiasing should be enabled.

**Return Value**

A result code. See "Font Manager Result Codes" (page 35).

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## SetFractEnable

Enables or disables fractional glyph widths. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
void SetFractEnable (
   Boolean fractEnable
);
```

**Parameters**

*fractEnable*

Specifies whether fractional widths or integer widths are to be used to determine glyph measurements. A value of `TRUE` indicates fractional glyph widths; a value of `FALSE` indicates integer glyph widths.

The `SetFractEnable` function assigns the value that you specify in the `fractEnable` parameter to the global variable `FractEnable`.

**Discussion**

The `SetFractEnable` function establishes whether or not the Font Manager provides fractional glyph widths to QuickDraw, which then uses them for advancing the pen during text drawing.

The Font Manager defaults to integer widths to ensure compatibility with existing applications. When fractional glyph widths are enabled, the Font Manager can determine the locations of glyphs more accurately than is possible with integer widths.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## SetFScaleDisable

Enables or disables the computation of font scaling factors by the Font Manager for bitmapped glyphs. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
void SetFScaleDisable (
   Boolean fscaleDisable
);
```

**Parameters**

*fscaleDisable*

Specifies whether bitmapped fonts are to be scaled. A value of `TRUE` indicates that font scaling is disabled; a value of `FALSE` indicates that font scaling is enabled.

If you set the `fscaleDisable` parameter to `TRUE`, the Font Manager disables font scaling, which means it responds to a request for a font size that is not available by computing font scaling factors of 1/1 and returning a smaller, unscaled bitmapped font with the widths of the requested size. If you set the `fscaleDisable` parameter to `FALSE`, the Font Manager computes scaling factors for bitmapped fonts.

**Discussion**

QuickDraw performs the actual scaling of glyph bitmaps for bitmapped fonts by using the font scaling factors computed and returned by the Font Manager.

When font scaling is enabled, the Font Manager can scale a bitmapped glyph that is present in the System file to imitate the appearance of a bitmapped glyph in another point size that is not present. By default, the Font Manager scales fonts to ensure compatibility with existing applications.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

**Declared In**

`Fonts.h`

## SetOutlinePreferred

Sets the preference for whether to use bitmapped or outline fonts when both kinds of fonts are available. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
void SetOutlinePreferred (
   Boolean outlinePreferred
);
```

**Parameters**

*outlinePreferred*

Specifies whether the Font Manager chooses an outline font or a bitmapped font when both are available to fill a font request. A value of `TRUE` indicates an outline font; a value of `FALSE` indicates a bitmapped font.

If you want the Font Manager to choose outline fonts over any bitmapped font counterparts, set the `outlinePreferred` parameter to `TRUE`. If you want it to choose bitmapped fonts, set the `outlinePreferred` parameter to `FALSE`.

**Discussion**

If an outline font and a bitmapped font are both available for a font request, the default behavior for the Font Manager is to choose the bitmapped font, in order to maintain compatibility with documents that were created on computer systems on which outline fonts were not available. The `SetOutlinePreferred` function sets the Font Manager's current preference for either bitmapped or outline fonts when both are available.

If only outline fonts are available, the Font Manager chooses them regardless of the value of the `outlinePreferred` parameter. If only bitmapped fonts are available, they are chosen. The Font Manager chooses bitmapped versus outline fonts on a size basis, before it takes stylistic variations into account, which can lead to unexpected results.

The preference you set is valid only during the current session with your application. The `outlinePreferred` parameter does not set a global variable.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

**Declared In**
`Fonts.h`

## SetPreserveGlyph

Temporarily changes the default behavior of the Font Manager, so that it does not scale oversized glyphs. (Deprecated in Mac OS X v10.4. There is no replacement function.)

```
void SetPreserveGlyph (
   Boolean preserveGlyph
);
```

**Parameters**

*`preserveGlyph`*
Specifies whether or not glyphs from an outline font are scaled to fit between the ascent and descent lines. If you set the value of the `preserveGlyph` parameter to `TRUE`, the measurements of all glyphs are preserved, which means that your application may have to alter the leading between lines in a document if some of the glyphs extend beyond the ascent or descent lines. If you set the value of the `preserveGlyph` parameter to `FALSE`, all glyphs are scaled to fit between the ascent and descent lines.

**Discussion**
The `SetPreserveGlyph` function establishes how the Font Manager treats glyphs that do not fit between the ascent and descent lines for the current font. The default behavior for the Font Manager is to scale a glyph from an outline font so that it fits between the ascent and descent lines; however, this alters the appearance of the glyph.

You can determine the current behavior of the Font Manager in this regard by calling the `GetPreserveGlyph` function. To ensure that documents have the same appearance whenever they are opened, you need to call `GetPreserveGlyph` and save the value that it returns with your documents and restore it each time a document is displayed by your application.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

**Declared In**
`Fonts.h`

# Deprecated in Mac OS X v10.5

### FMGetGeneration

Retrieves the value of the generation count. (Deprecated in Mac OS X v10.5. Use `ATSGetGeneration` instead.)

```
FMGeneration FMGetGeneration (
   void
);
```

**Return Value**
The generation count. See the ATS Types documentation for a description of the `FMGeneration` data type.

**Discussion**
Any operation that adds, deletes, or modifies one or more font families or fonts triggers an update of the global generation count. You can use this function in conjunction with the iteration functions to identify changes made to the font database.

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

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**
`Fonts.h`

# Document Revision History

This table describes the changes to *Font Manager Reference*.

| Date | Notes |
|---|---|
| 2007-12-11 | Added deprecation and replacement technology information for functions deprecated in Mac OS X v10.4 and v10.5. |
| 2006-07-13 | Made formatting changes. |
| 2006-07-24 | Added information on deprecated functions. |
| 2006-03-08 | Minor content update. |
|  | Added description of the function `FMFontGetCGFontRefFromFontFamilyInstance` (page 11). |
| 2006-01-10 | Fixed typographical errors. |
| 2005-11-09 | Corrected a typographical error. |
| 2003-06-16 | Corrected a typographical error in the title meta tag. |
| 2002-12-04 | Updated formatting; added additional information about the function `FMGetFontFamilyFromName` (page 46). |
| 2002-04-01 | Major document revision. |

# Index