

This chapter describes font objects and the functions you can use to manipulate them. Read this chapter if you use any kind of font object for the QuickDraw GX shapes you create.

Before reading this chapter, you should be familiar with the information in the chapter “Introduction to QuickDraw GX Typography” in this book. You should also be familiar with the information discussed in *Inside Macintosh: QuickDraw GX Objects*.

This chapter introduces QuickDraw GX font objects and describes their properties. It then shows you how to create, dispose of, and manipulate these objects to

- gain access to the system’s font list
- specify in a style object how a font should be used—for example, specify encodings, variations, and font features
- edit a font and add your own fonts

This chapter also lists and cross-references font-related QuickDraw GX functions that are described elsewhere in this book and in other parts of *Inside Macintosh*.

About Font Objects

Fonts are represented in QuickDraw GX as **font objects**. To draw or print text, you must reference or use a font object.

Fonts come in a variety of formats and can be stored in many different ways. In QuickDraw GX, fonts are consolidated into a single object type that hides the complexity of the font data from your application. With QuickDraw GX, you can have a single object type, as well as a single set of methods of accessing the font data.

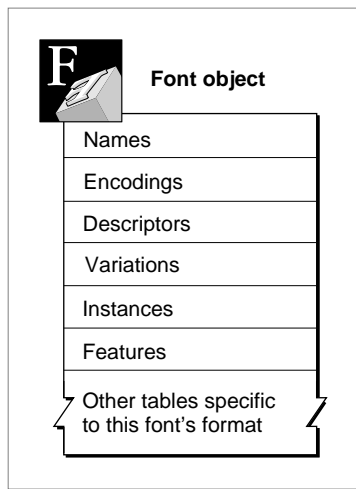
Each QuickDraw GX font object encapsulates a particular style of text—for example, Times Bold or Times Italic. The same font object is used for all point sizes.

A collection of QuickDraw GX font objects that share a common design is known as a **font family**—for example, Times, which is composed of Times Roman, Times Bold, Times Italic, and Times Bold Italic. By grouping all such fonts into a font family, QuickDraw GX allows your application to display only family names in the Font menu, thereby simplifying for the user the process of choosing a font.

The interface to each QuickDraw GX font object is entirely procedural—you cannot access any information in the fonts directly. To manipulate the pieces of information in the font object—for example, its properties—you must use QuickDraw GX functions.

Font Object Properties

QuickDraw GX font objects have six accessible properties, as shown in Figure 7-1. Note that, because a font is an object and not a data structure, the order of the properties as shown in Figure 7-1 is completely arbitrary.

Figure 7-1 The QuickDraw GX font object and its accessible properties

The six accessible properties include

- **Names.** Text in a font that contains information about the font.
- **Encodings.** Internal conversion tables for interpreting a specific character set.
- **Descriptors.** Identifying characteristics used to measure the stylistic attributes of one font in comparison to another font—for example, whether one font is “bolder” than another.
- **Variations.** A setting along an axis. This can be the range, from minimum to maximum, of each variation axis in a font.
- **Instances.** A named font variation coordinate.
- **Features.** A type that holds information about a font feature.

Every QuickDraw GX font object contains font names and encodings. The other properties shown in Figure 7-1 offer additional functionality but may not be present in every font. In addition to these six accessible properties, a QuickDraw GX font object contains other properties specific to the font format, depending on whether the format is TrueType GX, Type 1 GX, or 'NFNT', or another format.

QuickDraw GX provides functions for manipulating each of these font object properties. These functions are described in the section “Font Objects Reference” beginning on page 7-21.

Names

Each QuickDraw GX font object contains a set of font names. **Font names** provide specific information about a font, such as its family name, style, copyright date, version, and manufacturer. Some font names can be used to build menus in your application—for example, family and style—whereas other names are used to identify the font

uniquely—for example, the unique and PostScript names. Here are some examples of font names with predefined selectors:

- **Font family.** This name is shared by all styles in a family. An example is “New York”.
- **Style.** This stylistic variation distinguishes a font from other members of the same family. Examples are “Regular”, “Italic”, “Bold”, or “Black”.
- **Unique name.** This is the manufacturer’s name for the font. An example is “Apple Computer New York Black 3.0 8/10/92”.
- **Full font name.** An example is “New York Black”.
- **Copyright.** This is the manufacturer’s copyright notice. An example of this is “© Apple Computer, Inc. 1993.”
- **Version.** This is the font manufacturer’s version number. An example is “3.0.”
- **PostScript name.** This is the PostScript-legible name of the font. An example is “NewYork-Black”.
- **Trademark.** This the trademark holder’s name. An example is “Palatino is a registered trademark of Linotype AG”.
- **Manufacturer.** This is the font manufacturer’s name. An example is “Apple Computer, Inc.”

Note

In addition to predefined name selectors, a font may contain other names that describe parts of the font, such as ligatures or swashes. However, these names do not have predefined name selectors. You can gain access to these names through QuickDraw GX features and variations, which are discussed later in this section. ♦

Font names are identified by a font’s platform, script, and language, which are discussed in the next section.

A QuickDraw GX font object can have font names in any of the languages specified by the `gxFontLanguage` enumeration. For example, a font manufacturer can store the English name of a font as “Geneva Bold” and the French name of the font as “Geneva Gras.” The currently defined languages are listed in “Languages” beginning on page 7-28.

Encodings

Each QuickDraw GX font object contains a certain number of **character encodings**. Each encoding is an internal conversion table for interpreting a specific character set—that is, a way to map a character code to a glyph code for that font.

Like font names, font encodings are also identified by platform, script, and language. A **platform** describes what standard the character set was designed for, such as the Macintosh computer or the Unicode standard. Possible platform values are listed in “Font Platforms” on page 7-25. A custom platform is one whose encoding does not correspond to a specific standard. A **script** is a writing system, or a set of characters with basic rules of use in creating a visual depiction of one or many languages. For example, the Arabic script can be used to depict written Arabic, Egyptian, or classical Arabic. Each script has a

Font Objects

unique set of attributes. Roman script has a general left-to-right direction of text, whereas Arabic script has a general right-to-left direction of text. Printed Roman characters are relatively independent of each other; Arabic glyphs change shape depending on the glyphs that surround them. Some writing systems, such as Roman and Cyrillic, are basically **alphabetic**: glyphs symbolize discrete phonemic elements in the language. Other writing systems, including Japanese Kana, are **syllabic**: the glyphs stand for syllables in the language. Still other writing systems—namely, Japanese Kanji, Chinese Hanzi, and Korean Hanja—include **ideographic** glyphs. In these systems, glyphs do not represent pronunciation alone but are related to the component meanings of words. A typical character set for an ideographic writing system can be quite large, ranging from 7000 to more than 30,000 characters. A written **language** refers to the whole body of written words—and methods of combining words to create meaning—used by a particular group of people.

Figure 7-2 shows examples of alphabetic, syllabic, and ideographic representations of characters.

Figure 7-2 Words with alphabetic, syllabic, and ideographic characters

Alphabetic	Syllabic	Ideographic
English	にほんご	中 文

In general, the encoding of a font won't be identified by a specific language because platform and script are enough to identify a character encoding. The exception to this rule occurs in a few Macintosh scripts, where two or more languages in the same script represent two different character sets. For example, Turkish and Croatian are both languages in the Roman script but have different encodings.

Character Code Sizes

QuickDraw GX supports character code sizes larger than 7-bit ASCII codes (0-127). These include

- 16-bit character codes, as used by Microsoft and Unicode, in which each character is stored as a 16-bit number
- a mixed 8-/16-bit encoding, as used by Chinese, Japanese, and Korean scripts, in which certain byte values are set aside to signal the first byte of a 2-byte character. The value of the first byte specifies whether you include the value of the next byte as well.
- 8-bit character codes, where each character is stored as an 8-bit number

Font Objects

You can use a font’s encoding to find out what character code size it uses. Because each encoding table in a font is identified by platform and script, each combination of platform and script specifies the size of the character code that the table can translate. Depending on platform and script, your application may need to generate character codes of different sizes.

Table 7-1 enumerates the platforms and scripts that QuickDraw GX supports in relation to character code sizes. Note that all are either 8- or 16-bit, but some scripts—such as Chinese, Japanese, and Korean—use a mixed 8-/16-bit encoding.

Table 7-1 Character code sizes among various platforms and scripts

Character code size (bits)	Platform	Script
16	Unicode	(All Unicode versions)
16	Custom	Custom 16-bit script
16	Microsoft	(All Microsoft scripts)
mixed 8/16	Macintosh	Chinese
mixed 8/16	Macintosh	Korean
mixed 8/16	Macintosh	Japanese
mixed 8/16	Macintosh	Simplified Chinese
mixed 8/16	Custom	Custom 8-bit/16-bit script
8	Macintosh	(Any Macintosh script not already listed previously)
8	Custom	Custom 8-bit script

To find out what encodings a font supports, you use the `GXGetFontEncoding` function, described on page 7-44. To search for all the fonts that support a given encoding, you use the `GXFindFonts` function, described on page 7-33.

Font Descriptors

Each font object within a family has a certain number of identifiable characteristics called **font descriptors**, and these define such attributes in a font as its weight, width, italic slant and optical point size—that is, the point size for which the font was designed. A font descriptor is a data structure that allows your application to read and measure the stylistic attributes of one font versus another font—to determine, for example, if one font is “bolder” than another.

Font descriptors give numerical values for these stylistic attributes.

Table 7-2 lists some predefined descriptors. For more information about ornamental sets, see the chapter “Layout Styles” in this book.

Font Objects

Table 7-2 A list of predefined font descriptors

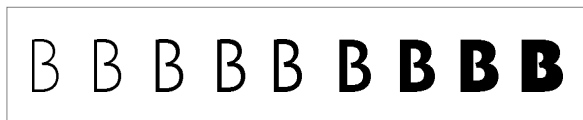
Descriptor	Descriptor tag	Description
Weight	'wght'	The thickness of the line used to draw a glyph of the font. A plain font might have a value of 1.0.
Width	'wdth'	The width or narrowness of the glyphs. A plain font might have a value of 1.0.
Italic slant	'slnt'	The number of degrees, from -90 to 90, by which the glyphs lean. A plain font might have a value of 0 degrees.
Optical point size	'opsz'	The point size for which the font was designed. A common value might be 12.0 points.
Nonalphabetic	'nalf'	The value corresponding to a nonalphabetic form provided by a font. The values are identical to those for ornamental sets: dingbats equal 1, Pi characters equal 2, fleurons equal 3, decorative borders equal 4, international symbols equal 5, and math symbols equal 6. An undefined value or a value of 0 indicates that the font is alphabetic.

Font Variations

For most fonts, a single list of descriptors is enough to describe its appearance. However, some fonts are capable of generating a wide range of stylistic changes. Such a font contains **font variation** axes, each of which describes a particular stylistic attribute and the range of values that the font can use.

Font variation axes are named by the same tags used by descriptors—for example, 'wght'. Each axis has a minimum, maximum, and default value. The minimum and maximum values determine the range of values that the variation axis covers. If a font contains both a font descriptor and a font variation with the same tag, the default value of the variation is equal to the descriptor.

In Figure 7-3, the variation axis 'wght' has a default value of 1.0, and minimum and maximum values of 0.62 and 1.3. This font can create a range of glyphs of varying thicknesses—from light to bold—that your application can draw.

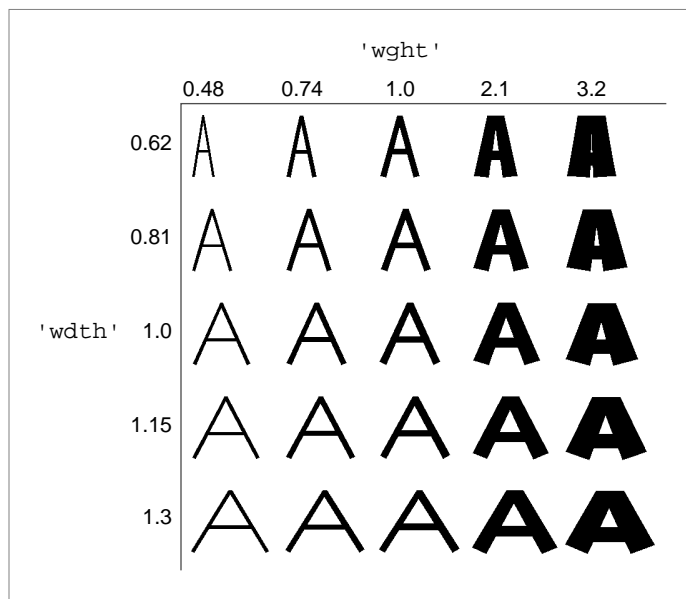
Figure 7-3 Font variations along the 'wght' variation axis

A font may also name specific values along a variation axis as font instances, described in the next section.

In addition, your application can combine multiple style coordinates. For example, a font may have a 'wght' axis and a 'wdth' axis. The user can then select any combination of bold and condensed values—such as 75 percent bold and 50 percent condensed.

Figure 7-4 shows an example of a font variation with two axes. In Figure 7-4, the weight axis has a minimum value of 0.48 and a maximum of 3.2, and the width axis has a minimum of 0.62 and a maximum of 1.3.

Figure 7-4 Font variations for the 'wght' and 'wdth' axes



QuickDraw GX provides your application with functions that allow you to count the number of font variation axes in a particular font and to specify a font variation by index or by tag (such as 'wdth') in the font's list of font variations.

Font Instances

A **font instance** is a named font variation coordinate. It is a setting identified by a type designer that includes a value for each variation axis in the font. A font instance provides that set of values with a name, which is stored in the names property of the font object. For example, if a font has a weight variation axis whose default is 1.0 and whose maximum is 1.5, a font instance might be "Demibold" with a 'wght' axis value of 1.2.

Your application can make font instances available to the user as part of the user's font and style selection mechanism, along with other font and style selections.

Font Features

QuickDraw GX fonts can include **font features**—changes to the selection of glyphs in the font, such as automatic ligature formation and cursive connections between glyphs. Some of these changes are essential for writing systems such as Arabic or Hindi that require changes to glyphs and words depending on context; some are useful simply to enhance the appearance of text. Apple Computer, Inc. has defined a standard set of font features.

Features are grouped into **types**; each feature type has some number of **feature settings**. For example, in the lowercase feature type, the settings would represent all lowercase, uppercase, or small caps. This means that in a style or shape object, you can specify features with particular settings that modify what gets drawn.

Font features allow your application to draw different versions of a letter—for example, small caps, inferiors, and ligatures.

The feature types and their feature settings are described in the chapter “Layout Styles” in this book.

Note

Feature types and feature selectors are defined and listed in the **QuickDraw GX Font Feature Registry**, a document maintained by Apple Computer, Inc. To obtain the latest version of the feature registry, please contact Apple Computer at the AppleLink address FONTREGISTRY. ♦

QuickDraw GX Font Formats

QuickDraw GX provides a single, consistent application programming interface (API) for all GX font objects. It also provides you with functions for finding a certain font and then accessing its properties rather than requiring your application to interpret the font’s data directly.

At the same time, the API gives you flexibility in your implementation of a font object—for example, in your choice of font format and methods of storing and referencing fonts. Font objects in QuickDraw GX can represent a variety of font formats, including but not limited to

- TrueType GX
- Type 1 GX
- 'NFNT'

Your application can use a font in any of these font formats. Because all fonts are accessible through the same set of functions, your application should not need to know the font’s format.

How Font Objects Are Stored and Referenced

In the QuickDraw GX architecture, each font object has a **storage type**, describing the method used to store the font. A font is stored in one of four ways:

- with a resource ID
- with a memory handle
- with a file specification
- as a 'FOND' resource

Along with its storage type, each font has a storage reference, which specifies the instance of the storage type it uses.

You can determine how a font is stored simply by checking its associated storage type, which is of the `gxFontStorageTag` type. Table 7-3 shows the storage types and what each means.

Table 7-3 QuickDraw GX storage types

Storage tag	Storage reference
<code>gxResourceFontStorage</code>	Resource handle
<code>gxHandleFontStorage</code>	Handle
<code>gxFileFontStorage</code>	File reference
<code>gxNfntFontStorage</code>	A 32-bit value with the <code>txFace</code> in the high-order 16 bits and the <code>txFont</code> in the low-order 16 bits.

Note

When QuickDraw GX is initialized, it searches the Fonts Folder for resources of type 'sfnt' and 'FOND'. Then `GXNewFont` is called with the 'sfnt' resources—after which, `GXNewFont` is called with each 'FOND' resource that does not reference any 'sfnt' resources. In this way, when your application uses QuickDraw GX, all available 'sfnt' resources and bitmap-only 'FOND' resources are already instantiated as font objects. ♦

QuickDraw GX creates a list of fonts currently available in the system and identifies each font object by a unique 4-byte ID. This font ID is the font object reference found in the font property of the style object; it is also the same as the `fontID` parameter used in the functions described in the “Font Objects Reference” section of this chapter. It is not, however, the same as the storage reference in Table 7-3.

IMPORTANT

Although this font ID is a unique reference to a font, it is *not* guaranteed to be unique across different applications, because it is regenerated each time an application is launched. Therefore, your application should never save this ID in a document; instead, it should use the font's unique name. ▲

Font Objects

Font Attributes

Each font has a set of **font attributes**, which are a group of flags that modify the behavior or identity of the font. These flags are defined in the `gxFontAttributes` enumeration.

Font Embedding

Font embedding refers to the technique of storing the font object's binary data in a document. The advantage of this technique is that the text in the document always displays the correct font, even if it is moved to another computer. The disadvantage is that this technique increases the size of the document.

Two QuickDraw GX functions implement font embedding: `GXNewFont` and `GXFlattenFont`.

The `GXNewFont` function, described on page 7-64, takes the binary data of a font and returns the font object. The `GXFlattenFont` function, described on page 7-65, takes a font object and returns its binary data.

Font Tables

The information in a font object is organized into a series of tables, each identified by a 4-byte tag. Table 7-4 shows some of the standard tables.

Table 7-4 Font tables and their contents

Storage table	Attributes
'name '	Font names
'cmap '	Font encodings
'fdesc '	Font descriptors

QuickDraw GX provides functions for looking for a specific table, iterating through the tables, and changing or deleting a table. These functions are described in “Advanced Font Functions” beginning on page 7-63.

There are other tables that are private to that font's format.

IMPORTANT

You should never alter the data in a font, including the font tables, unless your application is a font editor or otherwise needs to change the data in or a name of a font. Most applications, such as word processors, do not need to change the data in a font. ▲

The tables of a TrueType GX font are described in *QuickDraw GX Font Formats*, available from APDA. In most cases, a general-purpose application does not need to know or use the formats of the font's tables.

The List of Available Fonts

When you initialize a QuickDraw GX graphics client, QuickDraw GX creates font objects for all of the fonts in the system Font folder. These become available as a list to the application. You can access and retrieve this list of available fonts by calling the `GXFindFonts` function, described in the following section “Using Font Objects” and on page 7-33.

The Default Font

QuickDraw GX provides you with functions to manipulate the default font. When you first create a typographic shape, for example, its style is initialized to have a font reference of `nil`. If QuickDraw GX is passed a `nil` for the font parameter, it substitutes the default font—initially, Helvetica, if it is available—for that parameter. To manipulate the default font and change from Helvetica to a different font, you can use the `GXGetDefaultFont` and `GXSetDefaultFont` functions, described on page 7-35 and page 7-36, respectively.

Using Font Objects

In QuickDraw GX, you can perform a variety of operations with font objects. These operations can be grouped into the following categories:

- getting information about the available fonts in the system
- gaining access to the properties of fonts
- changing or deleting information stored in the tables of a font—but only for fonts created by your application, not those stored in the system

Note

QuickDraw GX functions that begin with `GXGet` and `GXFind` are similar in that they retrieve roughly the same information. However, the `GXGet` functions find the information by its index in the font’s list of that type of information and return or name a tag or a font name, whereas the `GXFind` functions use a tag to find the information and return its index. ♦

Getting Information About Available Fonts

If you want to get the list of available fonts, you call `GXFindFonts` first to get the number of fonts available, as shown in Listing 7-1. The function returns the number of fonts that meet the search criteria. The `GXFindFonts` function, described on page 7-33, allows you to get a list of available fonts or any subset of available fonts—from a single font to a list of font families—in the system.

Font Objects

The function copies the fonts it finds into the final parameter, which serves as the buffer references for your application. You can then create a buffer large enough to store the font references and call `GXFindFonts` again, passing it the buffer (`fontList`) to get the list.

Listing 7-1 Obtaining a list of available fonts in the system

```
/* return the number of fonts */
long count = GXFindFonts(nil, 0, 0, 0, 0, 0, nil, 1,
                        gxSelectToEnd, nil);
/* copy the fonts into the font list */
gxFont* fontList = (gxFont*)NewPtr(count * sizeof(gxFont));
GXFindFonts(nil, 0, 0, 0, 0, 0, nil, 1, count, fontList);
```

To get a reference to the second available font, you can call

```
GXFindFonts(nil, 0, 0, 0, 0, 0, nil, 0, 2, 1, &myFont);
```

To get the number of font families available, use

```
GXFindFonts(nil, gxFamilyFontName, 0, 0, 0, 0, nil, 0, 1,
            gxSelectToEnd, nil);
```

To get a reference to the second font family in the list of available font families, use

```
GXFindFonts(nil, gxFamilyFontName, 0, 0, 0, 0, nil, 0, 2, 1,
            &myFont)
```

To get information about the number of fonts available in a single font family, you call

```
count = GXFindFonts(myFontFamily, 0, 0, 0, 0, 0, nil, 0, 1,
                    gxSelectToEnd, nil);
```

where `myFontFamily` is a reference to the font whose family you want information about.

Likewise, if you want a reference to the second font in a particular font family, call

```
GXFindFonts(myFontFamily, 0, 0, 0, 0, 0, nil, 0, 2, 1, &myFont);
```

Drawing With a Specific Font

To find a font by name, you use `GXFindFonts`. You call this function when you open a document that specifies a font. For example, if you want a reference to the font Chicago, use

```
GXFindFonts(nil, gxFullFontName, gxMacintoshPlatform,
            gxRomanScript, gxEnglishLanguage, strlen("Chicago"),
            "Chicago", 1, 1, &myFont);
```

Once you have found a specific font, you can use the following code to associate the font object with the shape object, so that a particular shape uses that font:

```
GXSetShapeFont(myShape, myFont);
```

Now you can draw the shape.

```
GXDrawShape(myShape);
```

Gaining Access to Font Properties

This section describes how you can gain access to and manipulate various font properties.

Getting a Font Name

If your application needs the name of a font (for example, to display it in a menu), you can use the `GXGetFontName` function.

If you want to get a font name by index (for example, if you are iterating through all font names in the font), you can also call the `GXGetFontName` function to get the size of a font name and then call it a second time to retrieve the name, as shown in Listing 7-2.

Note

If you are trying to get a font name by its meaning, platform, script, and language, use `GXFindFontName`. ♦

Listing 7-2 Using the `GXGetFontName` function

```
long length = GXGetFontName(myFont, index, nil, nil, nil, nil,
                            nil);
unsigned char* name = NewPtr(length);
GXGetFontName(myFont, index, nil, nil, nil, nil, name);
```

Font Objects

To return the font's full name as a C string, you can write the `FindFontFullName` library function, as shown in Listing 7-3.

Listing 7-3 Extracting a full name as a C string

```
/* return the font's full name as a C string */
short GXFindFontFullName(gxFont fontID, char name[])
{
    short length

    length = FindFontName(fontID, gxFullFontName,
                          gxMacintoshPlatform, gxRomanScript,
                          gxEnglishLanguage,
                          (unsigned char*)name, nil);

    name[length] = 0;
    return length;
}
```

Adding a Font Instance

If you add an attribute—for example, a font instance—to a font that has a font name, you must also add the name for that attribute. To add the attribute name, you must get an unused font name for the new name by using the `GXNewFontNameID` function.

Listing 7-4 shows one method of adding a new font instance and a font name for that instance to a font. The function takes a font, a font name, and coordinates for the new font instance. It gets a new `gxFontName` from the `GXNewFontNameID` function, adds the new font name to the font using the `GXSetFontName` function, and then adds the font instance to the font, using the `GXSetFontInstance` function to coordinate the new font instance with its associated font name.

Listing 7-4 Adding a new font name to a font

```
void AddFontInstance(gxFont theFont, const char instanceName[],
                    const gxFontVariation coord[])
{
    gxFontName nameID;

    nameID = GXNewFontNameID(theFont);
    GXSetFontName(theFont, nameID, gxMacintoshPlatform,
                  gxRomanScript, gxEnglishLanguage,
                  strlen(instanceName),
                  (unsigned char*)instanceName);
    GXSetFontInstance(theFont, 0, nameID, coord);
}
```

The `GXNewFontNameID` function is described on page 7-40. The `GXSetFontName` function is described on page 7-41. The `GXSetFontInstance` function is described on page 7-57.

Retrieving Font Features

This section describes the code you can use for retrieving font features. This technique is useful if your application needs to put in menus for features and instances. Listing 7-5 shows how you can retrieve an array of font features—in this case, ligature settings available in the font.

Listing 7-5 Retrieving an array of ligature settings

```
/*
Return the array of ligature settings available
in this font, or nil if they are not available.
*/
gxFontFeatureSetting* ReturnLigatureSettings(gxFont fontID)
{
    long index;
    gxFontName nameID;
    gxFontFeatureSetting* settings;
    settings = nil;
    nameID = GXFindFontFeature(fontID, ligaturesType, nil, &count,
                               nil, &index);

    if (nameID != 0)
    {
        settings = (gxFontFeatureSetting*)NewPtr(count *
            sizeof(gxFontFeatureSetting));
        GXGetFontFeature(fontID, index, nil, nil,
            settings, nil);
    }
    return settings;
}
```

The `GXFindFontFeature` function is described on page 7-62. The `GXGetFontFeature` function is described on page 7-61.

Font Objects

Determining Font Variations

Listing 7-6 shows you how to determine whether a font has a variation axis. This technique is useful if your application needs to put in sliders for font variations.

Listing 7-6 Determining font variations

```
/*
Determines whether font has a "weight" variation axis; if it does,
returns its maximum value.
*/
Fixed ReturnMaxWeightVariation(gxFont fontID)
{
    long index;
    Fixed max;
    index = GXFindFontVariation(fontID, 'wght', nil, nil, &max,
                                nil);

    if (index == 0)
        max = 0; // no 'wght' axis available
    return max;
}
```

The GXFindFontVariation function is described on page 7-55.

Retrieving Language-Specific Font Lists

Listing 7-7 shows you how to retrieve a language-specific font list. This technique is useful, for example, if you need to add just Japanese fonts in a Font menu. Listing 7-7 shows an example of how you can return all the fonts that support Japanese characters.

Listing 7-7 Retrieving all fonts that support Japanese characters

```
/* return all of the fonts that support Japanese characters*/
long FindJapaneseFonts(gxFont fontList[])
{
    long numJapaneseFonts = GXFindFonts(nil, gxNoFontName,
    gxMacintoshPlatform, gxJapaneseScript, gxNoLanguage,
    0, nil, 1, gxSelectToEnd, fontList);
    return numJapaneseFonts;
}
```

The GXFindFonts function is described on page 7-33.

Manipulating Font Tables

To get the number of font tables in a font, you can use the `GXCountFontTables` function, described on page 7-70. To get an entire font table that you specify by its index in the font's list of font tables, you use the `GXGetFontTable` function, described on page 7-71. To get part of a font table that you specify by index, you use the `GXGetFontTableParts` function, described on page 7-72.

To get an entire font table that you specify by its table tag, use `GXFindFontTable`. To get part of a font table that you specify by table tag from a font, you use `GXFindFontTableParts`.

To change part of an existing font table or add a new font table, use `GXSetFontTable`. To delete a font table from a font permanently, you use `GXDeleteFontTable`.

IMPORTANT

You should never alter the data in a font, including the font tables, unless your application is a font editor or otherwise needs to change the data in a font. Most applications, such as word processors, do not need to examine or change the data in a font. ▲

The tables of a TrueType GX font are described in *QuickDraw GX Font Formats*, available from APDA.

If you need to take advantage of some of QuickDraw GX's advanced font functions—for example, if your application is a font editor—you can use the `GXGetFontTable` function. This function allows you to retrieve the size of a particular font table and then call it a second time to retrieve the actual data from the table, as shown in Listing 7-8.

Listing 7-8 Using the `GXGetFontTable` function to retrieve a table

```
long size = GXGetFontTable(fontID, index, nil, nil);
void* tableData = NewPtr(size);
GXGetFontTable(fontID, index, tableData, nil);
```

The `GXGetFontTable` function is described on page 7-71.

Font Objects Reference

This section describes the enumerations, constants, data types, and functions that are specific to the QuickDraw GX font object.

“Basic Constants and Data Types” and “Advanced Constants and Data Types” list the enumerated types and structures that provide information about fonts.

“Basic Font Functions” beginning on page 7-32 lists the QuickDraw GX functions you use to manipulate QuickDraw GX fonts. “Advanced Font Functions” beginning on page 7-63 lists the QuickDraw GX functions you use to change the information a font contains.

Basic Constants and Data Types

This section describes the constants and data types that you can use to provide basic information about and retrieve basic information from QuickDraw GX font objects. The data types discussed in this section include

- the `gxFont` type, which you use to reference a font
- the `gxFontVariation` type, which you use to reference font variation and font instance information
- the `gxFontDescriptorTag` type, which you use to reference a font descriptor
- the `gxFontVariationTag` type, which you use to reference a variation axis
- the `gxFontName` type which distinguishes the types of font names available in a font
- the `gxFontFeatureFlag` enumeration, which you use to get information about a particular font feature in the font—for example, whether the settings of that feature are mutually exclusive
- the `gxFontFeature` type, which is a reference to a specific font feature
- the `gxFontFeatureSetting` type, which contains a setting of a font feature and the name ID in the name table of that setting
- the `gxFontPlatform` type, which identifies the platform for a name or encoding
- the `gxFontScript` type, which specifies the script used by a platform
- the `gxFontLanguage` type, which specifies the language used by a script

The Font Object

To gain access to font objects in QuickDraw GX, you use always functions. Your application never gets a pointer or handle to the actual font data. To allow type checking, QuickDraw GX defines the `gxFont` data type as a pointer to a structure.

```
typedef struct gxPrivateFontRecord *gxFont;
```

To obtain or alter information in a font object, you pass `gxFont` to a function. In general, you should never need to change or gain access to the data in a font object directly.

Font Variations, Instances, and Descriptors

The `gxFontVariation` structure describes font variations, instances, and descriptors. The `gxFontVariationTag` type identifies a variation axis.

```
struct gxFontVariation {
    gxFontVariationTag    name;
    Fixed                  value;
} ;
typedef struct gxFontVariation gxFontDescriptor;
```

Field descriptions

name	The name of the variation axis, in a four-character tag. For example, 'wght' is the name of the weight variation axis.
value	A coordinate along the variation axis specified by name.

You can get the minimum and maximum values for an axis using the `GXGetFontVariation` function, described on page 7-54, or the `GXFindFontVariation` function, described on page 7-55.

The `gxFontDescriptor` structure is identical to the `gxFontVariation` structure. The `gxFontDescriptorTag` identifies a font descriptor. Note that descriptors are used to identify the style of a simple font—that is, a font that does not have variations.

```
typedef long gxFontDescriptorTag
```

Font variations and instances, which use the variation axes, are discussed in “Font Variations” on page 7-10.

The possible values for the name field of the `gxFontVariation` structure, whether it describes a variation, instance, or descriptor, are described in “Font Variations” on page 7-10.

Font Names

Font names are values in a font that contain information about the font. Each font name is distinguished by a value from the `gxFontName` type.

```
enum gxFontNames {
    gxNoFontName,
    gxCopyrightFontName,
    gxFamilyFontName,
    gxStyleFontName,
    gxUniqueFontName,
    gxFullFontName,
    gxVersionFontName,
    gxPostscriptFontName,
    gxTrademarkFontName,
    gxManufacturerFontName
};

typedef long gxFontName;
```

Constant descriptions

<code>gxNoFontName</code>	The font name is not specified. You can use this value with such functions as <code>GXFindFonts</code> (page 7-33) to indicate that, during a search for a font using other specific criteria, any font name constitutes a match.
---------------------------	---

Font Objects

<code>gxCopyrightFontName</code>	The manufacturer's copyright.
<code>gxFamilyFontName</code>	The font family name, such as "Geneva".
<code>gxStyleFontName</code>	The font style, such as "Bold".
<code>gxUniqueFontName</code>	The manufacturer's unique name for the font, such as "Apple Computer Geneva Bold 3.0".
<code>gxFullFontName</code>	The full font name, such as "Geneva Bold".
<code>gxVersionFontName</code>	The manufacturer's version number, such as "3.0". (The name does not need to include the word "version.")
<code>gxPostscriptFontName</code>	The PostScript-legible name of the font, such as "Geneva-Bold".
<code>gxTrademarkFontName</code>	The trademark holder's name.
<code>gxManufacturerFontName</code>	The name of the font's manufacturer.

New font names are registered in *QuickDraw GX Font Feature Registry*, described on page 7-12. Font names are described in "Names" on page 7-6.

Font Features

The `gxFontFeature` type identifies information about a specific font feature. The possible values for a font feature are described in the chapter "Layout Shapes" in this book.

```
typedef long gxFontFeature;
```

The `gxFontFeatureFlag` type defines font feature flags.

```
typedef long gxFontFeatureFlag;
```

Font feature flags provide information about a particular font feature, such as whether the settings of the feature can be combined or are mutually exclusive.

```
#define gxMutuallyExclusiveFeature 0x8000
```

Flag descriptions

`gxMutuallyExclusiveFeature`

If this flag is set, the settings for this font feature in the font are mutually exclusive. If this flag is not set, the settings can be combined with each other. For example, a lettercase is exclusive, whereas ligatures are nonexclusive.

You can determine the font feature flags of a font feature using the `GXGetFontFeature` function, described on page 7-61 or the `GXFindFontFeature` function, described on page 7-62.

Font Objects

A font feature setting structure contains one setting of a font feature and its associated name ID. A font feature may have one or more settings associated with it.

The `GXGetFontFeature` and `GXFindFontFeature` functions use the `gxFontFeatureSetting` structure.

```
struct gxFontFeatureSetting {
    unsigned short setting;
    unsigned short nameID;
} ;
typedef long gxFontFeatureFlag;
```

Field descriptions

`setting` The font feature setting. The values for this field are described in the chapter “Layout Shapes” in this book.

`nameID` A reference to the font name for this setting in the font.

To find a name for the font feature setting, you can use the `GXGetFontName` function, described on page 7-38.

Font Platforms

A font platform marks the class of encoding table and character code set the font uses. A font may contain multiple encoding tables. Each platform is distinguished by a value from the `gxFontPlatforms` enumeration.

```
enum gxFontPlatforms {
    gxGlyphPlatform = -1,
    gxNoPlatform,
    gxUnicodePlatform,
    gxMacintoshPlatform,
    gxReservedPlatform,
    gxMicrosoftPlatform,
    gxCustomPlatform
} ;

typedef long gxFontPlatform;
```

Constant descriptions

`gxGlyphPlatform`

This is a reserved value used by the QuickDraw GX graphics system. A font never uses this setting. The graphics system uses this for identifiers that store glyph codes rather than character codes.

`gxNoPlatform`

The platform is not specified. You can use this value with such functions as `GXFindFonts`, described on page 7-33, to indicate that, during a search for a font using other specific criteria, any type of platform constitutes a match.

Font Objects

`gxUnicodePlatform`

The platform uses the Unicode character code specifications. For more information about the Unicode encodings, see *The Unicode Standard: Worldwide Character Encoding*, volumes 1 and 2, available from Addison-Wesley.

`gxMacintoshPlatform`

The platform uses one of the Macintosh character code sets.

`gxReservedPlatform`

The platform is reserved for future use.

`gxMicrosoftPlatform`

This platform uses one of the Microsoft character code sets.

`gxCustomPlatform`

This is a nonstandard platform, specific to the font.

QuickDraw GX Macintosh Scripts

Script values, together with platform values and optionally language values, identify the character encoding for a font. Each platform may define a set of script codes. For more information about scripts and how they work, see “Encodings” on page 7-7.

The Macintosh platform defines a number of script codes for scripts used around the world. The `gxMacintoshScripts` enumeration defines values for these script codes used with the `gxMacintoshScript` type.

```
enum gxMacintoshScripts {
    gxNoScript,
    gxRomanScript,
    gxJapaneseScript,
    gxTraditionalChineseScript,
    gxChineseScript = gxTraditionalChineseScript,
    gxKoreanScript,
    gxArabicScript,
    gxHebrewScript,
    gxGreekScript,
    gxCyrillicScript,
    gxRussianScript = gxCyrillicScript,
    gxRSymbolScript,
    gxDevanagariScript,
    gxGurmukhiScript,
    gxGujaratiScript,
    gxOriyaScript,
    gxBengaliScript,
    gxTamilScript,
    gxTeluguScript,
    gxKannadaScript,
    gxMalayalamScript,
    gxSinhaleseScript,
```

Font Objects

```

    gxBurmeseScript,
    gxKhmerScript,
    gxThaiScript,
    gxLaotianScript,
    gxGeorgianScript,
    gxArmenianScript,
    gxSimpleChineseScript,
    gxTibetanScript,
    gxMongolianScript,
    gxGeezScript,
    gxEthiopicScript = gxGeezScript,
    gxAmharicScript = gxGeezScript,
    gxSlavicScript,
    gxEastEuropeanRomanScript = gxSlavicScript,
    gxVietnameseScript,
    gxExtendedArabicScript,
    gxSindhiScript = gxExtendedArabicScript,
    gxUninterpretedScript
} ;

typedef long gxFontScript;

```

The `gxNoScript` value indicates that no particular script is specified. You can use this value with such functions as `GXFindFonts`, described on page 7-33, to indicate that, during a search for a font using other specific criteria, any type of script constitutes a match. All other values in the `gxMacintoshScripts` enumeration refer to the names of script systems from around the world.

The `gxFontScript` type identifies the script for a name or encoding. Each platform has a corresponding enumeration of legal scripts.

The `gxCustomScripts` enumeration defines values for script codes used with the `gxCustomPlatform` type. It is not related to other scripts.

```

enum gxCustomScripts {
    gxCustom8bitScript = 1,
    gxCustom816bitScript,
    gxCustom16bitScript
};

typedef long gxFontScript;

```

The `gxMicrosoftScripts` enumeration defines values for script codes used with the `gxMicrosoftPlatform` type.

```

enum gxMicrosoftScripts {
    gxMicrosoftSymbolScript = 1,
    gxMicrosoftStandardScript
};

```

Languages

The `gxFontLanguage` type names the language of a particular font name. It is used primarily for names but also appears as the identifying block for an encoding.

The `gxMacintoshLanguages` enumeration lists the language values that can be used with the `gxFontLanguage` type.

```
enum gxMacintoshLanguages {
    gxNoLanguage,
    gxEnglishLanguage,
    gxFrenchLanguage,
    gxGermanLanguage,
    gxItalianLanguage,
    gxDutchLanguage,
    gxSwedishLanguage,
    gxSpanishLanguage,
    gxDanishLanguage,
    gxPortugueseLanguage,
    gxNorwegianLanguage,
    gxHebrewLanguage,
    gxJapaneseLanguage,
    gxArabicLanguage,
    gxFinnishLanguage,
    gxGreekLanguage,
    gxIcelandicLanguage,
    gxMalteseLanguage,
    gxTurkishLanguage,
    gxCroatianLanguage,
    gxTradChineseLanguage,
    gxUrduLanguage,
    gxHindiLanguage,
    gxThaiLanguage,
    gxKoreanLanguage,
    gxLithuanianLanguage,
    gxPolishLanguage,
    gxHungarianLanguage,
    gxEstonianLanguage,
    gxLettishLanguage,
    gxLatvianLanguage = gxLettishLanguage,
    gxSaamiskLanguage,
    gxLappishLanguage = gxSaamiskLanguage,
    gxFaeroeseLanguage,
    gxFarsiLanguage,
    gxPersianLanguage = gxFarsiLanguage,
```


Font Objects

```

gxRussianLanguage,
gxSimpChineseLanguage,
gxFlemishLanguage,
gxIrishLanguage,
gxAlbanianLanguage,
gxRomanianLanguage,
gxCzechLanguage,
gxSlovakLanguage,
gxSlovenianLanguage,
gxYiddishLanguage,
gxSerbianLanguage,
gxMacedonianLanguage,
gxBulgarianLanguage,
gxUkrainianLanguage,
gxByelorussianLanguage,
gxUzbekLanguage,
gxKazakhLanguage,
gxAzerbaijaniLanguage,
gxAzerbaijanArLanguage,
gxArmenianLanguage,
gxGeorgianLanguage,
gxMoldavianLanguage,
gxKirghizLanguage,
gxTajikiLanguage,
gxTurkmenLanguage,
gxMongolianLanguage,
gxMongolianCyrLanguage,
gxPashtoLanguage,
gxKurdishLanguage,
gxKashmiriLanguage,
gxSindhiLanguage,
gxTibetanLanguage,
gxNepaliLanguage,
gxSanskritLanguage,
gxMarathiLanguage,
gxBengaliLanguage,
gxAssameseLanguage,
gxGujaratiLanguage,
gxPunjabiLanguage,
gxOriyaLanguage,
gxMalayalamLanguage,
gxKannadaLanguage,
gxTamilLanguage,

```

Font Objects

```

    gxTeluguLanguage,
    gxSinhaleseLanguage,
    gxBurmeseLanguage,
    gxKhmerLanguage,
    gxLaoLanguage,
    gxVietnameseLanguage,
    gxIndonesianLanguage,
    gxTagalogLanguage,
    gxMalayRomanLanguage,
    gxMalayArabicLanguage,
    gxAmharicLanguage,
    gxTigrinyaLanguage,
    gxGallaLanguage,
    gxOromoLanguage = gxGallaLanguage,
    gxSomaliLanguage,
    gxSwahiliLanguage,
    gxRuandaLanguage,
    gxRundiLanguage,
    gxChewaLanguage,
    gxMalagasyLanguage,
    gxEsperantoLanguage,
    gxWelshLanguage = 129,
    gxBasqueLanguage,
    gxCatalanLanguage,
    gxLatinLanguage,
    gxQuechuaLanguage,
    gxGuaraniLanguage,
    gxAymaraLanguage,
    gxTatarLanguage,
    gxUighurLanguage,
    gxDzongkhaLanguage,
    gxJavaneseRomLanguage,
    gxSundaneseRomLanguage
} ;
typedef long gxFontLanguage;

```

The `gxNoLanguage` value indicates that no particular language is specified. You can use this value with such functions as `GXFindFonts`, described on page 7-33, to indicate that, during a search for a font using other specific criteria, any type of language constitutes a match.

All other values in the `gxMacintoshLanguages` enumeration refer to the English names of languages from around the world.

Advanced Constants and Data Types

This section describes constants and data types that you can use for advanced operations on QuickDraw GX font objects.

- The `gxFontStorageTag` enumeration lists the ways you can store a QuickDraw GX font.
- The `gxFontStorageReference` type references the font from its particular method of storage.
- The `gxFontFormatTag` identifier specifies the particular format, and therefore the particular font scaler, of the QuickDraw GX font.
- The `gxFontTableTag` type references the names of font tables.
- The `gxFontAttribute` enumeration specifies whether a font is a system font or an application-specific font and whether it can be edited.

Font Storage Tags

Fonts can be stored as resources, handles, or files. You can determine how a font is stored by checking its associated storage type, which is of the `gxFontStorageTag` type.

```
#define gxResourceFontStorage 0x72737263 /* 'rsrc' */
#define gxHandleFontStorage 0x686e646c /* 'hdl' */
#define gxFileFontStorage 0x62617373 /* 'bass' */
#define gxNfntFontStorage 0x6e666e74 /* 'nfnt' */
```

```
typedef long gxFontStorageTag;
```

Constant descriptions

`gxResourceFontStorage`

The font is stored in an 'sfnt' resource. The resource does not need to be loaded.

`gxHandleFontStorage`

The font is stored in a nonpurgeable handle.

`gxFileFontStorage`

The font is stored in an open file.

`gxNfntFontStorage`

The font is stored in a 'FOND' resource that references only 'FONT' and 'NFNT' resources.

The `gxFontStorageReference` type contains the reference to the resource, handle, or file where the resource is stored. If the font is stored in a resource, the reference is a handle to the resource; if the font is stored in a handle, the reference is the handle itself; and if the font is stored in a file, the reference is the file reference number. If the font is stored in an 'NFNT' resource, the reference is `styleBits * 65536 + FONDRessourceID`.

```
typedef void *gxFontStorageReference;
```

Font Table Tags

A font table tag is a 4-byte code that describes the type of table. For example, the table tag 'kern' means the table is a kerning table. For portability, when your application calls routines such as `gxFindFontTable`, you should specify tags in the hexadecimal notation `0x6B65726E`, because of byte-ordering concerns.

```
typedef long gxFontTableTag;
```

Font Attributes

Each font has a set of **font attributes**, which are a group of flags that modify the behavior or identity of the font. These flags are defined in the `gxFontAttributes` enumeration.

```
enum gxFontAttributes {
    gxSystemFontAttribute = 0x0001
    gxReadOnlyFontAttribute = 0x0002
};
```

```
typedef long gxFontAttribute;
```

Constant descriptions

`gxSystemFontAttribute`

The font object was not created by the application but was created by QuickDraw GX.

`gxReadOnlyFontAttribute`

The font object cannot be passed as a parameter to any of the font-editing functions, for example, `GXDeleteFontName`.

Fonts created by calling `GXNewFont` are marked as nonsystem fonts. Fonts stored in 'NFNT' resources are always marked read-only and cannot be edited in QuickDraw GX.

To determine the font attributes of a specified font, you can use the `GXGetFont` (page 7-67) or `GXFindFont` (page 7-67) function.

Basic Font Functions

This section describes the functions that access, retrieve, change, or delete information about the fonts in the system. With these functions, you can

- get a complete list of available fonts, or any subset of the list, such as a list only of font families
- determine what the default font for your application is and change it
- retrieve, add, or delete font names
- retrieve information about font encoding, features, and variations, without editing the font
- get and alter information about font descriptors and instances

Font Objects

These functions describe ways of altering the tables of the font. The advanced functions are useful mainly for font-related applications, such as font editors. In most cases, general-purpose applications, such as word processors, do not need the advanced functions.

Note

QuickDraw GX functions that begin with *GXGet* and *GXFind* are similar in that they retrieve roughly the same information. However, the *GXGet* functions find the information by its index in the font's list of that type of information and return a tag or a font name, whereas the *GXFind* functions use a tag to find the information and return its index. ♦

Getting the List of Available Fonts

You can get the list of available fonts, or specified subsets of fonts or font families, using *GXFindFonts*.

GXFindFonts

You can use the *GXFindFonts* function to get the list of available fonts or a list of fonts that match a particular set of criteria.

```
long GXFindFonts(gxFont family, gxFontName meaning,
                 gxFontPlatform platform, gxFontScript script,
                 gxFontLanguage language, long nameLength,
                 const unsigned char name[], long index,
                 long count, gxFont fonts[]);
```

family	A reference to the font whose font family determines the search range <i>GXFindFonts</i> uses. If this value is nil, then <i>GXFindFonts</i> applies the search criteria to all fonts, without regard to their families. If it is not nil, the search is limited to members of this family.
meaning	The kind of font name you want <i>GXFindFonts</i> to search for.
platform	The platform of the font. The platform, script, and language parameters identify the character set and language of the name.
script	The script of the font.
language	The language of the font.
nameLength	A byte count for the data stored in the name parameter.
name	The actual name of the font you are searching for. If you are searching for a particular font, you can include its name, such as "Geneva," here.
index	The number of the matching font to begin counting. A value of 1 means <i>GXFindFonts</i> should start with the first matching font it finds.

Font Objects

<code>count</code>	The maximum number of matches you want <code>GXFindFonts</code> to return. The function returns the actual number of matching fonts, which may be less than or equal to the value of <code>count</code> . If you want all possible matches, use the value <code>gxSelectToEnd</code> .
<code>fonts</code>	A pointer to a buffer your application provides, into which <code>GXFindFonts</code> copies the font references that match the given criteria. The function returns, in the <code>count</code> parameter, the number of fonts that matched. If the value of <code>fonts</code> is <code>nil</code> , <code>GXFindFonts</code> returns nothing in this parameter.

function result The number of fonts that match the search criteria.

DESCRIPTION

The `GXFindFonts` function takes search criteria (specified by the `family`, `meaning`, `name`, `platform`, `script`, `language`, and `index` parameters) and returns both the number of fonts that match those criteria and, if there are any matching fonts and the value of `fonts` is not `nil`, the font references of the fonts that match.

The values of the `nameLength` and `name` parameters specify the actual data for the font being searched for. Note that `nameLength` contains a byte count, which may not equal the number of characters in the name.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>count_is_less_than_one</code>	(debugging version)
<code>index_is_less_than_one</code>	(debugging version)
<code>inconsistent_parameters</code>	(debugging version)
<code>parameter_out_of_range</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	
<code>illegal_font_parameter</code>	

SEE ALSO

Various ways of using the `GXFindFonts` function are described in “Getting Information About Available Fonts” on page 7-15.

Font names are discussed in “Names” on page 7-6.

Platforms, scripts, languages, and encoding tables are discussed in “Encodings” on page 7-7.

Counting Glyphs in a Font

If your application needs to know how many glyphs are in a font (for example, to make waterfalls), it can count them with the `GXCountFontGlyphs` call.

GXCountFontGlyphs

You can use the `GXCountFontGlyphs` function to count the number of glyphs in a font.

```
long GXCountFontGlyphs (gxFont fontID);
```

`fontID` A reference to the font whose glyphs you want to count.

function result The number of glyphs in the font.

DESCRIPTION

The `GXCountFontGlyphs` function retrieves the number of glyphs present in the font referenced in the `fontID` parameter. Valid glyph codes for the font range from 0 to this function's result minus 1. Glyph codes can be set directly into a typographic shape and then drawn.

ERRORS, WARNING, AND NOTICES

Errors

```
out_of_memory
internal_font_error
illegal_font_parameter
```

Getting and Setting the Default Font

When you first create a typographic shape, its style is initialized to have a font reference of `nil`. When QuickDraw GX is passed `nil` for the font parameter, it substitutes the default font for that parameter. To manipulate the default font, you can use `GXGetDefaultFont` and `GXSetDefaultFont`. QuickDraw GX initializes this to Helvetica, if it is available.

GXGetDefaultFont

You can use the `GXGetDefaultFont` function to retrieve the current default font for your application.

```
gxFont GXGetDefaultFont(void);
```

function result A reference to the current application's default font.

Font Objects

DESCRIPTION

The `GXGetDefaultFont` function returns a reference to the application's default font. QuickDraw GX initializes it to a font, so the application does not need to set it.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`

SEE ALSO

You can change the default font with `GXSetDefaultFont`, described next.

GXSetDefaultFont

You can use the `GXSetDefaultFont` function to set the default font for your application.

```
gxFont GXSetDefaultFont(gxFont fontID);
```

`fontID` A reference to the font you want as the default font of your application.

function result A reference to the previous default font.

DESCRIPTION

The `GXSetDefaultFont` function changes your application's default font to the font specified by `fontID`. Because there is no system-wide default font, `GXSetDefaultFont` affects only the default font for the current application.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

SEE ALSO

To determine the current default font for your application, use `GXGetDefaultFont` (page 7-35).

Manipulating Font Names

QuickDraw GX allows you to work with font names in several ways. You can

- count the number of font names in a particular font (`GXCountFontNames`)
- specify a font name by index (`GXGetFontName`)
- specify a font name by type, platform, script, and language (`GXFindFontName`)
- find an available ID for a new font name (`GXNewFontID`)
- edit an existing font name, or create a new one in the font (`GXSetFontName`)
- delete a font name permanently from a font (`GXDeleteFontName`)

GXCountFontNames

You can use the `GXCountFontNames` function to determine the number of font names in a font.

```
long GXCountFontNames(gxFont fontID);
```

`fontID` A reference to the font whose font names you want to count.

function result The total number of entries in the names property of the font object. These entries include strings such as the names of features, which just identify specific aspects of the font.

DESCRIPTION

The `GXCountFontNames` function returns the total number of names in the font referenced by `fontID`. This total includes each occurrence of each name, including repetitions of the same name in different platforms, languages, or scripts, and other strings such as the names of features. You can use this number to iterate through the names in a font using the `GXGetFontName` function (described next).

ERRORS, WARNINGS, AND NOTICES

Errors

```
out_of_memory
internal_font_error
illegal_font_parameter
```

SEE ALSO

Font names are discussed in “Names” on page 7-6.

GXGetFontName

You can use the `GXGetFontName` function to find a font name by its index.

```
long GXGetFontName(gxFont fontID, long index, gxFontName *name,
                  gxFontPlatform *platform, gxFontScript *script,
                  gxFontLanguage *language, unsigned char text[]);
```

<code>fontID</code>	A reference to the font from which you want to extract font names.
<code>index</code>	A value between 1 and the number of font names. (The number of font names is returned by the <code>GXCountFontNames</code> function.)
<code>name</code>	A pointer to the type of font name.
<code>platform</code>	A pointer to the platform of the font name.
<code>script</code>	A pointer to the script of the font name.
<code>language</code>	A pointer to the language of the font name.
<code>text</code>	On return, the text of the font name. If <code>text</code> is set to <code>nil</code> , the function ignores it.

function result The byte length of the font name found. If no font name is found, the function returns 0.

DESCRIPTION

The `GXGetFontName` function takes a font reference and an index in the list of font names in the font and returns the font name and information about the name—its type, platform, script, and language—if those parameters are not `nil`.

If the function returns 0, `GXGetFontName` did not find a font name or modify any of the parameters.

Your application is responsible for allocating the memory for the `text` parameter.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>index_out_of_range</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	
<code>illegal_font_parameter</code>	

SEE ALSO

The `GXCountFontNames` function is described on page 7-37.

To search for a font name by its type, platform, script, or language, use the `GXFindFontName` function (described on page 7-39).

Font names are discussed in “Names” on page 7-6.

Platforms, scripts, encoding tables, and languages are discussed in “Encodings” on page 7-7.

GXFindFontName

You can use the `GXFindFontName` function to find a font name by its type, platform, script, and language.

```
long GXFindFontName(gxFont fontID, gxFontName name,
                    gxFontPlatform platform, gxFontScript script,
                    gxFontLanguage language, unsigned char text[],
                    long *index);
```

<code>fontID</code>	A reference to the font whose font names you want to search.
<code>name</code>	The type of font name you are searching for.
<code>platform</code>	The platform of the font name you are searching for.
<code>script</code>	The script of the font name you are searching for.
<code>language</code>	The language of the font name you are searching for.
<code>text</code>	On return, the text of the font name. If you pass <code>nil</code> for this parameter, the function ignores it.
<code>index</code>	On return, a pointer to the index of the font name in the font’s list of font names. If you pass <code>nil</code> for this parameter, the function ignores it.

function result The byte length of the font name. If no font name is found, the function returns 0.

DESCRIPTION

The `GXFindFontName` function searches the specified font for a font name that matches the values of the `name`, `platform`, `script`, and `language` parameters. If it finds a font name, `GXFindFontName` fills out the `text` and `index` parameters with the font name and its index in the font’s list of font names, if those parameters are not set to `nil`.

Your application is responsible for allocating the storage for the `text` parameter.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>inconsistent_parameters</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	
<code>illegal_font_parameter</code>	

Warnings

<code>index_out_of_range</code>

Font Objects

SEE ALSO

To search for a font name by its index, use the `GXGetFontName` function, described on page 7-38.

Font names are discussed in “Names” on page 7-6.

Platforms, scripts, languages, and encoding tables are discussed in “Encodings” on page 7-7.

GXNewFontNameID

You can use the `GXNewFontNameID` function to get a font name ID that is currently unused in a specific font. However, calling this function does not actually change the font.

```
gxFontName GXNewFontNameID(gxFont fontID);
```

`fontID` A reference to the specific font for which you want to find a font name ID.

function result A font name ID that is currently available.

DESCRIPTION

The `GXNewFontNameID` function finds an available font name ID for the font you specify. The function doesn’t reserve the font name ID for you; if you call the function twice in succession, it may return the same font name ID both times. You can use this font name ID with the `GXSetFontName` function (described next) to add a new font name that does not have a conflicting font name ID.

One use for this function is adding a new font instance, which requires specifying a variation coordinate and a name ID.

ERRORS, WARNINGS, AND NOTICES**Errors**

```
out_of_memory
internal_font_error
illegal_font_parameter
```

GXSetFontName

You can use the `GXSetFontName` function to add a new font name to or change an existing font name in a font.

```
long GXSetFontName(gxFont fontID, gxFontName name,
                  gxFontPlatform platform, gxFontScript script,
                  gxFontLanguage language, long nameLength,
                  const unsigned char text[]);
```

<code>fontID</code>	A reference to the font for which you want to add or substitute a font name.
<code>name</code>	The type of font name.
<code>platform</code>	The platform of the font name. The <code>platform</code> , <code>script</code> , and <code>language</code> parameters identify the character set and language of the name. You must include values for these parameters, because a font can store multiple versions of the same name, each one in a different platform, script, and language.
<code>script</code>	The script of the font name.
<code>language</code>	The language of the font name.
<code>nameLength</code>	The byte count of the text in the name parameter, which may be different from the character count, depending on the type of text.
<code>text</code>	A pointer to the name you want to add to the font or substitute for another font name. The <code>text</code> parameter can point to a list of 1-byte character codes or 2-byte character codes for Kanji or Unicode.

function result The index of the name added or changed in the font's list of font names.

DESCRIPTION

The `GXSetFontName` function adds a new font name or replaces an existing font name with the text in the name parameter. The function replaces the font name only if name, platform, script, and language match an existing name in the font; otherwise, `GXSetFontName` adds a new entry to the font.

If you add font names to a font, QuickDraw GX enlarges the font data accordingly.

IMPORTANT

The `GXSetFontName` function permanently changes the data in the font. In general, you should only call this function for fonts created by your application using `GXNewFont` and not for a system font. ▲

Font Objects

ERRORS, WARNINGS, AND NOTICES

Errors

<code>parameter_out_of_range</code>	(debugging version)
<code>inconsistent_parameters</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	
<code>illegal_font_parameter</code>	
<code>font_cannot_be_changed</code>	

SEE ALSO

The `GXNewFont` function is described on page 7-64.

To retrieve a font name by its index, use the `GXGetFontName` function, described on page 7-38. To retrieve a font name by its platform, script, or language, use the `GXFindFontName` function, described on page 7-39.

Font names are discussed in “Names” on page 7-6.

Platforms, scripts, languages, and encoding tables are discussed in “Encodings” on page 7-7.

GXDeleteFontName

You can use the `GXDeleteFontName` function to delete a font name from a font.

```
long GXDeleteFontName(gxFont fontID, long index,
                     gxFontName name, gxFontPlatform platform,
                     gxFontScript script,
                     gxFontLanguage language);
```

<code>fontID</code>	A reference to the font from which you want to delete a font name.
<code>index</code>	The index of the font name in the font’s list of font names. If this value is greater than 0, it represents the indexed location of the font name in the font; if this value is 0, <code>GXDeleteFontName</code> identifies the font name by the values of the meaning, platform, script, and language parameters.
<code>name</code>	The type of font name you want to delete.
<code>platform</code>	The platform of the font name. The platform, script, and language parameters uniquely identify the character set and language of the font name. You must include values for these parameters, because a font can store multiple versions of the same name, each one in a different platform, script, and language.
<code>script</code>	The script of the font name.
<code>language</code>	The language of the font name.

function result The index of the name deleted. If the function does not delete a name, it returns 0.

DESCRIPTION

The `GXDeleteFontName` function permanently deletes a font name from the font referenced by the `fontID` parameter. You can identify the font name you want deleted either by its index, or if `index` is equal to 0, by its meaning, platform, script, and language parameters.

If you delete font names from a font, QuickDraw GX decreases the font data accordingly.

IMPORTANT

The `GXDeleteFontName` function permanently changes the data in the font. In general, you should only call this function for fonts created by your application using `GXNewFont` and never for a system font. ▲

ERRORS, WARNINGS, AND NOTICES**Errors**

<code>out_of_memory</code>	
<code>internal_font_error</code>	
<code>illegal_font_parameter</code>	
<code>font_cannot_be_changed</code>	
<code>inconsistent_parameters</code>	(debugging version)

Warnings

<code>font_table_not_found</code>

SEE ALSO

The `GXNewFont` function is described on page 7-64.

Font names are discussed in “Names” on page 7-6.

Platforms, scripts, languages, and encoding tables are discussed in “Encodings” on page 7-7.

Manipulating Font Encodings

QuickDraw GX allows you to work with font encodings in several ways. You can

- count the number of encoding tables in a font (`GXCountFontEncodings`)
- specify by index the platform, script, and language values for an encoding table (`GXGetFontEncoding`)
- obtain the index for an encoding table that you specify by its platform, script, and language values (`GXFindFontEncoding`)
- obtain the glyph codes of a specific platform encoding that correspond to a string of character codes (`GXApplyFontEncoding`)

GXCountFontEncodings

You can use the `GXCountFontEncodings` function to retrieve the number of encoding tables in a font.

```
long GXCountFontEncodings(gxFont fontID);
```

`fontID` A reference to the font whose encoding tables you want to count.

function result The number of supported encoding tables in the font.

DESCRIPTION

The `GXCountFontEncodings` function returns the number of supported encoding tables in the font.

ERRORS, WARNINGS, AND NOTICES

Errors

```
out_of_memory
internal_font_error
illegal_font_parameter
```

GXGetFontEncoding

You can use the `GXGetFontEncoding` function to identify an encoding table in a font by its index.

```
gxFontPlatform GXGetFontEncoding(gxFont fontID, long index,
                                   gxFontScript *script,
                                   gxFontLanguage* language);
```

`fontID` A reference to the font you want to search for a specific encoding table.

`index` A value between 1 and the number of supported encoding tables. (The number of encoding tables is returned by `CountFontEncodings`.)

`script` On return, the script of the specified encoding table.

`language` On return, the language of the specified encoding table.

function result The table's platform value.

DESCRIPTION

The `GXGetFontEncoding` function identifies an encoding table in a font by its index. If the `script` parameter is not set to `nil`, `GXGetFontEncoding` returns the script value for that encoding table. If the `language` parameter is not set to `nil`, `GXGetFontEncoding` returns the language value for that encoding table.

ERRORS, WARNINGS, AND NOTICES**Errors**

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

Warnings

`index_out_of_range`

SEE ALSO

The `GXCountFontEncodings` function is described on page 7-44.

To search for an encoding table specified by platform, script, and language, use the `GXFindFontEncoding` function, described next.

Platforms, scripts and encoding tables are discussed in “Encodings” on page 7-7.

GXFindFontEncoding

You can use the `GXFindFontEncoding` function to find the index of an encoding table specified by platform, script, and language values.

```
long GXFindFontEncoding(gxFont fontID, gxFontPlatform platform,
                        gxFontScript script,
                        gxFontLanguage language);
```

<code>fontID</code>	A reference to the font you want to search for a specific encoding table.
<code>platform</code>	The platform of the encoding table.
<code>script</code>	The script of the encoding table.
<code>language</code>	The language of the encoding table.

function result The table’s index in the font’s list of encoding tables. If the function does not find the specified encoding table, it returns 0.

Font Objects

DESCRIPTION

The `GXFindFontEncoding` function searches the specified font for an encoding table that supports the specified platform, script, and language. If you specify a language, you must also specify a script, although you can specify a script without a language.

If you specify a language without naming a script or specify a script without naming a platform, `GXFindFontEncoding` returns the error `inconsistent_parameters`.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>inconsistent_parameters</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	
<code>illegal_font_parameter</code>	

Warnings

<code>font_language_not_found</code>

SEE ALSO

To search for an encoding table specified by index, use the `GXGetFontEncoding` function, described on page 7-44.

Platforms are discussed in “Encodings” on page 7-7. The `gxFontPlatform` data type is discussed in “Font Platforms” on page 7-25.

Scripts and encoding tables are discussed in “Encodings” on page 7-7. The `gxFontScript` data type is discussed in “QuickDraw GX Macintosh Scripts” on page 7-26.

GXApplyFontEncoding

You can use the `GXApplyFontEncoding` function to translate a list of character codes to the glyph indices that correspond to a particular encoding table.

```
long GXApplyFontEncoding(gxFont fontID, long index, long* length,
                        const unsigned char text[], long count,
                        unsigned short glyphs[], char was16Bit[]);
```

<code>fontID</code>	A reference to the font containing the encoding table you want to search.
<code>index</code>	The index of the encoding table to be used.
<code>length</code>	On entry, the maximum number of bytes of text you want the function to process. On return, this parameter specifies the number of bytes in the text parameter actually processed. This is useful for scripts such as Kanji, which contain mixed 8-bit and 16-bit text. If you pass <code>nil</code> for this parameter, the function ignores it.

Font Objects

<code>text</code>	An array of character codes that <code>GXApplyFontEncoding</code> should translate into glyph indices from the specified encoding table.
<code>count</code>	The maximum number of glyphs in the <code>glyphs</code> parameter.
<code>glyphs</code>	On return, an array of the resulting glyph codes. Your application must maintain a buffer to hold this array.
<code>was16bit</code>	An array of Boolean values, one for each glyph. Each value indicates whether the glyph is a translation of an 8-bit or 16-bit character code. The value <code>true</code> means the corresponding character code is 16-bit. If this parameter is <code>nil</code> , the function ignores it.

function result The number of glyphs processed by the function.

DESCRIPTION

The `GXApplyFontEncoding` function takes a font reference and an array of character codes from the `text` parameter and puts the array of corresponding glyph codes from the specified encoding table in the `glyphs` parameter. Typographic shapes perform this function automatically; thus, your application may never need to call this function directly.

The `GXApplyFontEncoding` function returns glyph codes that are zero-based. Glyph code 0 is the missing glyph.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>unknown_font_table_format</code>	
<code>out_of_memory</code>	
<code>internal_font_error</code>	
<code>illegal_font_parameter</code>	
<code>length_is_less_than_zero</code>	(debugging version)

Warnings

<code>index_out_of_range</code>

SEE ALSO

To find an encoding table by its index, use the `GXGetFontEncoding` function, described on page 7-44.

To find an encoding table by its platform, script, and language values, use the `GXFindFontEncoding` function, described on page 7-45.

Manipulating Font Descriptors

QuickDraw GX allows you to work with font descriptors in several ways. You can

- count the number of font descriptors in a particular font (GXCountFontDescriptors)
- get a font descriptor that you specify by its index (GXGetFontDescriptor)
- get a font descriptor that you specify by its tag, such as 'width' (GXFindFontDescriptor)
- add new information about a font descriptor, or create a new font descriptor (GXSetFontDescriptor)
- delete a font descriptor permanently from a font (GXDeleteFontDescriptor)

Font descriptors are described in “Font Descriptors” on page 7-9.

GXCountFontDescriptors

You can use the GXCountFontDescriptors function to get the number of descriptors in a font.

```
long GXCountFontDescriptors(gxFont fontID);
```

`fontID` A reference to the font whose descriptors you want to count.

function result The number of descriptors in the font.

DESCRIPTION

The GXCountFontDescriptors function returns the number of different descriptors available in the font. Each descriptor consists of a descriptor tag and a value.

ERRORS, WARNINGS, AND NOTICES

Errors

```
out_of_memory
internal_font_error
illegal_font_parameter
```

GXGetFontDescriptor

You can use the `GXGetFontDescriptor` function to return the font descriptor you specify by index.

```
gxFontDescriptorTag GXGetFontDescriptor(gxFont fontID, long index,
                                         Fixed* descriptorValue);
```

fontID A reference to the font from which you want to get the font descriptor information.

index A value between 1 and the number of descriptors in a font. (The number of descriptors is returned by `GXCountFontDescriptors`.)

descriptorValue
On return, the value assigned to this descriptor.

function result The descriptor tag of the specified descriptor.

DESCRIPTION

The `GXGetFontDescriptor` function returns the tag for the font descriptor you specify by index. The function then copies the descriptor's value into the `descriptorValue` parameter, if `descriptorValue` is not set to `nil`.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

Warnings

`index_out_of_range`

SEE ALSO

The `GXCountFontDescriptors` function is described on page 7-48.

To retrieve a font descriptor specified by descriptor tag, use the `GXFindFontDescriptor` function, described next.

GXFindFontDescriptor

You can use the `GXFindFontDescriptor` function to return the index and value for a font descriptor you specify by tag.

```
long GXFindFontDescriptor(gxFont fontID,
                          gxFontDescriptorTag descriptorTag,
                          Fixed* descriptorValue);
```

`fontID` A reference to the font from which you want to get font descriptor information.

`descriptorTag` The descriptor tag you are searching for.

`descriptorValue` On return, the value assigned to this descriptor.

function result The index of the specified descriptor. If the function does not find the descriptor, it returns 0.

DESCRIPTION

The `GXFindFontDescriptor` function returns the index for the font descriptor you specify by tag. If the function returns a positive index, it also returns the descriptor's value in the `descriptorValue` parameter, if `descriptorValue` is not set to `nil`.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

SEE ALSO

You can retrieve a font descriptor specified by index using the `GXGetFontDescriptor` function, described on page 7-49.

GXSetFontDescriptor

You can use the `GXSetFontDescriptor` function to add a new descriptor to or change an existing descriptor in a font.

```
long GXSetFontDescriptor(gxFont fontID, long index,
                        gxFontDescriptorTag descriptorTag,
                        Fixed descriptorValue);
```

fontID A reference to the font you want to add a font descriptor to or change a font descriptor in.

index The index of the descriptor in the font's list of descriptors, if this value is greater than 0. If it is 0, then `GXSetFontDescriptor` uses the `descriptorTag` parameter.

descriptorTag The tag of the descriptor you want to add or change. If the value of `index` is greater than 0, this parameter should be set to 0.

descriptorValue The value you want assigned to this descriptor.

function result The index of the descriptor that was changed or added.

DESCRIPTION

The `GXSetFontDescriptor` function changes the value of the descriptor in the font you specify by tag. If no matching descriptor is found, `GXSetFontDescriptor` adds a new descriptor to the font with the given descriptor tag and value.

If you add font descriptors to a font, QuickDraw GX is responsible for enlarging the font data accordingly.

IMPORTANT

The `GXSetFontDescriptor` function permanently changes the data in the font. In general, you should only call this function for fonts created by your application using `GXNewFont` and never for a system font. ▲

ERRORS, WARNINGS, AND NOTICES

Errors

<code>illegal_font_parameter</code>	
<code>font_cannot_be_changed</code>	
<code>inconsistent_parameters</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	

Warnings

`index_out_of_range`

GXDeleteFontDescriptor

You can use the `GXDeleteFontDescriptor` function to delete a font descriptor you specify by index or tag.

```
long GXDeleteFontDescriptor(gxFont fontID, long index,
                           gxFontDescriptorTag descriptorTag);
```

<code>fontID</code>	A reference to the font from which you want to delete a font descriptor.
<code>index</code>	The index of the descriptor you want to delete. If this value is greater than 0 and the <code>descriptorTag</code> parameter is set to 0, the value of <code>index</code> is the index of the descriptor in the font's descriptor list. If this parameter is set to 0 and the <code>descriptorTag</code> parameter contains a value other than <code>nil</code> , <code>GXDeleteFontDescriptor</code> uses the value of the <code>descriptorTag</code> parameter to search for the descriptor.
<code>descriptorTag</code>	The tag you want to delete.
<i>function result</i>	The index of the deleted descriptor. If the function does not delete a descriptor, it returns 0.

DESCRIPTION

The `GXDeleteFontDescriptor` function permanently deletes the specified descriptor from the font. You can specify the descriptor either by the value of its index in the font's list of descriptors, or by its tag in the `descriptorTag` parameter, if the `index` parameter is set to 0.

If you delete font descriptors from a font, QuickDraw GX is responsible for decreasing the font data accordingly.

IMPORTANT

The `GXDeleteFontDescriptor` function permanently changes the data in the font. In general, you should only call this function for fonts created by your application using `GXNewFont` and never for a system font. ▲

ERRORS, WARNINGS, AND NOTICES

Errors

<code>illegal_font_parameter</code>	
<code>font_cannot_be_changed</code>	
<code>inconsistent_parameters</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	

Warnings

<code>font_table_not_found</code>
<code>index_out_of_range</code>

Manipulating Font Variations

QuickDraw GX allows you to work with font variations in several ways. You can

- count the number of font variation axes in a particular font (`GXCountFontVariations`)
- get a font variation that you specify by its index in the font's list of font variations (`GXGetFontVariation`)
- get a font variation that you specify by its tag (such as 'width') in the font's list of font variations (`GXFindFontVariation`)

You cannot add variations to or delete variations from a font because the format of a font variation varies from one font format to another.

Font variations are described in “Font Variations” on page 7-10.

GXCountFontVariations

You can use the `GXCountFontVariations` function to get the number of variation axes available for a font.

```
long GXCountFontVariations(gxFont fontID);
```

`fontID` A reference to the font containing the variations.

function result The number of variations in the font.

DESCRIPTION

The `GXCountFontVariations` function returns the number of different variation axes available from the font.

ERRORS, WARNINGS, AND NOTICES

Errors

```
out_of_memory
internal_font_error
illegal_font_parameter
```

GXGetFontVariation

You can use the `GXGetFontVariation` function to get the tag for a specific variation axis in a font you specify by index.

```
gxFontVariationTag GXGetFontVariation(gxFont theFont, long index,
                                       Fixed* minValue,
                                       Fixed* defaultValue,
                                       Fixed* maxValue,
                                       gxFontName* nameID);
```

<code>theFont</code>	A reference to the font containing the font variation you want to find.
<code>index</code>	The index of the variation you want in the list of variations in the font.
<code>minValue</code>	On return, the minimum value for the variation.
<code>defaultValue</code>	On return, the default value for the variation. This value is exactly the same as the font's descriptor value for this tag.
<code>maxValue</code>	On return, the maximum value for the variation.
<code>nameID</code>	On return, the name ID for this variation in the font.

function result The tag for the variation specified.

DESCRIPTION

The `GXGetFontVariation` function returns the tag for the variation specified by the `index` parameter. The function also returns the minimum value, default value, maximum value, and name ID of the variation, if those parameters are not set to `nil`. The name ID identifies the name (for instance, "weight") of the specified variation axis.

You can use the font name ID returned by `GXGetFontVariation` with the `GXFindFontName` function (page 7-39) to retrieve the name for a given variation from the font names property of the font object.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

Warnings

`index_out_of_range`

SEE ALSO

To search for a specific variation by its tag, use the `GXFindFontVariation` function, described next.

GXFindFontVariation

You can use the `GXFindFontVariation` function to find a variation in a font you specify by its tag.

```
long GXFindFontVariation(gxFont theFont,
                        gxFontTableTag variationTag,
                        Fixed* minValue, Fixed* defaultValue,
                        Fixed* maxValue, gxFontName* nameID);
```

`theFont` A reference to the font containing the font variation you want to find.

`variationTag` The tag of the variation you are searching for.

`minValue` On return, the minimum value for the variation.

`defaultValue` On return, the default value for the variation.

`maxValue` On return, the maximum value for the variation.

`nameID` On return, the ID of the name in the font for this variation.

function result The index of the variation specified.

DESCRIPTION

The `GXFindFontVariation` function returns the index for the variation specified by `variationTag`. The function also returns the minimum value, default value, maximum value, and name ID of the variation, if those parameters are not set to `nil`.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

SEE ALSO

To retrieve a specific variation by its index, use the `GXGetFontVariation` function (page 7-54).

You can use the font name ID returned by `GXFindFontVariation` with the `GXFindFontName` function (page 7-39) to retrieve the name for a given variation from the names property of the font object.

Manipulating Font Instances

QuickDraw GX allows you to work with font instances in several ways. You can

- count the number of font instances in a particular font (`GXCountFontInstances`)
- get a specific font instance in the font's list of font instances (`GXGetFontInstance`)
- set information about a font instance, or create a new one in the font (`GXSetFontInstance`)
- delete a font instance permanently from a font (`GXDeleteFontInstance`)

Font instances are described in “Font Instances” on page 7-11.

GXCountFontInstances

You can use the `GXCountFontInstances` function to retrieve the number of font instances available in a font.

```
long GXCountFontInstances(gxFont theFont);
```

`theFont` A reference to the font whose font instances you want to count.

function result The number of font instances in the font.

DESCRIPTION

The `GXCountFontInstances` function returns the number of font instances available in the font.

ERRORS, WARNINGS, AND NOTICES

Errors

```
out_of_memory
internal_font_error
illegal_font_parameter
```

GXGetFontInstance

You can use the `GXGetFontInstance` function to get the font name for a font instance you specify by index.

```
gxFontName GXGetFontInstance(gxFont theFont, long index,
                              gxFontVariation variation[]);
```

`theFont` A reference to the font whose font instance you want to retrieve.

Font Objects

`index` The index number of the font instance you want.
`variation` On return, an array of the variation data for this instance.

function result The font name for the font instance specified.

DESCRIPTION

The `GXGetFontInstance` function returns the font name for the font instance specified by the value of the `index` parameter. Then `GXGetFontInstance` copies the font variation settings for that instance into the `variation` parameter, if `variation` is not set to `nil`.

Each instance is always identified by a complete set of font variations. A font instance contains a value for each font variation available, even if that value is only the default font variation setting.

Your application must allocate enough memory in the `variation` parameter to store as many font variations as are available; you can determine this number by calling `GXCountFontVariations`. You can pass the `variation` parameter to `GXSetStyleFontVariation` or `GXSetShapeFontVariation` if you want to draw text with this font instance.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

Warnings

`index_out_of_range`

SEE ALSO

See the chapter “Typographic Styles” in this book for information on the `GXSetStyleFontVariation` or `GXSetShapeFontVariation` function, if you want to draw text with this font instance.

GXSetFontInstance

You can use the `GXSetFontInstance` function to add a font instance to a font or change an existing one.

```
long GXSetFontInstance(gxFont fontID, long index,
                      gxFontName name,
                      const gxFontVariation variation[]);
```

`fontID` A reference to the font in which to add or change a font instance.

Font Objects

<code>index</code>	The index of the instance you want to change. If the value of <code>index</code> is 0 and the value of <code>name</code> does not match any existing instance, <code>GXSetFontInstance</code> creates a new instance. If the value of <code>index</code> is greater than 0 or the value of <code>name</code> matches an existing instance, <code>GXSetFontInstance</code> replaces the instance with the values of the <code>name</code> and <code>variation</code> parameters.
<code>name</code>	The font name of the instance you want to change.
<code>variation</code>	An array of the variation data for the instance that you want to add to the font.

function result The index of the new or changed font instance.

DESCRIPTION

The `GXSetFontInstance` function adds a new font instance to a font or replaces the name and data for an existing instance.

The `GXSetFontInstance` function does not create the actual name for the instance; it only stores the ID of the font name for that instance. You must also call the `GXSetFontName` function if you are either changing the name of an existing instance or adding a new instance.

Each instance must have a complete set of font variations. A font instance contains a value for each font variation available, even if that value is only the default font variation setting.

IMPORTANT

The `GXSetFontInstance` function permanently changes the data in the font. In general, you should only call this function on fonts created by your application using `GXNewFont`. ▲

ERRORS, WARNINGS, AND NOTICES

Errors

<code>illegal_font_parameter</code>	
<code>font_cannot_be_changed</code>	
<code>parameter_out_of_range</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	
<code>inconsistent_parameters</code>	(debugging version)

Warnings

<code>font_table_not_found</code>

SEE ALSO

If you call `GXSetFontInstance` to create a font instance, you should then call `GXSetFontName` (page 7-41) to create the name for the instance. To get an unused name ID for a new instance, call `GXNewFontNameID` page 7-40.

GXDeleteFontInstance

You can use the `GXDeleteFontInstance` function to delete a font instance from a font.

```
long GXDeleteFontInstance(gxFont fontID, long index,
                          gxFontName nameID);
```

`fontID` A reference to the font from which you want to delete a font instance.

`index` The index of the font instance you want to delete. If this value is greater than 0, it specifies the instance to be deleted. If it is 0, then `GXDeleteFontInstance` deletes the instance that matches the value of `nameID`.

`nameID` The font name of the font instance you want to delete.

function result The index of the deleted instance. If the function does not delete a font instance, it returns 0.

DESCRIPTION

The `GXDeleteFontInstance` function permanently removes the specified font instance from the font and changes the index values for the following font instances.

IMPORTANT

The `GXDeleteFontInstance` function permanently changes the data in the font. In general, you should only call this function for fonts created by your application using `GXNewFont`. ▲

The `GXDeleteFontInstance` function does not delete the name of the instance. If you call `GXDeleteFontInstance`, you should then call `GXDeleteFontName` to delete the actual name.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>illegal_font_parameter</code>	
<code>font_cannot_be_changed</code>	
<code>inconsistent_parameters</code>	(debugging version)
<code>parameter_out_of_range</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	

Warnings

`font_table_not_found`

Manipulating Font Features

QuickDraw GX allows you to work with font features in several ways. You can

- determine the number of font features available in a font by (GXCountFontFeatures)
- get the name of a specific font feature by specifying its index in the font's list of features (GXGetFontFeature)
- get a specific font feature from a font by specifying its feature type (GXFindFontFeature)

You cannot add font features to or delete font features from a font.

Font features are described in the chapter “Layout Styles” in this book.

GXCountFontFeatures

You can use the GXCountFontFeatures function to determine the number of font features available in a font.

```
long GXCountFontFeatures(gxFont fontID);
```

`fontID` A reference to the font whose features you want to count.

function result The number of font features types available in the font.

DESCRIPTION

You can use the GXCountFontFeatures function to determine the number of font features available in a font.

Note

These are feature types and not feature type/feature selector pairs. Thus, a font supporting only common and rare ligatures would return 1 (for ligature type) and not 2 for the selectors. ♦

ERRORS, WARNINGS, AND NOTICES

Errors

```
out_of_memory
internal_font_error
illegal_font_parameter
```


SEE ALSO

To iterate through all of the font features in a font, use `GXGetFontFeature`, described next.

To search for a specific font feature, use the `GXFindFontFeature` function, described on page 7-62.

GXGetFontFeature

You can use the `GXGetFontFeature` function to get the name of a specific font feature by specifying its index in the font's list of features.

```
gxFontName GXGetFontFeature(gxFont fontID, long index,
                             gxFontFeatureFlag* flags,
                             long* settingCount,
                             gxFontFeatureSetting settings[],
                             gxFontFeature* feature);
```

<code>fontID</code>	A reference to the font from which you want to retrieve a font feature name.
<code>index</code>	The index of the font feature you are searching for.
<code>flags</code>	Flags associated with the font feature specified. This is returned to the caller if not <code>nil</code> .
<code>settingCount</code>	The number of settings for this font feature. This is returned to the caller if not <code>nil</code> .
<code>settings</code>	The settings of this font feature. This is returned to the caller if not <code>nil</code> . Your application is responsible for the storage.
<code>feature</code>	The font feature. Returned to the caller if not <code>nil</code> .

function result The name ID of the font feature.

DESCRIPTION

The `GXGetFontFeature` function takes a font and an index in the font's list of font features and returns the name ID of the feature. You can use this function to build a menu of font features.

If the function result is not `nil`, you can use it to get the text of the feature's name using the `GXFindFontName` function. This function returns the text of the name of the feature in a readable string, possibly in one or several encodings.

The application is responsible for allocating sufficient space for the settings variable, based on the settings count.

Font Objects

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

Warnings

`index_out_of_range`

SEE ALSO

The `GXFindFontName` function is described on page 7-39.

To search for a specific font feature, use the `GXFindFontFeature` function, described next.

To get the number of font features available in the font, use the `GXCountFontFeatures` function, described on page 7-60.

GXFindFontFeature

You can use the `GXFindFontFeature` function to get a specific font feature from a font by specifying its feature type.

```

gxFontName GXFindFontFeature(gxFont fontID, gxFontFeature feature,
                             gxFontFeatureFlag* flags,
                             long* settingCount,
                             gxFontFeatureSetting settings[],
                             long* index);

```

<code>fontID</code>	A reference to the font from which you want to retrieve a font feature name.
<code>feature</code>	The feature type of the desired feature.
<code>flags</code>	On return, the feature's flags. This is returned to the caller if not <code>nil</code> .
<code>settingCount</code>	The number of settings for the specified feature. This is returned to the caller if not <code>nil</code> .
<code>settings</code>	The settings for the specified feature. This is returned to the caller if not <code>nil</code> .
<code>index</code>	The index of the feature in the font. This is returned to the caller if not <code>nil</code> .

function result The name ID of the feature in the font. If the font does not contain the desired feature, the function returns 0.

DESCRIPTION

The `GXFindFontFeature` function takes a font ID and feature type, and returns the name ID of the specified feature in the font.

You can use the `GXFindFontName` function with the function result to get a readable string that is the name of the feature.

The application is responsible for allocating sufficient space for the settings variable, based on the settings count.

ERRORS, WARNINGS, AND NOTICES**Errors**

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

SEE ALSO

The `GXFindFontName` function is described on page 7-39.

To iterate through all of the font features in a font, use the `GXGetFontFeature` function, described on page 7-61.

To get the number of font features available in the font, use the `GXCountFontFeatures` function, described on page 7-60.

Advanced Font Functions

The functions described in this section allow you to alter the basic tables of the font or manipulate the font feature data in the font. These advanced font functions are useful primarily for font-related applications, such as font editors. In most cases, general-purpose applications, such as word processors, do not use these functions.

Adding, Removing, and Flattening Fonts

You can add a new font to the list of available fonts using `GXNewFont` and remove it using `GXDisposeFont`. You can also flatten the shape of a font using `GXFlattenFont`.

GXNewFont

You can use the `GXNewFont` function to create a font object and add it to the list of available fonts.

```
gxFont GXNewFont(gxFontStorageTag storage,
                 gxFontStorageReference reference,
                 gxFontAttribute attributes);
```

storage The storage type of the font you are adding. Possible storage type values are listed in “Font Storage Tags” on page 7-31.

reference The storage reference to the font data, usually an 'sfnt' resource.

attributes The attribute for the font you are adding.

function result A reference to the font you are adding.

DESCRIPTION

The `GXNewFont` function adds a font to the list of registered fonts. `GXNewFont` does not make a copy of the font’s data or create a new resource, file, or handle. The font’s data is specified in the `reference` parameter.

You must balance a call to `GXNewFont` with a call to `GXDisposeFont`, described next, when your application no longer needs to use that font.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>illegal_font_storage_type</code>	(debugging version)
<code>illegal_font_storage_reference</code>	(debugging version)
<code>illegal_font_attributes</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	

SEE ALSO

Storage types and storage references are discussed in “How Font Objects Are Stored and Referenced” on page 7-13.

GXDisposeFont

You can use the `GXDisposeFont` function to remove a font from the list of available fonts.

```
void GXDisposeFont(gxFont fontID);
```

`fontID` A reference to the font you want to remove from the list of registered fonts.

DESCRIPTION

The `GXDisposeFont` function removes a font from the list of available fonts. The font must be one you have added to the list.

IMPORTANT

You should *not* call `GXDisposeFont` for a system font. You should call it only for fonts you have created or added by means of `GXNewFont`. ▲

The `GXDisposeFont` function frees private storage and caches allocated by the system. Your application is responsible for disposing the actual font data.

ERRORS, WARNINGS, AND NOTICES

Errors

`illegal_font_parameter`

GXFlattenFont

You use the `GXFlattenFont` function to flatten all or part of a font object. You flatten a font to be embed it into a spool file.

```
void GXFlattenFont(gxFont source, struct scalerStream* stream,
                  struct gxSpoolBlock* block);
```

`source` A reference to the font you want to flatten.

`stream` The scaler stream you want for the font.

`block` The spool block.

DESCRIPTION

The `GXFlattenFont` function takes a font, a `gxScalerStream` structure, and spool block and flattens the font so that you can include the flattened font with a flattened shape. The `scalerStream` structure is used only by font-scaling programs and is not described here.

Font Objects

ERRORS, WARNINGS, AND NOTICES

Errors

out_of_memory	
internal_font_error	
illegal_font_parameter	
unflattening_interrupted_by_client	
null_font_scaler_context	
null_font_scaler_input	
invalid_font_scaler_context	
invalid_font_scaler_input	
invalid_font_scaler_font_data	
font_scaler_newblock_failed	
font_scaler_getfonttable_failed	
font_scaler_bitmap_allocation_failed	
font_scaler_outline_allocation_failed	
required_font_scaler_table_missing	
unsupported_font_scaler_outline_format	
unsupported_font_scaler_stream_format	
unsupported_font_scaler_font_format	
font_scaler_hinting_error	
font_scaler_rasterizer_error	
font_scaler_internal_error	
font_scaler_invalid_matrix	
font_scaler_fixed_overflow	
font_scaler_api_version_mismatch	
font_scaler_streaming_aborted	
unknown_font_scaler_error	
spoolProcedure_is_nil	(debugging version)
parameter_out_of_range	(debugging version)
inconsistent_parameters	(debugging version)

SEE ALSO

The `GXSpoolBlock` structure is described in the chapter “Shape Objects” in *Inside Macintosh: QuickDraw GX Objects*.

Getting and Setting Basic Font Storage Information

You can get a font’s storage type, storage reference, and font attributes using the `GXGetFont` function. You can then use these values to get the font ID using the `GXFindFont` function. In addition, you can change a font’s storage type, storage reference, and font attribute using the `GXSetFont` function.

GXGetFont

You can use the `GXGetFont` function to get a font's storage type, storage reference, and font attributes.

```
gxFontStorageTag GXGetFont(gxFont fontID,
                           gxFontStorageReference *reference,
                           gxFontAttribute *attributes);
```

fontID A reference to the font whose storage reference you want.

reference On return, a reference to the data of the font you specify. If you set this parameter to `nil`, the function ignores it.

attributes The font attributes, returned by the function. If you set this parameter to `nil`, the function ignores it.

function result The storage type of the font specified by `fontID`.

DESCRIPTION

The `GXGetFont` function returns the storage reference of the data storage type, and attributes of the font you specify. You should call this function if your application needs to know the font's storage type or storage reference.

ERRORS, WARNINGS, AND NOTICES

Errors

`illegal_font_parameter`

GXFindFont

You can use the `GXFindFont` function to get a single font with a particular storage type and storage reference.

```
gxFont GXFindFont(gxFontStorageTag storage,
                  gxFontStorageReference reference,
                  gxFontAttribute* attributes);
```

storage The storage type of the font you want to find.

reference The storage reference of the font you want to find.

attributes On return, the font attributes. If you set this parameter to `nil`, the function ignores it.

function result A reference to the font that matches the values given in the `storage` and `reference` parameters.

Font Objects

DESCRIPTION

The `GXFindFont` function returns the font with the storage type and reference you specify. If the `attribute` parameter is not `nil`, `GXFindFont` copies the font's attributes into it. If no matching font is found, `GXFindFont` returns `nil`.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>illegal_font_storage_type</code>	(debugging version)
<code>out_of_memory</code>	
<code>illegal_font_parameter</code>	

GXSetFont

You can use the `GXSetFont` function to change a font's storage type, storage reference, and font attributes.

```
void GXSetFont(gxFont fontID, gxFontStorageTag storage,
               gxFontStorageReference reference,
               gxFontAttribute attributes);
```

<code>fontID</code>	A reference to the font whose storage information you want to change.
<code>storage</code>	The storage type you want to assign to the font.
<code>reference</code>	The storage reference you want to assign to the font.
<code>attributes</code>	On return, the font attributes. If you set this parameter to <code>nil</code> , the function ignores it.

DESCRIPTION

The `GXSetFont` function changes a font's storage type and storage reference to the values you provide. If you want to change only one of these values, you should call `GXGetFont` to get the original storage reference or storage type values and include it in the proper parameter of `GXSetFont`.

SPECIAL CONSIDERATIONS

You should not use the `GXSetFont` function on a system font object, because you should not change the information of any font object other than those created by your application.

ERRORS, WARNINGS, AND NOTICES

Errors	
<code>illegal_font_storage_type</code>	(debugging version)
<code>illegal_font_attributes</code>	(debugging version)
<code>illegal_font_storage_reference</code>	(debugging version)
<code>illegal_font_parameter</code>	

SEE ALSO

The `GXGetFont` function is described on page 7-67.

GXGetFontFormat

You can use the `GXGetFontFormat` function to determine the internal format of a font.

```
gxFontFormatTag GXGetFontFormat(gxFont fontID);
```

`fontID` A reference to the font whose format you want.

function result The type of font scaler the font uses, in the form of a 4-byte tag.

DESCRIPTION

The `GXGetFontFormat` function returns a tag that specifies the font scaler of the font. This tag corresponds to the type of font scaler QuickDraw GX uses when drawing this font. Here are possible values:

Scaler tag	Font scaler
<code>'true'</code>	TrueType
<code>'typ1'</code>	Adobe Type 1
<code>'nfnt'</code>	<code>'NFNT'</code> bitmapped font

Other tags may refer to other font scalers.

ERRORS, WARNINGS, AND NOTICES

Errors	
<code>out_of_memory</code>	
<code>internal_font_error</code>	

Manipulating Font Tables

QuickDraw GX allows you to work with font tables in several ways. You can

- get the number of font tables present in a particular font (`GXCountFontTables`)
- retrieve part or all of a font table that you specify by index (`GXGetFontTable`, `GXGetFontTableParts`)
- retrieve part or all of a font table that you specify by table tag (`GXFindFontTable`, `GXFindFontTableParts`)
- add or change part or all of font table in a font (`GXSetFontTable`, `GXSetFontTableParts`)
- retrieve part or all of a font table (`GXDeleteFontTable`)

GXCountFontTables

You can use the `GXCountFontTables` function to get the number of font tables present in a particular font.

```
long GXCountFontTables(gxFont fontID);
```

`fontID` A reference to the font whose font tables you want to count.

function result The number of tables in the font.

DESCRIPTION

The `GXCountFontTables` function returns the number of tables in the font named by the `fontID` parameter.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

SEE ALSO

To retrieve the size and data of an entire font table that you specify by index, use the `GXGetFontTable` function, described next.

To retrieve the size and data of part of a font table that you specify by index, use `GXGetFontTableParts` function, described on page 7-72.

To retrieve the size and data of an entire font table that you specify by table tag, use the `GXFindFontTable` function, described on page 7-73.

To retrieve the size and data of part of a font table that you specify by table tag, use the `GXFindFontTableParts` function, described on page 7-74.

GXGetFontTable

You can use the `GXGetFontTable` function to retrieve an entire font table that you specify by index.

```
long GXGetFontTable(gxFont fontID, long index, void* tableData,
                   gxFontTableTag* tableTag);
```

<code>fontID</code>	A reference to the font from which you want to retrieve the font table.
<code>index</code>	The font table's index in the font's list of tables. The number of font tables in the font is returned by <code>GXCountFontTables</code> .
<code>tableData</code>	On return, the data of the specified font table, if this parameter is not <code>nil</code> . Your application is responsible for allocating the memory for this parameter.
<code>tableTag</code>	On return, the tag of the table you are searching for, if this parameter is not <code>nil</code> .

function result The size of the font table, in bytes. If the table is 0 bytes long, or if the index is out of range, `GXGetFontTable` returns 0.

DESCRIPTION

The `GXGetFontTable` function takes an index between the values of 1 and the number of tables in the font and returns the size of the font table.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

Warnings

`font_table_index_out_of_range`

SEE ALSO

The `GXCountFontTables` function is described on page 7-70.

To retrieve the size and data of part of a font table that you specify by index, use the `GXGetFontTableParts` function, described on page 7-72.

To retrieve the size and data of an entire font table that you specify by table tag, use the `GXFindFontTable` function, described on page 7-73.

Font Objects

To retrieve the size and data of part of a font table that you specify by table tag, use the `GXFindFontTableParts` function, described on page 7-74.

GXGetFontTableParts

You can use the `GXGetFontTableParts` function to retrieve part of a font table you specify by index.

```
long GXGetFontTableParts(gxFont fontID, long index, long offset,
                        long length, void* tableData,
                        gxFontTableTag* tableTag);
```

<code>fontID</code>	A reference to the font from which you want to retrieve part of a font table.
<code>index</code>	The font table's index in the font's list of tables. The number of font tables in the font is returned by <code>GXCountFontTables</code> .
<code>offset</code>	The starting position in the table of the data you want to retrieve.
<code>length</code>	The portion of the table you want to retrieve, in bytes.
<code>tableData</code>	On return, the data of the specified font table, if this parameter is not <code>nil</code> . Your application is responsible for allocating the memory for this parameter.
<code>tableTag</code>	The tag of the table you are searching for, returned by the function, if this parameter is not <code>nil</code> .

function result The number of bytes returned.

DESCRIPTION

The `GXGetFontTableParts` function returns part of the table you specify using an offset into the table and the number of bytes you want to retrieve. You can use `GXGetFontTableParts` to read part of a font table that is too large to fit entirely into memory.

If the value of the `offset` parameter added to the value of the `length` parameter is greater than the total length of the table, `GXGetFontTableParts` returns only the data from the `offset` value to the end of the table.

ERRORS, WARNINGS, AND NOTICES

Errors

`out_of_memory`
`internal_font_error`
`illegal_font_parameter`

Warnings

`font_table_index_out_of_range`

SEE ALSO

The `GXCountFontTables` function is described on page 7-70.

To retrieve the size and data of an entire font table that you specify by index, use the `GXGetFontTable` function, described on page 7-71.

To retrieve the size and data of an entire font table that you specify by table tag, use the `GXFindFontTable` function, described next.

To retrieve the size and data of part of a font table that you specify by table tag, use the `GXFindFontTableParts` function, described on page 7-74.

GXFindFontTable

You can use the `GXFindFontTable` function to retrieve an entire font table that you specify by table tag.

```
long GXFindFontTable(gxFont fontID, gxFontTableTag tableTag,
                    void* tableData, long* index);
```

`fontID` A reference to the font from which you want to retrieve the font table.

`tableTag` The tag of the table you want to retrieve.

`tableData` On return, the table is copied into this buffer.

`index` On return, the index of the table in the font.

function result The size of the table, in bytes. If no table has a tag that matches the value of `tableTag`, then the function returns 0.

DESCRIPTION

The `GXFindFontTable` function takes a font table tag and returns the size of that table, in bytes. If the value of `tableData` is not `nil`, then `GXFindFontTable` copies the font table into it. Your application is responsible for maintaining this buffer. If the `index` parameter is not `nil`, then `GXFindFontTable` copies the table's index into it.

ERRORS, WARNINGS, AND NOTICES**Errors**

`out_of_memory`

`internal_font_error`

`illegal_font_parameter`

SEE ALSO

To retrieve the size and data of part of a font table that you specify by table tag, use the `GXFindFontTableParts` function, described next.

Font Objects

To retrieve the size and data of an entire font table that you specify by index, use the `GXGetFontTable` function, described on page 7-71.

To retrieve the size and data of part of a font table that you specify by index, use the `GXGetFontTableParts` function, described on page 7-72.

GXFindFontTableParts

You can use the `GXFindFontTableParts` function to retrieve part of a font table you specify by table tag.

```
long GXFindFontTableParts(gxFont fontID, gxFontTableTag tableTag,
                          long offset, long length,
                          void* tableData, long* index);
```

<code>fontID</code>	A reference to the font from which you want to retrieve part of a font table.
<code>tableTag</code>	The tag of the table you want to retrieve.
<code>offset</code>	The starting position in the table of the data you want to retrieve, in bytes.
<code>length</code>	The amount of the table you want to retrieve, in bytes.
<code>tableData</code>	On return, the table is copied into this buffer.
<code>index</code>	On return, the index of the table in the font.

function result The number of bytes processed by the function.

DESCRIPTION

The `GXFindFontTableParts` function retrieves the part of the font table you specify by table tag. The function result is the number of bytes processed. You can use this function to read a table that is too large to fit in available memory.

If the value of `offset` added to the value of `length` exceeds the length of the table, then only the bytes from the `offset` value to the end of the table are processed. If the `index` parameter is not `nil`, then `GXFindFontTableParts` copies the table's index into it.

ERRORS, WARNINGS, AND NOTICES

Errors

```
out_of_memory
internal_font_error
illegal_font_parameter
```

SEE ALSO

To retrieve the size and data of an entire font table that you specify by table tag, use the `GXFindFontTable` function, described page 7-73.

To retrieve the size and data of an entire font table that you specify by index, use the `GXGetFontTable` function, described on page 7-71.

To retrieve the size and data of part of a font table that you specify by index, use the `GXGetFontTableParts` function, described on page 7-72.

GXSetFontTable

You can use the `GXSetFontTable` function to add or change an entire font table in a font.

```
long GXSetFontTable(gxFont fontID, long index,
                   gxFontTableTag tableTag,
                   long length, const void* tableData);
```

<code>fontID</code>	A reference to the font in which you want to add or replace a table.
<code>index</code>	The number of the table in the font's list of tables. (The number of font tables in the font is returned by <code>GXCountFontTables</code> .)
<code>tableTag</code>	The tag of the table you want to add or replace.
<code>length</code>	The length of the table you want to add or replace.
<code>tableData</code>	A pointer to the new data for the table.

function result The index of the table added or changed.

DESCRIPTION

The `GXSetFontTable` function replaces the existing table with the data in the `tableData` parameter or adds a new table. If the value of `index` is greater than 0 or the value of `tableTag` matches the table tag for an existing table, that table's data is resized to the value of the `length` parameter and replaced with the data in the `tableData` parameter. If the value of `index` is 0 and the value of `tableTag` does not match an existing table tag, `GXSetFontTable` adds a new table to the font with the tag specified in `tableTag` and the data from the `tableData` parameter.

If you enlarge or decrease a font table, QuickDraw GX is responsible for enlarging or decreasing the data in the font.

IMPORTANT

The `GXSetFontTable` function permanently changes the data in the font. In general, you should only call this function for fonts created by your application using `GXNewFont`. ▲

Font Objects

ERRORS, WARNINGS, AND NOTICES

Errors

<code>inconsistent_parameters</code>	(debugging version)
<code>illegal_font_parameter</code>	
<code>font_cannot_be_changed</code>	
<code>out_of_memory</code>	
<code>internal_font_error</code>	

Warnings

<code>font_table_index_out_of_range</code>
--

SEE ALSO

The `GXCountFontTable` function is described on page 7-70.

The `GXNewFont` function is described on page 7-64.

To change part of an existing font table instead of the entire table, use the `GXSetFontTableParts` function described next.

GXSetFontTableParts

You can use the `GXSetFontTableParts` function to change part of a font table by adding or replacing a part of the table.

```
long GXSetFontTableParts(gxFont fontID, long index,
                        gxFontTableTag tableTag, long offset,
                        long oldLength, long newLength,
                        const void* tableData);
```

<code>fontID</code>	A reference to the font whose font table is to be affected.
<code>index</code>	The index of the table in the list of font tables, if this value is greater than 0. If it is 0, <code>GXSetFontTableParts</code> searches by the value of the <code>tableTag</code> parameter.
<code>tableTag</code>	The tag of the table you want to add to or change.
<code>offset</code>	The number of bytes from the beginning of the table to the place where you want to begin replacing data in the table.
<code>oldLength</code>	The number of bytes you want to replace.
<code>newLength</code>	The number of bytes you want to add to the table.
<code>tableData</code>	The new data for the table.

function result The index of the table added or changed.

DESCRIPTION

The `GXSetFontTableParts` function replaces part of the table you specify using an offset from the beginning of the table, replacing the number of bytes in `oldLength` with the number of bytes in `newLength`.

If you enlarge or decrease a font table, QuickDraw GX enlarges or decreases the font.

IMPORTANT

The `GXSetFontTableParts` function permanently changes the data in the font. In general, you should only call this function for fonts created by your application using `GXNewFont`. ▲

ERRORS, WARNINGS, AND NOTICES**Errors**

<code>illegal_font_parameter</code>	
<code>font_cannot_be_changed</code>	
<code>inconsistent_parameters</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	

Warnings

<code>font_table_index_out_of_range</code>
--

SEE ALSO

To change the contents of an entire existing font table instead of just part of a font table, use the `GXSetFontTable` function described on page 7-75.

GXDeleteFontTable

You can use the `GXDeleteFontTable` function to delete a table from a font.

```
long GXDeleteFontTable(gxFont fontID, long index,
                      gxFontTableTag tableTag);
```

<code>fontID</code>	A reference to the font from which you want to delete the font table.
<code>index</code>	The index of the table you want to delete, if this value is greater than 0. If this value is 0, <code>GXDeleteFontTable</code> uses the value of the <code>tableTag</code> parameter to determine the correct table.
<code>tableTag</code>	The table tag of the font table you want to delete, if the value of the <code>index</code> parameter is 0.

function result The index of the table deleted. If the function does not delete a font table, it returns 0.

Font Objects

DESCRIPTION

The `GXDeleteFontTable` function deletes the table specified by `index`, if that value is greater than 0, or by `tableTag`, if the value of `index` is 0.

If you delete a font table, QuickDraw GX decreases the size of the font.

IMPORTANT

The `GXDeleteFontTable` function permanently changes the data in the font. In general, you should only call this function for fonts created by your application using `GXNewFont`. ▲

ERRORS, WARNINGS, AND NOTICES

Errors

<code>illegal_font_parameter</code>	
<code>font_cannot_be_changed</code>	
<code>inconsistent_parameters</code>	(debugging version)
<code>out_of_memory</code>	
<code>internal_font_error</code>	

Warnings

<code>font_table_index_out_of_range</code>
--

Changing Font Data

If you have changed the data in a font without using one of the QuickDraw GX font functions described in this chapter, you must call `GXChangedFont` to alert the system that you have changed the font.

GXChangedFont

You can use the `GXChangedFont` function if you have changed a font's data and want to alert the system that its private caches are invalid.

```
void GXChangedFont(gxFont fontID);
```

`fontID` A reference to the font you have changed.

DESCRIPTION

If you have changed a font's data directly (without using one of the QuickDraw GX font functions), you should call `GXChangedFont` to alert the system that its private caches are invalid.

QuickDraw GX automatically calls `GXChangedFont` when you alter a font using one of the system's font-editing functions, such as `GXSetFontTable`.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>illegal_font_parameter</code>

Summary of Font Objects

Basic Constants and Data Types

The Font Object

```
typedef struct gxPrivateFontRecord *gxFont;
```

Font Variations, Instances, and Descriptors

```
typedef long gxFontVariationTag;
```

```
typedef long gxFontDescriptorTag;
```

```
struct gxFontVariation {
    gxFontVariationTag name;
    Fixed    value;
} ;
```

```
typedef struct gxFontVariation gxFontDescriptor;
```

Font Names

```
enum gxFontNames {
    gxNoFontName,
    gxCopyrightFontName,
    gxFamilyFontName,
    gxStyleFontName,
    gxUniqueFontName,
    gxFullFontName,
    gxVersionFontName,
    gxPostscriptFontName,
    gxTrademarkFontName,
    gxManufacturerFontName,
    gxLastReservedFontName = 256
} ;
typedef long gxFontName;
```

Font Feature Flags

```
#define gxMutuallyExclusiveFeature 0x8000
```

Font Objects

Font Features

```
typedef long gxFontFeature;

struct gxFontFeatureSetting {
    unsigned short setting;
    unsigned short nameID;
}

typedef long gxFontFeatureFlag;
```

Font Platforms

```
enum gxFontPlatforms {
    gxGlyphPlatform = -1,
    gxNoPlatform,
    gxUnicodePlatform,
    gxMacintoshPlatform,
    gxReservedPlatform,
    gxMicrosoftPlatform,
    gxCustomPlatform,
} ;

typedef long gxFontPlatform;
```

QuickDraw GX Macintosh Scripts

```
enum gxUnicodeScripts {
    gxUnicodeDefaultSemantics = 1,
    gxUnicodeV1_1Semantics,
    gxISO10646_1993Semantics
} ;

enum gxMacintoshScripts {
    gxNoScript,
    gxRomanScript,
    gxJapaneseScript,
    gxTraditionalChineseScript,
    gxChineseScript = gxTraditionalChineseScript,
    gxKoreanScript,
    gxArabicScript,
    gxHebrewScript,
    gxGreekScript,
    gxCyrillicScript,
    gxRussianScript = gxCyrillicScript,
```

Font Objects

```

gxRSymbolScript,
gxDevanagariScript,
gxGurmukhiScript,
gxGujaratiScript,
gxOriyaScript,
gxBengaliScript,
gxTamilScript,
gxTeluguScript,
gxKannadaScript,
gxMalayalamScript,
gxSinhaleseScript,
gxBurmeseScript,
gxKhmerScript,
gxThaiScript,
gxLaotianScript,
gxGeorgianScript,
gxArmenianScript,
gxSimpleChineseScript,
gxTibetanScript,
gxMongolianScript,
gxGeezScript,
gxEthiopicScript = gxGeezScript,
gxAmharicScript = gxGeezScript,
gxSlavicScript,
gxEastEuropeanRomanScript = gxSlavicScript,
gxVietnameseScript,
gxExtendedArabicScript,
gxSindhiScript = gxExtendedArabicScript,
gxUninterpretedScript
} ;

typedef long gxFontScript;

enum gxCustomScripts {
    gxCustom8bitScript =1,
    gxCustom816bitScript,
    gxCustom16bitScript
};

enum gxMicrosoftScripts {
    gxMicrosoftSymbolScript =1,
    gxMicrosoftStandardScript
};

```

QuickDraw GX Macintosh Languages

```
enum gxMacintoshLanguages {
    gxNoLanguage,
    gxEnglishLanguage,
    gxFrenchLanguage,
    gxGermanLanguage,
    gxItalianLanguage,
    gxDutchLanguage,
    gxSwedishLanguage,
    gxSpanishLanguage,
    gxDanishLanguage,
    gxPortugueseLanguage,
    gxNorwegianLanguage,
    gxHebrewLanguage,
    gxJapaneseLanguage,
    gxArabicLanguage,
    gxFinnishLanguage,
    gxGreekLanguage,
    gxIcelandicLanguage,
    gxMalteseLanguage,
    gxTurkishLanguage,
    gxCroatianLanguage,
    gxTradChineseLanguage,
    gxUrduLanguage,
    gxHindiLanguage,
    gxThaiLanguage,
    gxKoreanLanguage,
    gxLithuanianLanguage,
    gxPolishLanguage,
    gxHungarianLanguage,
    gxEstonianLanguage,
    gxLettishLanguage,
    gxLatvianLanguage = gxLettishLanguage,
    gxSaamiskLanguage,
    gxLappishLanguage = gxSaamiskLanguage,
    gxFaeroeseLanguage,
    gxFarsiLanguage,
    gxPersianLanguage = gxFarsiLanguage,
    gxRussianLanguage,
    gxSimpChineseLanguage,
    gxFlemishLanguage,
    gxIrishLanguage,
    gxAlbanianLanguage,
```

Font Objects

```

gxRomanianLanguage,
gxCzechLanguage,
gxSlovakLanguage,
gxSlovenianLanguage,
gxYiddishLanguage,
gxSerbianLanguage,
gxMacedonianLanguage,
gxBulgarianLanguage,
gxUkrainianLanguage,
gxByelorussianLanguage,
gxUzbekLanguage,
gxKazakhLanguage,
gxAzerbaijaniLanguage,
gxAzerbaijanArLanguage,
gxArmenianLanguage,
gxGeorgianLanguage,
gxMoldavianLanguage,
gxKirghizLanguage,
gxTajikiLanguage,
gxTurkmenLanguage,
gxMongolianLanguage,
gxMongolianCyrLanguage,
gxPashtoLanguage,
gxKurdishLanguage,
gxKashmiriLanguage,
gxSindhiLanguage,
gxTibetanLanguage,
gxNepaliLanguage,
gxSanskritLanguage,
gxMarathiLanguage,
gxBengaliLanguage,
gxAssameseLanguage,
gxGujaratiLanguage,
gxPunjabiLanguage,
gxOriyaLanguage,
gxMalayalamLanguage,
gxKannadaLanguage,
gxTamilLanguage,
gxTeluguLanguage,
gxSinhaleseLanguage,
gxBurmeseLanguage,
gxKhmerLanguage,
gxLaoLanguage,

```

Font Objects

```

gxVietnameseLanguage,
gxIndonesianLanguage,
gxTagalogLanguage,
gxMalayRomanLanguage,
gxMalayArabicLanguage,
gxAmharicLanguage,
gxTigrinyaLanguage,
gxGallaLanguage,
gxOromoLanguage = gxGallaLanguage,
gxSomaliLanguage,
gxSwahiliLanguage,
gxRuandaLanguage,
gxRundiLanguage,
gxChewaLanguage,
gxMalagasyLanguage,
gxEsperantoLanguage,
gxWelshLanguage = 129,
gxBasqueLanguage,
gxCatalanLanguage,
gxLatinLanguage,
gxQuechuaLanguage,
gxGuaraniLanguage,
gxAymaraLanguage,
gxTatarLanguage,
gxUighurLanguage,
gxDzongkhaLanguage,
gxJavaneseRomLanguage,
gxSundaneseRomLanguage
} ;
typedef long gxFontLanguage;

#define gxLastCustomLanguage 256

```

Font Format Tag

```

typedef long gxFontFormatTag;

```


Advanced Constants and Data Types

Font Storage Tags and References

```
#define gxResourceFontStorage 0x72737263/* 'rsrc' */
#define gxHandleFontStorage 0x686e646c/* 'hndl' */
#define gxFileFontStorage 0x62617373/* 'bass' */
#define gxNfntFontStorage 0x6e666e74/* 'nfnt' */
```

```
typedef long gxFontStorageTag;
```

```
typedef void *gxFontStorageReference;
```

Font Table Tags

```
typedef long gxFontTableTag;
```

Font Attributes

```
enum gxFontAttributes {
    gxSystemFontAttribute = 0x0001,
    gxReadOnlyFontAttribute = 0x0002
};
typedef long gxFontAttribute;
```

Basic Font Functions

Getting the List of Available Fonts

```
long GXFindFonts          (gxFont family, gxFontName meaning,
                           gxFontPlatform platform, gxFontScript script,
                           gxFontLanguage language, long nameLength,
                           const unsigned char name[], long index,
                           long count, gxFont fonts[]);
```

Counting Glyphs in a Font

```
long GXCountFontGlyphs    (gxFont fontID);
```

Getting and Setting the Default Font

```
gxFont GXGetDefaultFont   (void);
gxFont GXSetDefaultFont   (gxFont fontID);
```

Font Objects

Manipulating Font Names

```

long GXCountFontNames      (gxFont fontID);
long GXGetFontName         (gxFont fontID, long index, gxFontName *name,
                           gxFontPlatform *platform, gxFontScript *script,
                           gxFontLanguage *language,
                           unsigned char text[]);
long GXFindFontName        (gxFont fontID, gxFontName name,
                           gxFontPlatform platform, gxFontScript script,
                           gxFontLanguage language, unsigned char text[],
                           long *index);
gxFontName GXNewFontNameID (gxFont fontID);
long GXSetFontName         (gxFont fontID, gxFontName name,
                           gxFontPlatform platform, gxFontScript script,
                           gxFontLanguage language, long nameLength,
                           const unsigned char text[]);
long GXDeleteFontName      (gxFont fontID, long index, gxFontName name,
                           gxFontPlatform platform, gxFontScript script,
                           gxFontLanguage language);

```

Manipulating Font Encodings

```

long GXCountFontEncodings  (gxFont fontID);
gxFontPlatform GXGetFontEncoding
                           (gxFont fontID, long index,
                           gxFontScript *script,
                           gxFontLanguage* language);
long GXFindFontEncoding    (gxFont fontID, gxFontPlatform platform,
                           gxFontScript script,
                           gxFontLanguage language);
long GXApplyFontEncoding   (gxFont fontID, long index, long* length,
                           const unsigned char text[], long count,
                           unsigned short glyphs[], char was16Bit[]);

```

Manipulating Font Descriptors

```

long GXCountFontDescriptors (gxFont fontID);
gxFontDescriptorTag GXGetFontDescriptor
                           (gxFont fontID, long index,
                           Fixed* descriptorValue);
long GXFindFontDescriptor  (gxFont fontID, gxFontDescriptorTag
                           descriptorTag, Fixed* descriptorValue);
long GXSetFontDescriptor   (gxFont fontID, long index,
                           gxFontDescriptorTag descriptorTag,
                           Fixed descriptorValue);

```

```
long GXDeleteFontDescriptor (gxFont fontID, long index,
                             gxFontDescriptorTag descriptorTag);
```

Manipulating Font Variations

```
long GXCountFontVariations (gxFont fontID);
gxFontVariationTag GXGetFontVariation
    (gxFont theFont, long index, Fixed* minValue,
     Fixed* defaultValue, Fixed* maxValue,
     gxFontName* nameID);
long GXFindFontVariation (gxFont theFont, gxFontTableTag variationTag,
                          Fixed* minValue, Fixed* defaultValue,
                          Fixed* maxValue, gxFontName* nameID);
```

Manipulating Font Instances

```
long GXCountFontInstances (gxFont, theFont);
gxFontName GXGetFontInstance (gxFont theFont, long index,
                              gxFontVariation variation[]);
long GXSetFontInstance (gxFont fontID, long index, gxFontName name,
                       const gxFontVariation variation[]);
long GXDeleteFontInstance (gxFont fontID, long index, gxFontName nameID);
```

Manipulating Font Features

```
long GXCountFontFeatures (gxFont fontID);
gxFontName GXGetFontFeature (gxFont fontID, long index,
                             gxFontFeatureFlag* flags, long* settingCount,
                             gxFontFeatureSetting settings[],
                             gxFontFeature* feature);
gxFontName GXFindFontFeature (gxFont fontID, gxFontFeature feature,
                              gxFontFeatureFlag* flags, long* settingCount,
                              gxFontFeatureSetting settings[], long* index);
```

Advanced Font Functions

Adding, Removing, and Flattening Fonts

```
gxFont GXNewFont (gxFontStorageTag storage,
                  gxFontStorageReference reference,
                  gxFontAttribute attributes);
void GXDisposeFont (gxFont fontID);
void GXFlattenFont (gxFont source,
                   struct gxScalerStream* stream, struct
                   gxSpoolBlock* block);
```

Font Objects

Getting and Setting Basic Font Storage Information

```

gxFontStorageTag GXGetFont    (gxFont fontID,
                                gxFontStorageReference *reference,
                                gxFontAttribute *attributes);

gxFont GXFindFont             (gxFontStorageTag storage,
                                gxFontStorageReference reference,
                                gxFontAttribute* attributes);

void GXSetFont                (gxFont fontID, gxFontStorageTag storage,
                                gxFontStorageReference reference,
                                gxFontAttribute attributes);

gxFontFormatTag GXGetFontFormat
                                (gxFont fontID);

```

Manipulating Font Tables

```

long GXCountFontTables        (gxFont fontID);

long GXGetFontTable           (gxFont fontID, long index, void* tableData,
                                gxFontTableTag* tableTag);

long GXGetFontTableParts      (gxFont fontID, long index, long offset,
                                long length, void* tableData,
                                gxFontTableTag* tableTag);

long GXFindFontTable          (gxFont fontID, gxFontTableTag tableTag,
                                void* tableData, long* index);

long GXFindFontTableParts     (gxFont fontID, gxFontTableTag tableTag,
                                long offset, long length, void* tableData,
                                long* index);

long GXSetFontTable           (gxFont fontID, long index,
                                gxFontTableTag tableTag, long length,
                                const void* tableData);

long GXSetFontTableParts      (gxFont fontID, long index,
                                gxFontTableTag tableTag, long offset,
                                long oldLength, long newLength,
                                const void* tableData);

long GXDeleteFontTable        (gxFont fontID, long index,
                                gxFontTableTag tableTag);

```

Changing Font Data

```

void GXChangedFont            (gxFont fontID);

```