

Advanced Printing Features

This chapter describes how your application can use printing-related objects in ways that may not be required for most applications. Read the information in this chapter if you want your application to read or modify print files after they have been printed, create and use custom paper types, or explicitly control the way that QuickDraw GX performs certain printing operations.

To use this chapter, you should also be familiar with the printing-related objects, including collection objects that QuickDraw GX uses to store job and format information, as introduced in the chapter “Introduction to Printing With QuickDraw GX” in this book. Because the objects and techniques discussed in this chapter build on applications that already provide core printing features, you should be familiar with these features, as introduced in the chapter “Introduction to Printing With QuickDraw GX” and discussed in detail in the “Core Printing Features” chapter of this book.

This chapter describes the concepts required to use advanced QuickDraw GX printing features and terms and then explains how to

- manipulate a job object; for example, using its reference constant property
- work with a printer object to obtain information about the device it represents, such as information about the driver, its resolution, and color printing capabilities
- manipulate a print file object that represents a spooled file or a portable digital document
- manipulate a paper-type object to define paper sizes for different requirements
- optimize printing for specific devices

This chapter also describes the resource for Printing Status dialog boxes, as well as status constants. Although you can customize Printing Status dialog boxes in your application, they are used primarily by printer drivers and printing extensions. For information about the use of Printing Status dialog boxes by printer drivers and printing extensions, see the resource chapter of *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers*.

About Advanced Printing Features

Advanced printing features make use of objects described in the chapters “Core Printing Features” and “Page Formatting and Dialog Box Customization” in this book. This chapter shows how these objects can be used in additional ways to implement features not typically required by every application that implements QuickDraw GX printing.

For example, the paper-type object is always associated with a format object. A paper type that matches the format is provided by QuickDraw GX as a core feature. Ordinarily, your application need not modify it. If, for example, your application needs to restrict the printable area of a page to reserve room for a letterhead, it can create a paper-type object that defines a new paper size. Although the technique is straightforward, the feature is considered advanced because applications are not required to create

Advanced Printing Features

paper-type objects. Typically, the default paper-type object is sufficient for most applications.

To implement other advanced printing features, you use the printer and print file objects.

- You can use a printer object to determine the device characteristics of a desktop printer, such as its resolution
- You can use a print file object to determine the contents of a file that has been printed and change them, if you wish.

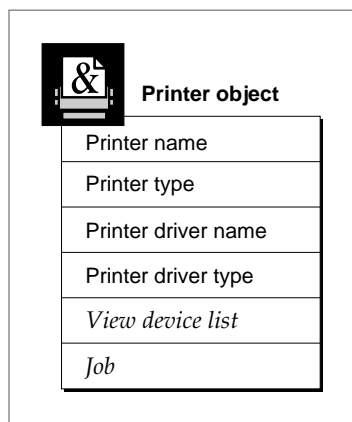
The following sections describe the printer and print file objects.

Printer Objects

Each job object references two printer objects. One printer object specifies the output printer on which the document is printed. The other printer object defines the formatting printer that specifies how the document is formatted. A user chooses an output printer in the Print dialog box and a formatting printer in the Page Setup dialog box. When a job object is created, its output printer is the currently selected desktop printer, and the formatting printer is specified by the output printer's printer driver.

Each printer object has six accessible properties, as shown in Figure 4-1. Note that, because a printer object is a private data structure, the order of the properties as shown in Figure 4-1 is completely arbitrary. Properties in italics indicate references to other objects.

Figure 4-1 The printer object



The properties of a printer object are:

- **Printer name.** This property contains the name of the printer. A user specifies a printer by name in the Print dialog box. For example, a user could choose the printer “My Printer” from the list of available printers.
- **Printer type.** This property specifies the creator type of a printer. It is a 4-character signature that uniquely identifies a kind of printer. You are responsible for registering the printer type with Developer Technical Support at Apple Computer. An example of a printer type is 'LWRW' for a LaserWriter.
- **Printer driver name.** This property specifies the name of the printer driver to which the job is printed. A user specifies a printer driver from the Chooser if the desired printer is not already on the desktop.
- **Printer driver type.** This property specifies the kind of printer driver. Table 4-1 shows some printer driver types provided by QuickDraw GX.
- **View device list.** This property contains a list of references to the view devices associated with a printer. Each view device specifies a print resolution (dots-per-inch) and color space (for example, CMYK or a grayscale space) that is supported by the printer. For more information about the relationship between printer objects and view devices, see the section “Printer View Devices” beginning on page 4-8.
- **Job.** This property contains a reference to a job object. Through this reference, you can access a job object associated with a printer object. The job object is discussed in the chapter “Core Printing Features” in this book.

Printer Driver Types

Table 4-1 shows the printer driver types defined by QuickDraw GX. Do not make assumptions about the kinds of service provided by a printer driver based on its type alone. For example, two PostScript drivers may be subtly different.

Table 4-1 Printer driver types

Constant	Value	Explanation
<code>gxAnyPrinterType</code>	'univ'	Universal type of printer
<code>gxRasterPrinterType</code>	'rast'	Raster printer
<code>gxPostscriptPrinterType</code>	'post'	Postscript printer
<code>gxVectorPrinterType</code>	'vect'	Vector printer
<code>gxPortableDocPrinterType</code>	'gxpd'	Portable digital document maker
	'????'	Unknown driver type

Note

You are responsible for registering your printer driver type with Developer Technical Support. ♦

Printer View Devices

A printer object's view device list specifies the resolutions and color spaces that can be used with a printer. These view devices are created by the printer driver. Your application can access, but not change, these characteristics. The printer's resolution is stored in the view device's mapping property as the scaling factor. The printer's color space is stored in the bitmap shape that represents the imageable area of the device. A view device object contains other properties as well; however, these properties are not used in printing. For more information about view device objects, see the view-related objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

For example, the LaserWriter IISC GX driver creates a view device list with only one view device, because the printer supports only one color space, black-and-white, and one resolution, 300 dots-per-inch. The view device's mapping property specifies a scaling factor of 4.17, both horizontally and vertically, to achieve the 300 dots-per-inch resolution. The scaling factor is determined by dividing the printer's resolution, 300 dots-per-inch, by 72, which represents the resolution when the scaling factor is 1.

As another example, the ImageWriter II GX printer driver supports printing at two resolutions in each of two color spaces:

- 144 dpi, with a 4-bit indexed CMYK (cyan, magenta, yellow, black) color space
- 144 dpi, 1-bit indexed color space
- 72 dpi, with a 4-bit indexed CMYK color space
- 72 dpi, 1-bit indexed color space

The driver creates a view device list with four view device references. The printer driver sets up the mapping property in each view device to specify the correct scaling factor. For an example of how to obtain the scaling factor, see the section "Determining a Printer's Resolution" on page 4-26.

Note

A printer driver inherits a view device for a 72 dpi, 24-bit RGB color space from QuickDraw GX and modifies the list as necessary to include the view devices that the driver actually supports. For more information about writing a printer driver, see the printer driver chapter of *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers*. ♦

Color Matching for Printers

QuickDraw GX provides a color profile object that is used to specify color-matching information for a printer. The color profile object is discussed in the color and color-related objects chapter of *Inside Macintosh: QuickDraw GX Objects*. Your application can access the color profile object associated with a printer driver or a particular page of output or set these color profile objects using the following functions:

Function	Purpose
GXFindPrinterProfile	Determine the color profile for a printer
GXFindFormatProfile	Determine the color profile for a format object
GXSetPrinterProfile	Set the color profile for a printer
GXSetFormatProfile	Set the color profile for a format object

For more information about these functions, see “Color Profile Functions” beginning on page 4-84. For an example of retrieving the color profile and color space from a view device, see “Retrieving the Color Profile and Color Space for a Printer” on page 4-27.

Print File Objects

A **print file object** represents the file that QuickDraw GX creates when your application prints a document. If the document is printed to a printer, the print file contains the spooled input to the printer driver. If the document is printed as a portable digital document, the print file’s contents are kept in an application-independent form along with data, such as font information, that allows the document to be viewed without the application that created it.

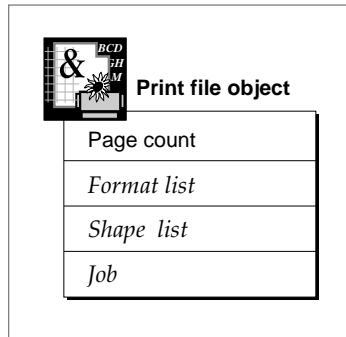
You can use print file objects to

- open, save, and close print files
- retrieve the contents of a print file or the formats associated with it
- count the pages in a print file, and insert, replace, and delete pages
- retrieve the job object stored with the print file

Advanced Printing Features

Print file objects have four accessible properties, as shown in Figure 4-2. Note, that because a print file object is a private data structure, the order of the properties as shown in Figure 4-2 is completely arbitrary. Properties in italics indicate references to other objects.

Figure 4-2 The print file object



The properties of a print file object are:

- **Page count.** This property contains the number of pages in the print file.
- **Format list.** This property contains a list of references to format objects, one reference for each page in the file. The first reference is the default format for the print job.
- **Shape list.** This property contains a list of references to shape objects, one reference for each page in the file. Each page is stored as a picture shape in the file, whether the file was created page-by-page or shape-by-shape for each page. Thus, the first reference in the list is the picture shape for the first page, the second reference is the shape for the second page, and so on.
- **Job.** This property is a reference to the job object associated with the file when it is open. The properties of this job object match those of the job object used to create the print file.

Synonyms

You can use **synonyms** to provide alternative printing directives instead of those generated by QuickDraw GX. You are never required to use a synonym. They are available for you to use if you want to explicitly control the way that QuickDraw GX renders output.

For example, if you have special-purpose PostScript code for printing a shape and wish to use it instead of the PostScript code that QuickDraw GX produces, you can create a synonym for your code and attach it to the shape object. When the shape is printed, the instructions associated with the synonym can be used to render the output.

If you use a synonym, the printer driver also must support its use; otherwise, the synonym is ignored. The synonym is interpreted by the printer driver; thus one printer driver may choose to implement a synonym using PostScript and another printer driver might use a proprietary language to implement the same synonym.

You use a synonym by creating a tag object and setting up a reference to that tag in the shape object or another kind of object. A tag object is a QuickDraw GX object that provides you with the ability to associate data with objects, such as shapes, styles, inks, colors, and transforms. For more information about tag objects, see the tag objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

QuickDraw GX provides five kinds of synonyms:

- Direct PostScript synonyms, which allow you to explicitly specify PostScript operators for rendering images. You can use these synonyms with shape, style, ink, and transform objects to control the behavior of these objects when printing.
- Style synonyms, such as dashes, line caps, or patterns that can be associated with style objects.
- Halftone synonyms, which specify the halftone to be applied when a shape or page is printed. For general information about halftones, see the view-related objects chapter of *Inside Macintosh: QuickDraw GX Objects*.
- Cubic synonyms, which provide alternative directives for rendering path shapes. For information about path shapes, see the geometric shapes chapter of *Inside Macintosh: QuickDraw GX Graphics*.
- Picture synonyms, which specify QuickDraw picture data for rendering pages. For example, QuickDraw GX uses picture synonyms to spool the output of documents designed for printing with the Macintosh Printing Manager.

Note

Synonyms remain with the shape or a related object, such as the shape's ink, style, or transform. If you cut or copy a shape and then paste it, the synonyms in the tag objects associated with the shape move with the shape. Synonyms also stay with the shape if the print file that contains the shape is redirected. ♦

Table 4-2 identifies the synonyms that QuickDraw GX provides.

Table 4-2 QuickDraw GX printing synonyms

Constant	Value	Explanation
<code>gxPostScriptTag</code>	<code>'post'</code>	Specifies PostScript operators
<code>gxPostControlTag</code>	<code>'psct'</code>	Specifies control information for a PostScript printer
<code>gxDashSynonymTag</code>	<code>'sdsh'</code>	Specifies dashes, for example, with the PostScript <code>setdash</code> operator
<code>gxLineCapSynonymTag</code>	<code>'lcap'</code>	Specifies line caps, for example, with the PostScript <code>setlinecap</code> operator
<code>gxPatternSynonymTag</code>	<code>'ptrn'</code>	Specifies a pattern, for example, on vector devices
<code>gxFormatHalftoneTag</code>	<code>'half'</code>	Specifies halftones, for example, the PostScript halftone graphics state
<code>gxCubicSynonymTag</code>	<code>'cubx'</code>	Specifies a cubic representation for a path
<code>gxQuickDrawPictTag</code>	<code>'pict'</code>	Specifies a shape in QuickDraw picture format

The following sections describe the contents of the tag objects that you create for each of these synonyms. For an example of how to use a synonym, see “Using Synonyms,” which begins on page 4-38.

General-Purpose PostScript Operator Synonym

If you want QuickDraw GX to use your own PostScript operators for rendering an object, you may create a `gxPostScriptTag` synonym and attach it to the object. If you only need to specify specific operators or set up the halftone graphics state, you may be able to use one of the special-purpose synonyms listed in Table 4-2.

You can reference a tag object that contains the `gxPostScriptTag` synonym from a shape object, style object, ink object, or transform object. The kind of object that references the tag object controls the kind of PostScript operators you can use.

- With a shape object, you can use PostScript printing operators to render the shape.
- With a style object, you can use PostScript operators to define all stylistic characteristics for the shapes that refer to the style object.

Advanced Printing Features

- With an ink object, you can use PostScript operators to define the color and transfer mode for the shapes that refer to the ink object.
- With a transform object, you can use PostScript operators to define the clip and mapping of the shapes that refer to the transform object. The `gxPostScriptTag` synonym may be ignored under certain conditions, such as when a transform object's mapping changes the perspective.

The data in the `gxPostScriptTag` synonym is pure PostScript code that is generated as one continuous PostScript data stream. There is no data type that defines the structure of this synonym. You can attach multiple tag objects to an object. This allows you to distribute data into smaller, more manageable pieces that require less memory to load. For best results, you should limit the data in a `gxPostScriptTag` synonym to 8 KB.

If you choose to write your own PostScript code, it is extremely important to make your PostScript code portable, especially if users create portable digital documents. To create portable PostScript code, try to follow these guidelines:

- Write PostScript code so that it runs on output devices that support Level 1 and devices that support Level 2.
- Do not make assumptions about the current “PostScript state” of the output device.
- Do not make assumptions about the fonts that are installed in the output device.

Note

The y-axis of the QuickDraw GX coordinate system is the reverse of the y-axis of the PostScript coordinate system. ♦

PostScript Control Information Synonym

A shape object can refer to a tag object that contains the `gxPostControlTag` synonym. The synonym includes flags that indicate how to modify the PostScript graphics state.

The `gxPostControlTag` synonym provides data specific to PostScript devices that may be necessary for these devices to properly render the data contained within the `gxPostScriptTag` synonym. You are not required, however, to have a `gxPostControlTag` synonym when you use `gxPostScriptTag` synonyms.

A shape object can refer to, at most, only one `gxPostControlTag` synonym. Information in this synonym affects all `gxPostScriptTag` synonyms attached to a shape object.

Advanced Printing Features

The `gxPostControl` structure defines the contents of a `gxPostControlTag` synonym:

```
struct gxPostControl {
    long flags;
};
```

Field descriptions

flags A flag that specifies how a shape is embedded in the PostScript data stream. If it is `gxNoSave`, the PostScript data should be encapsulated between a save and restore combination. If `gxNoSave` is not specified or the `gxPostControlTag` synonym is not present, the save and restore combination is used.

Dash Synonym

A style object can refer to a tag that contains the `gxDashSynonymTag` synonym. This tag causes QuickDraw GX to print simple dashes. For example, this synonym may cause the printer driver to use the PostScript `setdash` operator instead of the specification in the dash property of the style. The phase for the `setdash` operator might still be taken from the phase value stored in the dash property of the style object.

The `gxDashSynonym` structure defines the contents of a `gxDashSynonymTag` synonym:

```
struct gxDashSynonym {
    long size;
    fixed dashLength[gxAnyNumber];
};
```

Field descriptions

size The number of elements in a dash array.

dashLength An array of lengths for the dashes.

Line Cap Synonym

A style object can refer to a tag that contains the `gxLineCapSynonym` synonym. For example, this synonym may cause the printer driver to print with the PostScript `linecap` operator instead of the specification in the cap property of the style.

The `gxLineCapSynonym` structure defines the `gxLineCapSynonymTag` synonym:

```
typedef long gxLineCapSynonym;
```

The structure is a long word that specifies one of the values in the `gxLineCaps` enumeration:

```
enum gxLineCaps{
    gxButtCap = 0,
    gxRoundCap = 1,
```

Advanced Printing Features

```

    gxSquareCap = 2,
    gxTriangleCap = 3
};

```

Constant descriptions

<code>gxButtCap</code>	Use a square cap, such as the PostScript butt cap, for the line cap.
<code>gxRoundCap</code>	Use a round cap, such as the PostScript round cap, for the line cap.
<code>gxSquareCap</code>	Use a square cap, such as the PostScript projecting square cap, for the line cap.
<code>gxTriangleCap</code>	Use a triangle cap.

Halftone Synonym

QuickDraw GX supports halftones to represent more colors than can be represented on a printer by alternating available colors in a fixed cell size to represent more colors. QuickDraw GX, by default, chooses the appropriate halftone for you; however, you can choose to specify halftone information on a shape-by-shape or page-by-page basis yourself.

To provide halftone information for a particular shape object, the shape's ink object must refer to a tag object that contains the `gxFormatHalftoneTag` synonym. This allows halftones to be specified for individual inks. Shapes that are drawn with the same ink use the same halftone. An ink that does not refer to the `gxFormatHalftoneTag` synonym uses the page's halftone.

Note

If you specify halftone information on a page-by-page basis, you use the `format-halftone` property in the format collection associated with the page's format. For more information about this property, see the chapter "Page Formatting and Dialog Box Customization" in this book. ♦

The `gxFormatHalftoneInfo` structure defines the contents of a `gxFormatHalftoneTag` synonym:

```

struct gxFormatHalftoneInfo {
    long          numHalftones;
    gxHalftone    halftones[1];
};

```

Field descriptions

<code>numHalftones</code>	The number of halftones available for use.
<code>halftones</code>	The array of halftone specifications.

Advanced Printing Features

Halftones are specified in the `gxHalftone` structures, which are described completely in the view-related objects chapter of *Inside Macintosh: QuickDraw GX Objects*:

```
struct gxHalftone{
    fixed        angle;           /* direction of halftone */
    fixed        frequency;       /* cells per inch */
    gxDotType     method;         /* kind of pattern */
    gxTintType    tinting;        /* tint calculation method */
    gxColor       dotColor;       /* color of foreground */
    gxColor       backgroundColor; /* color of background */
    gxColorSpace  tintSpace;      /* color space for tint */
};
```

You can specify any number of halftones. QuickDraw GX selects appropriate halftones from the list of available halftones. Its selection is based upon the `tinting` field in the halftone structures:

- When you print to a black-and-white PostScript device, QuickDraw GX looks for a halftone structure that specifies `gxLuminanceTint` in the `tinting` field. If no halftone specifies this value, it looks for a halftone specifies `gxComponent4Tint` as its tinting method. Component 4 is the black component in the CMYK (cyan, magenta, yellow, and black) space. If no halftone specifies this tinting method either, the first halftone in the list is used.
- When you print to a color PostScript device, a maximum of four halftones are used. QuickDraw GX attempts to locate halftones for the following tint calculation methods: `gxComponent1Tint` for the cyan halftone, `gxComponent2Tint` for the magenta halftone, `gxComponent3Tint` field for the yellow halftone, and `gxComponent4Tint` for the black halftone. If a tinting method is in the list more than once, the first one in the list is used.

If a halftone for the `gxComponent4Tint` method is not in the list, QuickDraw GX uses the `gxLuminanceTint` tinting method for the black halftone. If the `gxLuminanceTint` tinting method cannot be found either, QuickDraw GX uses the first halftone in the list for the black halftone.

If QuickDraw GX cannot find a halftone for the `gxComponent1Tint`, `gxComponent2Tint`, or `gxComponent3Tint` tinting methods, it uses the black halftone for the missing tinting method.

It is only possible to use halftones to the extent that a particular PostScript device supports them. The dot color and background color of a halftone are ignored because QuickDraw GX assumes that the dot color for a black-and-white device is black and the dot color for a color device with the `gxComponent2Tint` tinting method is magenta.

Note

Continuous tone output devices, such as a 32-bit color printer, may choose to ignore the halftone synonym because halftones are not needed on these output devices. ♦

Pattern Synonym

A style object can refer to a tag object that contains the `gxPatternSynonymTag` synonym. This synonym causes QuickDraw GX to print with the pattern specified in the tag instead of the specification in the pattern property of the style. For example, vector devices typically support crosshatch patterns.

The `gxPatternSynonymTag` structure defines the contents of a `gxPatternSynonymTag` synonym:

```
struct gxPatternSynonym {
    long      patternType;
    fixed     angle;
    fixed     spacing;
    fixed     thickness;
    gxPoint   anchorPoint;
};
```

Field descriptions

<code>patternType</code>	The pattern type, either <code>gxHatch</code> or <code>gxCrossHatch</code> .
<code>angle</code>	The angle of the lines in the pattern.
<code>spacing</code>	The distance between the lines in the pattern.
<code>thickness</code>	The thickness of the lines in the pattern.
<code>anchorPoint</code>	A point that specifies the upper-left corner at which the pattern begins.

Cubic Synonym

A path shape object can refer to a tag object that contains the `gxCubicSynonymTag` synonym. This synonym causes QuickDraw GX to print with a representation of the shape using cubics, such as Bezier curves, instead of the quadratic Bezier curves specified in the shape's geometry.

The data in this synonym is ignored, however, when

- it is attached to any shape object other than a path
- the shape object's transform hierarchy changes the perspective
- the shape object exceeds the PostScript point limit for the destination device
- the shape object is used as a pattern, dash, clip, cap, or join

Advanced Printing Features

The `gxCubicSynonymTag` synonym contains a stream of flags and points. The flags are specified in the `gxCubicSynonym` enumeration:

```
enum gxCubicSynonym{
    gxIgnoreFlag      = 0x0000,
    gxLineToFlag      = 0x0001,
    gxCurveToFlag     = 0x0002,
    gxMoveToFlag      = 0x0003,
    gxClosePathFlag   = 0x0004
};
```

Constant descriptions

<code>gxIgnoreFlag</code>	Ignore this flag; get the next one.
<code>gxLineToFlag</code>	Draw a line from the current point to the point specified after this flag.
<code>gxCurveToFlag</code>	Draw a curve from the current point through the three points specified after this flag.
<code>gxMoveToFlag</code>	Move the start of a new contour, which becomes the current point, to the point specified after this flag.
<code>gxClosePathFlag</code>	Close the contour.

The point, line, or curve specified after a line follow the conventions for a point, line, or curve, (`gxPoint`, `gxLine`, or `gxCurve`), respectively. The rendering of the shape still depends on the fill of the shape object and the shape object's style, ink, and transform.

Each flag is a short integer; however, QuickDraw GX only considers the low 8 bits. Your application can store application-specific flags in the other 8 bits of the word. Set bits that are not used to 0.

QuickDraw Picture Synonym

When QuickDraw GX spools a document containing QuickDraw imaging commands, it creates and flattens, for each page, a QuickDraw GX rectangle shape with an attached tag object of tag type 'pict' (the QuickDraw GX constant for that tag type is `gxQuickDrawPictTag`). The tag object contains information that specifies the characteristics and location of a file containing QuickDraw picture data for that page.

When QuickDraw GX subsequently despools the file, it (or the printer driver) uses the QuickDraw GX Translator to convert the QuickDraw picture data into a QuickDraw GX picture shape before printing it. The tag object contains a `gxQuickDrawPict` structure:

```
struct gxQuickDrawPict {
    gxTranslationOptions    options;
    Rect                   srcRect;
    Point                  styleStretch;
```

Advanced Printing Features

```

        unsigned long                dataLength;
        struct gxBitmapDataSourceAlias  alias;
};

```

Field descriptions

options	The translation options to be used by the QuickDraw GX Translator when converting the QuickDraw data.
srcRect	The source rectangle for the translation, in QuickDraw coordinates. It controls scaling of the image. This rectangle is the QuickDraw picture frame that bounds the QuickDraw data.
styleStretch	The scale factor (both horizontal and vertical) to apply to certain items, such as dashes, in QuickDraw picture comments.
dataLength	The length of the QuickDraw picture data, in bytes.
alias	A structure that defines the location of the file containing the QuickDraw data, and the offset within the file to that data.

The QuickDraw GX rectangle shape that the tag object is attached to specifies the destination bounding rectangle for drawing the QuickDraw data (in QuickDraw coordinates). The relative sizes of the source rectangle and destination rectangle control the scaling of the image when it is translated.

The QuickDraw GX Translator is described in the environment chapter of *Inside Macintosh: QuickDraw GX Environment and Utilities*. Tag objects are described in the tag objects chapter of *Inside Macintosh: QuickDraw GX Objects*. The `gxBitmapDataSourceAlias` structure is described in the bitmap shapes chapter of *Inside Macintosh: QuickDraw GX Graphics*. QuickDraw picture data is described in *Inside Macintosh: Imaging With QuickDraw*.

Printing Modes

When you print, QuickDraw GX and the printer driver set up your document for printing based on the specifications in the printer driver. For example, if you print to a PostScript printer, QuickDraw GX converts the picture shapes to the appropriate PostScript directives for you—your application does not need to get involved.

There can be cases, however, in which your application may wish to allow the user to specify an alternative way of printing. Thus, the user may choose to print in a **direct mode**, which is a mode that bypasses QuickDraw GX imaging. For example, direct mode may be used in the following cases:

- to send text to an ImageWriter with built-in fonts
- to send PostScript-only output; for example, by attaching tag objects to empty shape objects, in which the tag object contains PostScript code

The most common reason that a direct mode may be used is to speed up printing. The major drawback to direct-mode printing is that the user cannot redirect the print file that was created during printing to another printer.

Advanced Printing Features

Direct mode is a kind of **job format mode**. QuickDraw GX supports three job format modes, which are shown in Table 4-3. Variables of type `gxJobFormatMode` are used to store the print job format mode.

Table 4-3 Print job format modes

Constant	Value	Explanation
<code>gxGraphicsJobFormatMode</code>	<code>'grph'</code>	Graphics output, which is used as the default for QuickDraw GX printing
<code>gxTextJobFormatMode</code>	<code>'text'</code>	Text-only output
<code>gxPostScriptJobFormatMode</code>	<code>'post'</code>	PostScript-only output

A printer driver may not support all of these modes, or it may support additional modes that the application and printer driver agree to support. To support a job format mode other than `gxGraphicsJobFormatMode`, the application must specify the available modes. The printer driver uses this list of modes to choose its preferred mode. When the user chooses to use direct mode, the user is selecting the printer driver's preferred mode of printing.

For information about how a printer driver sets the preferred mode, see the printer driver chapter of *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers*. For an example of how to set the available modes and set the preferred mode in response to the user choosing direct mode, see “Implementing Direct-Mode Printing” on page 4-35.

IMPORTANT

Only use `gxTextJobFormatMode` printing when the user requests direct-mode printing. ▲

Pen Tables for Vector Devices

If a device driver for a vector device sets up a pen table, your application can access it to determine the colors and sizes of the pens in the device's carousel. The driver sets up a pen table in a tag object and creates a reference to the tag object in the view device object associated with the vector device. For more information about how a driver sets up a pen table, see the printing messages chapter of *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers*.

Your application can reference this pen table by retrieving the contents of the `gxPenTableTag` tag, which is defined as `'pent'`, from the view device object associated with the vector device. For example, if the user creates a line with a thickness that is smaller than a pen's thickness, your application could detect this situation and warn the user that the screen display will not match the printed output.

Advanced Printing Features

The `gxPenTableEntry` structure defines the data available for each pen in the carousel:

```
struct gxPenTableEntry {
    Str31    penName;
    gxColor  penColor;
    fixed    penThickness;
    short    penUnits;
    short    penPosition;
};
```

The contents of the `gxPenTableTag` tag object contain one or more of these pen table entries. The contents of the `gxPenTableTag` tag object are defined by a `gxPenTable` structure:

```
struct gxPenTable{
    short          numPens;
    gxPenTableEntry pens[1];
};
```

Several constants are available for comparison with the contents of the `penUnits` field:

- Use `gxDeviceUnits` to specify device-specific units.
- Use `gxMmUnits` to specify millimeters.
- Use `gxInchesUnits` to specify inches.

Using Advanced Printing Features

This section shows you how to implement advanced QuickDraw GX printing features in your application. This section shows you several ways to manipulate job, printer, print file, and paper-type objects. It also shows you how to set up direct-mode printing and use synonyms.

Using Advanced Job Object Functions

QuickDraw GX advanced job object functions allow you to obtain specific information about a particular job object. You can use these functions to

- retrieve printer driver and device information
- set or retrieve a job object's reference constant
- copy job objects

Obtaining Printer and Printer Driver Information for a Job

The job object contains information about the output and formatting printers. You can obtain references to these printer objects with the `GXGetJobOutputPrinter` and `GXGetJobFormattingPrinter` functions, respectively. Listing 4-1 shows how to obtain the reference to the output printer with the `GXGetJobOutputPrinter` function.

You can use these references to call functions to obtain additional information about the printer and its driver from the printer object's properties. Listing 4-1 also shows how to obtain the printer's name using the `GXGetPrinterName` function, the printer driver's name using the `GXGetPrinterDriverName` function, the printer's type using the `GXGetPrinterType` function, and the printer driver's type using the `GXGetPrinterDriverType` function.

For example, you could obtain this information and display it to the user in a dialog box. In this case, you need to convert the printer type and printer driver type to strings. One way you can do this is with the `BlockMove` function, as shown in Listing 4-1.

Listing 4-1 Obtaining the names and types of a printer and printer driver

```
OSErr MyShowJobPrinterInfo(MyDocumentPtr myDocument)
{
    OSErr      err;
    gxPrinter  jobPrinter;
    OSType     deviceOSType, driverOSType;
    Str255     deviceName, deviceType, driverName, driverType;
    ...
    /*
       Get the current printer for this job. From that, get the
       current device name, driver name, device type, and driver
       type.
    */
    if (err == noErr)
    {
        jobPrinter = GXGetJobOutputPrinter(myDocument->documentJob);

        GXGetPrinterName(jobPrinter, deviceName);
        GXGetPrinterDriverName(jobPrinter, driverName);
        deviceOSType = GXGetPrinterType(jobPrinter);
        driverOSType = GXGetPrinterDriverType(jobPrinter);
    }
}
```

Advanced Printing Features

```

err = GXGetJobError(myDocument->documentJob);
if (err == noErr)
/*
    Since the device and driver type are OSTypes, convert
    them to the Pascal strings to display.
*/
{
    BlockMove(&deviceOSType, &deviceType[1], (long)
                                   (deviceType[0] = 4));
    BlockMove(&driverOSType, &driverType[1], (long)
                                   (driverType[0] = 4));
}
...
return err;
};

```

Getting and Setting the Reference Constant for a Job Object

QuickDraw GX maintains a reference constant in each job object for your application's use. You can use the `GXGetJobRefCon` function to obtain the reference constant and use the `GXSetJobRefCon` function to set it. These functions allow you to associate your own data with a particular job object.

For example, Listing 4-2 shows how you can store a pointer to the document data in the reference constant of a job object for use by an override function that is called by QuickDraw GX.

Listing 4-2 Setting the job object's reference constant property

```

OSErr MyDoFormatDialog(MyDocumentPtr myDocument)
{
    OSErr          err;
    gxFormat        pageFormat;
    gxDialogResult  result;
    gxEditMenuRecord editMenuRec;
    ...

    /*
        Store the pointer to the document in the job object's
        reference constant to access it within the GXFormatDialog
        override.
    */
    GXSetJobRefCon(myDocument->documentJob, myDocument);
}

```

Advanced Printing Features

```

    /* Display and handle the Custom Page Setup dialog box. */
    result = GXFormatDialog(pageFormat, nil, &editMenuRec);
    ...
    return err;
}

```

Listing 4-3 shows the override of the `GXFormatDialog` function, in which the format's job object is retrieved. From the job object, the reference constant property is retrieved, allowing access to the document associated with the job object from the override function.

Listing 4-3 Getting the job object's reference constant property

```

OSErr MyFormatDialogOverride(gxFormat aFormat, StringPtr title,
                             gxDialogResult *result)
{
    MyDocumentPtr    myDocument;
    gxJob            formatJob;

    /*
     * Get the current job object by calling GXGetJob. Retrieve the
     * pointer to the document, and use it to set up the dialog box
     * panel.
     */
    formatJob = GXGetJob();
    myDocument = GXGetJobRefCon(formatJob);
    MySetUpPanel(aFormat, myDocument,
                 GXGetMessageHandlerResFile());

    /* Finally, forward this message to other handlers.*/
    return Forward_GXFormatDialog(aFormat, title, result);
}

```

Copying Job Object Information

You can duplicate a job object using the `GXCopyJob` function. This function allows you to take an existing job object that references the output and formatting printers, format object, and other job-specific information, and duplicate it for use with another document. Listing 4-4 shows how to copy the job object from the source document to the destination document. References to formats are no longer valid after you change job objects because the formats are based on another job object. You must set these references to `nil`.

Listing 4-4 Copying job object information

```
OSErr MyCopyJobToDoc(MyDocumentPtr srcDocument, MyDocumentPtr
                    destDocument)
{
    long pg;

    /*
     * Copy the job object information. Note that this changes any
     * formats that the destination job originally had (and the
     * old references become invalid).
     */
    GXCopyJob(srcDocument->documentJob, destDocument->documentJob);

    /* Invalidate any old format object references */
    for (pg = 1; pg <= destDocument->numPages; pg++)
        destDocument->pageFormat[pg - 1] = nil;

    return GXGetJobError(srcDocument->documentJob);
}
```

Working With Printer Objects

Each job object references two printer objects, a formatting printer and an output printer. A printer object is implicitly created by the `GXNewJob` function. There is no external application interface to create or dispose of printer objects.

Examples of how to retrieve a printer object's properties, such as the printer name, printer type, driver name, and driver type are shown in Listing 4-1 on page 4-22. The following sections show you how to obtain the view devices associated with a printer and use them to determine a printer's resolution, color space, and color profile.

Determining a Printer's Resolution

You can determine a printer's resolution from the view devices to which the printer refers. The mapping property of the view device object contains a matrix in which the scaling information is stored. Listing 4-5 shows how to obtain the highest resolution that a printer supports.

Listing 4-5 Determining a printer's resolution

```
void MyGetFormatDeviceResolution(gxJob whichJob,
                                fixed *hRes, fixed *vRes)
{
    gxPrinter      formatPrinter;
    long           numViewDevices, idx;
    gxViewDevice   printerVDev;
    gxMapping       vDevMapping;

    *hRes = 0;
    *vRes = 0;

    /*
       Get the formatting printer and the number of
       view devices for that printer.
    */
    formatPrinter = GXGetJobFormattingPrinter(whichJob);
    numViewDevices = GXCountPrinterViewDevices (formatPrinter);

    /* Loop through the view devices that this printer supports. */
    for (idx = 1; idx <= numViewDevices; idx++)
    {
        printerVDev = GXGetPrinterViewDevice(formatPrinter, idx);
        GXGetViewDeviceMapping(printerVDev, &vDevMapping);

        if ((vDevMapping.map[0][0] > *hRes) &&
            (vDevMapping.map[1][1] > *vRes))
        {
            *hRes = vDevMapping.map[0][0];
            *vRes = vDevMapping.map[1][1];
        }
    }
}
```

Advanced Printing Features

```

/*
    Convert scaling factors (multiples of 72 dpi) into
    resolutions.
*/
*hRes = FixedMultiply(*hRes, ff(72));
*vRes = FixedMultiply(*vRes, ff(72));
}

```

Retrieving the Color Profile and Color Space for a Printer

If you wish to retrieve the color profile for a printer, you can call the `GXFindPrinterProfile` function to obtain the reference to a printer's color profile object, or you can call the `GXFindFormatProfile` function to obtain the reference to a format's color profile object. You can set these color profiles with the `GXSetPrinterProfile` and `GXSetFormatProfile` functions, respectively. These functions are described in the reference section of this chapter, starting on page 4-84.

If you want to obtain the color profile of a printer associated with a job object, you can obtain the printer object and, with this reference, you can obtain a reference to the printer's view device. The view device's bitmap shape points to both the color set and the color profile for the printer. Listing 4-6 shows how to retrieve the color profile and color space for the formatting printer.

Listing 4-6 Retrieving the printer's color profile and color space

```

gxColorProfile MyGetFormattingPrinterProfile
(MyDocumentPtr myDocument, gxColorSpace *theSpace)
{
    gxShape          deviceBitMap;
    gxBitmap          deviceBits;
    gxPrinter         formattingPrinter;
    gxColorProfile    theProfile;
    gxViewDevice      printerDevice;

    /* Get the first profile for the formatting printer. */
    formattingPrinter =
        GXGetJobFormattingPrinter(myDocument->documentJob);
    GXFindPrinterProfile(formattingPrinter, nil, 1, &theProfile);
}

```

Advanced Printing Features

```

/*
    Look at the characteristics of the formatting printer's
    view device and retrieve the printer's color space.
*/
printerDevice = GXGetPrinterViewDevice(formattingPrinter, 0);
deviceBitMap = GXGetViewDeviceBitmap(printerDevice);
GXGetBitmap(deviceBitMap, &deviceBits, nil);
*theSpace = deviceBits.space;
GXDisposeShape(deviceBitMap);

return theProfile;
}

```

Listing 4-7 shows how the printer's color profile and color space may be used to determine if a color to be printed is in gamut and to convert the color into the printer's color space.

Listing 4-7 Using the printer's color profile to convert colors

```

Boolean MyMakePrinterColor(gxJob theJob,  gxColor *sourceColor,
                                gxColor *printedColor)
{
    gxColorProfile    printerProfile;
    gxColorSpace      printerSpace;
    Boolean           inGamut;

    /* Get the printer's profile. */
    printerProfile = MyGetFormattingPrinterProfile(theJob,
                                                    &printerSpace);

    /*
        Copy the source color, see if it is in gamut, and convert it
        into the device's color space.
    */
    *printedColor = *sourceColor;
    inGamut = GXCheckColor(printedColor, printerSpace, nil,
                            printerProfile);
    GXConvertColor(printedColor, printerSpace, nil,
                   printerProfile);
    return inGamut;
}

```


Note

For more information about colors, color profiles, and color spaces, see the color and color-related objects chapter of *Inside Macintosh: QuickDraw GX Objects*. ♦

Manipulating Print File Objects

Print files can only be created by printing, which causes the document to be spooled to the file. A portable digital document is a print file created by the user printing to a PDD Maker GX desktop printer.

After you create a print file, your application or another application can manipulate it. QuickDraw GX allows your application to

- open and close a print file
- save a print file
- retrieve a job object associated with a print file
- retrieve page or page format data from a print file
- count the pages in a print file
- delete, replace, or insert pages

Opening and Closing a Print File

You use the `GXOpenPrintFile` function to open a print file and use the `GXClosePrintFile` function to close one. You must provide a job object when you open the print file. You can dispose of the job object after the file is closed.

Listing 4-8 shows how to open and close a print file. It also shows how to determine the number of pages in a print file with the `GXCountPrintFilePages` function.

Listing 4-8 Opening and closing a print file

```
OSErr MyGetPrintFilePages(FSSpec *printFSSpec, long *numCopies)
{
    OSErr      err;
    gxPrintFile thePrintFile;
    gxJob       fileJob;

    /*
     * Create a new job object for GXOpenPrintFile, open the print
     * file object, get the number of pages in it, close it, and
     * check for errors. Finally, dispose of the temporary job
     * object and return.
     */
    err = GXNewJob(&fileJob);
```

Advanced Printing Features

```

    if (err == noErr)
    {
        thePrintFile = GXOpenPrintFile(fileJob, printFSSpec,
                                         fsCurPerm);

        *numCopies = GXCountPrintFilePages(thePrintFile);
        GXClosePrintFile(thePrintFile);

        err = GXGetJobError(fileJob);
        GXDisposeJob(fileJob);
    }

    return err;
}

```

Saving a Print File

You use the `GXSavePrintFile` function to save a print file. You should save a print file after you have added, deleted, or modified its pages, formats, or job object information.

Obtaining the Job Object for a Print File

You use the `GXGetPrintFileJob` function to obtain the job object associated with a particular print file object. This function is useful for determining which job object was associated with the print file when the file was opened by the `GXOpenPrintFile` function, if the reference to the job object was not saved.

Reading Print File Data

You use the `GXReadPrintFilePage` function to retrieve a page from a print file along with its page format. The page is returned as a single picture shape, which is how it is stored in the file, even if the page was created with several calls to `GXDrawShape`.

When you call `GXReadPrintFilePage`, you must specify the page number for the page, starting from 1. You must also specify which view ports you want the picture shape to refer to, so that the shape can be drawn through them when it is displayed onscreen. Listing 4-9 shows how to read a page from a print file.

Listing 4-9 Reading a page from a print file

```

OSErr MyReadPrintFilePage(MyDocumentPtr myDocument, FSSpec
                           *printFSSpec, long whichPg,
                           gxFormat *pgFormat, gxShape *pgShape)
{
    gxPrintFile    thePrintFile;

    /*
     * Open the print file object, read the page, close the file,
     * and check for errors.
     */
    thePrintFile = GXOpenPrintFile(myDocument->documentJob,
                                   printFSSpec, fsCurPerm);
    GXReadPrintFilePage(thePrintFile, whichPg, 1,
                        &myDocument->documentViewPort, pgFormat, pgShape);
    GXClosePrintFile(thePrintFile);

    return GXGetJobError(myDocument->documentJob);
}

```

Counting the Pages in a Print File

You use the `GXCountPrintFilePages` function to count the number of pages in the print file object that you specify. See Listing 4-8 on page 4-29 for an example.

Adding or Deleting Print File Pages

After the user prints a file, you can replace, delete, or insert pages. You use the `GXReplacePrintFilePage` function to replace a single page from a print file. You can use the `GXDeletePrintFilePageRange` function to delete a range of pages within a specified print file. You can use the `GXInsertPrintFilePage` function to insert a page in a print file. For changes to the print file to take effect permanently, you must call `GXSavePrintFile` before you call `GXClosePrintFile`.

Defining Different Paper Sizes

QuickDraw GX allows you to define unique paper types for the individual pages of a printable document. You can use the `GXNewPaperType` function to create a new paper-type object for the specified job object, or you can use the `GXGetNewPaperType` function to load a paper-type object from a resource. You use the `GXGetJobPaperType` function to obtain a specific paper-type object by its index into the total set of paper-type object definitions that are accessible from a specific job object. You can use the `GXCountJobPaperTypes` to obtain the total number of paper-type object definitions that are accessible to a particular job object.

Creating a Paper-Type Object

Listing 4-10 shows how to create a new paper-type object. When you create a paper-type object, you specify its name and rectangles that define the paper type's page size and paper size.

Listing 4-10 Creating a new paper-type object

```
OSErr MyCreatePaperType(MyDocumentPtr myDocument, Str31 paperName,
                        gxRectangle *pageSize, gxRectangle *paperSize,
                        gxPaperType *newPaperType)
{
    *newPaperType = GXNewPaperType(myDocument->documentJob,
                                    paperName, pageSize, paperSize);

    return GXGetJobError(myDocument->documentJob);
}
```

You use the GXDisposePaperType function to dispose of a paper-type object when it is no longer needed.

Obtaining the Name of a Paper Type

You use the GXGetPaperTypeName function to obtain a paper-type object's name. Listing 4-11 shows how to use this function to obtain the name of a paper-type object associated with a format object.

Listing 4-11 Obtaining a paper-type object's name

```
OSErr MyGetPaperTypeName(MyDocumentPtr myDocument, Str255
                        paperTypeName)
{
    gxPaperType thePaperType;
    long        curPage;
    gxFormat    pgFormat;

    /*
     * Get the format object for the current page. If it is nil,
     * you should use the default format.
     */
    curPage = myDocument->curPage;
    pgFormat = myDocument->pageFormat[curPage - 1];
    if (pgFormat == nil)
        pgFormat = GXGetJobFormat(myDocument->documentJob, 1);
}
```

```

    /* Get the format object's paper-type and name. */
    thePaperType = GXGetFormatPaperType(pgFormat);
    GXGetPaperTypeName(thePaperType, paperTypeName);

    return GXGetJobError(myDocument->documentJob);
}

```

Obtaining the Dimensions of a Paper Type

You use the `GXGetPaperTypeDimensions` function to obtain the page rectangle and the paper rectangle associated with a paper-type object. The page rectangle is the imageable portion of a page. The paper rectangle defines the size of a page. The rectangle size is specified in fixed 72 dpi units. Listing 4-12 shows how to use this function.

Listing 4-12 Obtaining page and paper rectangles for a paper-type object

```

OSErr MyGetPaperTypeDimensions(MyDocumentPtr myDocument,
                               gxRectangle *pageBounds,
                               gxRectangle *paperBounds)
{
    gxPaperType    thePaperType;
    long           curPage;
    gxFormat       pgFormat;

    /*
     * Get the format object for the current page. If it is nil, use
     * the job object's default format.
     */
    curPage = myDocument->curPage;
    pgFormat = myDocument->pageFormat[curPage - 1];
    if (pgFormat == nil)
        pgFormat = GXGetJobFormat(myDocument->documentJob, 1);

    /*
     * Get the format's paper type and the paper type's bounds.
     * Note that you can also use GXGetFormatDimensions to do this.
     */
    thePaperType = GXGetFormatPaperType(pgFormat);
    GXGetPaperTypeDimensions(thePaperType, pageBounds,
                             paperBounds);

    return GXGetJobError(myDocument->documentJob);
}

```

Scanning the Paper Types Available to a Job

You use the `GXForEachJobPaperTypeDo` function to call an application-defined function for each paper-type object that is accessible to a particular job. The parameters for the `GXForEachJobPaperTypeDo` function, in order, are:

- the job object whose paper-type objects you wish to examine or change
- a pointer to the application-defined function you want to execute on these paper-type objects
- a pointer to a reference constant that refers to additional data you want to make available to the application-defined function
- a Boolean value that specifies whether you wish to include paper-type objects associated with the formatting printer (`true`) or those associated with the output printer (`false`)

Listing 4-13 shows you how to call an application-defined function, `MyPaperTypeFunction`, for each paper-type object associated with the print job's output printer. The pointer to the reference constant is `nil`.

Listing 4-13 Executing a function for each paper-type object

```
OSErr MyListAllPaperTypes(MyDocumentPtr myDocument)
{
    GXForEachJobPaperTypeDo(myDocument->documentJob,
                           (gxPaperTypeProc) MyPaperTypeFunction, nil,
                           false);
    return GXGetJobError(myDocument->documentJob);
}
```

An application-defined function executed by the `GXForEachJobPaperTypeDo` function is defined as follows:

```
typedef gxLoopStatus (*gxPaperTypeProc) (gxPaperType aPaperType,
                                          void *refCon);
```

The first parameter to the application-defined function is the paper-type object that is to be processed. It is set by the `GXForEachJobPaperTypeDo` function to the next paper-type object automatically. The second parameter is the reference constant passed in by the call to `GXForEachJobPaperTypeDo`. The application-defined function returns a loop status, which it may set to terminate the `GXForEachJobPaperTypeDo` function before every paper-type object has been processed.

Listing 4-14 shows an example of an application-defined function that retrieves the paper type's name and dimensions and can be used to display them. It always returns `gxKeepLooping`, which prevents the `GXForEachJobPaperTypeDo` function from terminating until each paper-type object has been processed.

Listing 4-14 Executing a procedure for each paper-type object

```

pascal gxLoopStatus MyPaperTypeFunction(gxPaperType thePaperType,
                                         void *refCon)
{
    gxRectangle pageBounds, paperBounds;
    Str255      paperTypeName;

    /* Get the paper-type object's name. */
    GXGetPaperTypeName(thePaperType, paperTypeName);

    /* Add code here to display the paper-type object's name. */
    ...

    /* Get the paper-type object's dimensions. */
    GXGetPaperTypeDimensions(thePaperType, &pageBounds,
                             &paperBounds);

    /* Add code here to display the dimensions. */
    ...

    /* Keep looping until all paper types are accessed. */
    return gxKeepLooping;
}

```

Implementing Direct-Mode Printing

Some printer drivers support direct-mode printing, also known as text job format mode printing, in which the generality of QuickDraw GX printing is traded off for faster output using unique features built into the printer hardware. For example, an ImageWriter II printer contains built-in fonts, and its printer driver can make use of them to provide faster printing of text. The printer driver typically allows the user to choose direct-mode printing in these cases.

To allow printing in a nongraphics mode, you must call the `GXSetAvailableJobFormatModes` function to inform the printer driver of all the modes that the application supports, such as `gxGraphicsJobFormatMode` for QuickDraw GX printing, `gxTextJobFormatMode` for direct-mode printing, and `gxPostScriptJobFormatMode` for PostScript-only printing.

All applications should support QuickDraw GX printing. Your application might support direct-mode printing by reformatting the document to match the way it will look when printed, or support PostScript-only output by warning the user that the output cannot be retrieved from a print file when printed in this mode.

Advanced Printing Features

Note

If you are reformatting the document to match the fonts built into the printer, you must query the printer for the fonts, line lengths, and other information using the `GXJobFormatModeQuery` function. For more information about the information that can be obtained, see the following section, “Formatting for Text Job Format Mode Printing.” ♦

If you want to know the mode in effect after the user dismisses the Page Setup dialog box, you can call `GXGetJobFormatMode`. To change it, you can call `GXSetJobFormatMode`.

Formatting for Text Job Format Mode Printing

If the user chooses to print in a direct mode and the driver’s preferred mode is `gxTextJobFormatMode`, you may choose to reformat the document based on the characteristics of the printer. You must query the printer driver to obtain these characteristics by calling the `GXJobFormatModeQuery` function, which is described on page 4-83.

QuickDraw GX provides an enumerated data type whose values specify the characteristics that you may determine. You use one of these values in the `GXGetJobFormatModeQuery` function to specify the characteristic of interest. Table 4-4 identifies these characteristics. Variables of type `gxQueryType` are used to store the kind of request.

Table 4-4 Text job format mode query options

Constant	Explanation
<code>gxGetJobFormatLineConstraintQuery</code>	Used to determine line constraint characteristics
<code>gxGetJobFormatFontConstraintQuery</code>	Used to determine font positioning constraints
<code>gxGetJobFormatFontCommonStylesQuery</code>	Used to determine the style name, such as “normal” or “bold”
<code>gxSetStyleJobFormatCommonStyleQuery</code>	Used to set style contents
<code>gxGetJobFormatFontsQuery</code>	Used to determine font information

A query returns a pointer to a data structure that contains the requested information. The kind of data structure depends on the kind of query.

Advanced Printing Features

The following structures are used to interpret the source and destination data:

- For the `gxGetJobFormatLineConstraintQuery` query, the source data is `nil`, and the destination data is returned in a `gxPositionConstraintTable` structure:

```
struct gxPositionConstraintTable {
    gxPoint  phase;
    gxPoint  offset;
    long     numSizes;
    Fixed    sizes[1];
};
```

- For the `gxGetJobFormatFontConstraintQuery` query, the source data is a `gxFont` reference and the destination data is also returned in a `gxPositionConstraintTable` structure.

Note

A `numSizes` value of `gxConstraintRange` indicates a range of sizes, in which `size[0]` specifies the minimum size and `size[1]` specifies the maximum size. ♦

- For the `gxGetJobFormatFontCommonStylesQuery` query, the source data is a `gxFont` reference, and the destination data is returned in a `gxStyleNameTable` structure:

```
struct gxStyleNameTable {
    long     numStyleNames;      /* number of style names */
    Str255   styleNames[1];     /* any number of style names */
};
```

- For the `gxSetStyleJobFormatCommonStyleQuery` query, the source data is a style name from a `gxStyleNameTable` structure, and the destination data is returned in a `gxStyle` reference.
- For the `gxGetJobFormatFontsQuery` query, the source data is `nil`, and the destination data is returned in a `gxFontTable` structure:

```
struct gxFontTable {
    long     numFonts;           /* number of font references */
    gxFont   fonts[1];          /* any number of font references */
};
```

Using Synonyms

Synonyms allow you to specify how QuickDraw GX objects are to be printed. Normally, you do not need to use synonyms because QuickDraw GX and the printer driver determine how output is to be rendered and handle it for you. There may be occasions, however, when you want to explicitly specify how an object is to be printed. For example, you might want to specify how to render a path in cubics or explicitly specify the PostScript operators to use when printing an object.

A synonym is stored as a tag object that is referred to by the shapes, inks, transforms, or other objects that use it. There are several ways you can set up your tag object, which are described in the tag objects chapter of *Inside Macintosh: QuickDraw GX Objects*. Whenever you set up your tag, you must specify the tag type and the data itself. For example, the tag type for PostScript is `gxPostScriptTag`. Its data is a stream of PostScript, such as the following:

```
0 0 moveto 10 10 lineto stroke
```

For more information about the synonyms provided by QuickDraw GX, see the section “Synonyms” on page 4-11.

Advanced Printing Features Reference

This section describes the constants, data types, and functions that are specific to advanced printing features of QuickDraw GX.

The Constants and Data Types sections show the enumerations and data types for loop status information for paper-type objects and printer objects, job object direct modes, status dialog box information, paper-type object mapping information, paper-type object view device tag objects, and synonym information.

The “Functions” section describes functions for working with advanced job object functions, manipulating printer objects, working with QuickDraw GX print file objects, working with paper types, and formatting for specific devices.

Constants and Data Types for Advanced Printing Features

This section describes the data types and constants that you use for job format modes, status dialog box information, and pen tables for vector devices.

Job Format Modes

QuickDraw GX provides job format modes that allow a printer driver and an application to negotiate the best mode for printing. The `gxJobFormatMode` data type specifies modes, which are enumerated as follows:

```
enum {
    /* direct modes for job objects */
    gxGraphicsJobFormatMode    = (gxJobFormatMode) 'grph',
    gxTextJobFormatMode        = (gxJobFormatMode) 'text',
    gxPostscriptJobFormatMode  = (gxJobFormatMode) 'post'
};
typedef OSType gxJobFormatMode;
```

Constant descriptions

`gxGraphicsJobFormatMode`

If set, QuickDraw GX uses graphics mode.

`gxTextJobFormatMode`

If set, QuickDraw GX uses text mode.

`gxPostScriptJobFormatMode`

If set, QuickDraw GX uses PostScript mode.

The application calls the `GXSetAvailableJobFormatModes` function to inform the printer driver of the modes that the application supports, using a `gxJobFormatModeTable` structure to identify the supported modes.

```
struct gxJobFormatModeTable{
    long            numModes;        /* number of direct modes */
    gxJobFormatMode modes[1];       /* any number direct modes */
};
```

Field descriptions

`numModes`

The number of modes that the application supports.

`modes[1]`

An array that contains the modes.

Text Job Format (Direct) Mode

QuickDraw GX provides a text job format mode, sometimes called a direct mode, to format a document to optimize for particular features and capabilities of a device. For example, text mode provides a fast way to print text using the built-in fonts on a device. This feature provides a replacement for draft printing, which was available in previous versions of the printing architecture.

QuickDraw GX defines query types in the query type enumeration to be used with the `gxQueryType` data type:

```
enum {
    /* query types */
    gxGetJobFormatLineConstraintQuery    = (gxQueryType) 0,
    gxGetJobFormatFontsQuery             = (gxQueryType) 1,
    gxGetJobFormatFontCommonStylesQuery  = (gxQueryType) 2,
    gxGetJobFormatFontConstraintQuery    = (gxQueryType) 3,
    gxSetStyleJobFormatCommonStyleQuery  = (gxQueryType) 4
};

typedef long gxQueryType;
```

Constant descriptions

<code>gxGetJobFormatLineConstraintQuery</code>	Used to determine line constraint characteristics.
<code>gxGetJobFormatFontsQuery</code>	Used to determine font information.
<code>gxGetJobFormatFontCommonStylesQuery</code>	Used to determine the style name, such as “normal” or “bold.”
<code>gxGetJobFormatFontConstraintQuery</code>	Used to determine font positioning constraints.
<code>gxSetStyleJobFormatCommonStyleQuery</code>	Used to set style contents.

QuickDraw GX defines constraint ranges for the constraint table in the constraint range enumeration:

```
enum { gxConstraintRange = -1 };
```

Advanced Printing Features

QuickDraw GX stores constraint information in the position constraint table information structure:

```
struct gxPositionConstraintTable {
    gxPoint  phase;
    gxPoint  offset;
    long     numSizes;
    Fixed    sizes[1];
};
```

Field descriptions

phase	Where to start from the upper-left corner of the page.
offset	The distance between legal character positions.
numSizes	The number of sizes.
sizes[1]	An array of sizes.

QuickDraw GX stores style information in the style name table information structure:

```
struct gxStyleNameTable{
    long     numStyleNames;
    Str255   styleNames[1];
};
```

Field descriptions

numStyleNames	The number of style names.
styleNames[1]	An array of strings containing any number of style names.

QuickDraw GX stores font information in the font table information structure:

```
struct gxFontTable {
    long     numFonts;
    gxFont   fonts[1];
};
```

Field descriptions

numFonts	The number of fonts.
fonts	An array of fonts.

The Status Structure

QuickDraw GX defines status type IDs to report various conditions. Not all of these conditions can be reported from the application. For example, although QuickDraw GX defines a status ID for the percentage completion of a print job, it is not available to the application because printing takes place in the background. Status type IDs are specified in the following enumeration:

```
struct gxStatusRecord {
    unsigned short  statusType;
    unsigned short  statusId;
    unsigned short  statusAlertId;
    Signature       statusOwner;
    short           statResId;
    short           statResIndex;
    short           dialogResult;
    unsigned short  bufferLen;
    char            statusBuffer[1];
};

typedef struct gxStatusRecord gxStatusRecord;
```

Field descriptions

<code>statusType</code>	The type of status that this structure represents. This is one of the values shown in Table 4-5.
<code>statusId</code>	The ID of the status that this structure represents. If the value of this field is 0, there is no associated printing alert ('plrt') resource.
<code>statusAlertId</code>	The ID of the printing alert for this status.
<code>statusOwner</code>	The creator type of the owner of this status structure.
<code>statResId</code>	The resource ID for the status ('stat') resource used to process this status.
<code>statResIndex</code>	The index value for indexing into the status resource for this status.
<code>dialogResult</code>	The ID of the button string that was selected to dismiss the printing alert box associated with this status.
<code>bufferLen</code>	The number of bytes in the status buffer.
<code>statusBuffer</code>	This field is a buffer for the caller to store any additional information for use by the status-handling function.

Note

The triplet of values that includes the `statusOwner`, `statResId`, and `statResIndex` fields must be unique for each status structure. ♦

Table 4-5 shows the status types that you can specify in a status structure.

Table 4-5 Status type IDs

Constant	Value	Explanation
<code>gxNonFatalError</code>	1	Affects the icon in the status dialog box
<code>gxFatalError</code>	2	Sends a printing alert to the status dialog box
<code>gxPrinterReady</code>	3	Signals QuickDraw GX to leave alert mode
<code>gxUserAttention</code>	4	Signals initiation of a modal dialog box
<code>gxUserAlert</code>	5	Signals initiation of a printing alert box
<code>gxPageTransmission</code>	6	Signals that a page was sent to the printer and decrements the page counts in strings that are displayed to the user
<code>gxOpenConnectionStatus</code>	7	Signals QuickDraw GX to begin animation on printer icon
<code>gxInformationalStatus</code>	8	Specifies the default status type and has no side effects
<code>gxSpoolingPageStatus</code>	9	Signals that a page was spooled and increments the page count in the status dialog box
<code>gxEndStatus</code>	10	Signals that spooling has ended
<code>gxPercentageStatus</code>	11	Signals QuickDraw GX as to the amount of the job that is currently complete

Pen Tables for Vector Devices

QuickDraw GX defines a tag object for a paper-type object's view device in the pen table tag enumeration:

```
enum { gxPenTableTag = 'pent' };
```

QuickDraw GX defines paper-type object units in the paper-type units enumeration:

```
enum {
    gxDeviceUnits = 0,
    gxMmUnits     = 1,
    gxInchesUnits = 2
};
```

Advanced Printing Features

Constant descriptions

`gxDeviceUnits` If set, QuickDraw GX uses specific printer units.
`gxMmUnits` If set, QuickDraw GX uses millimeters.
`gxInchesUnits` If set, QuickDraw GX uses inches.

QuickDraw GX defines pen information in the pen not loaded enumeration:

```
enum { gxPenNotLoaded = -1};
```

QuickDraw GX stores pen table information in the pen table information structure:

```
struct gxPenTableEntry {
    Str31    penName;
    gxColor  penColor;
    fixed    penThickness;
    short    penUnits;
    short    penPosition;
};
```

Field descriptions

`penName` A string containing the name of the pen.
`penColor` The color that is part of the color set.
`penThickness` The size of the pen.
`penUnits` The units in which the pen thickness is defined.
`penPosition` The pen position in the carousel.

QuickDraw GX stores pen information in the pen table information structure:

```
struct gxPenTable {
    short          numPens;
    gxPenTableEntry pens[1];
};
```

Field descriptions

`numPens` The number of pen entries.
`pens[1]` An array of pen entries.

Constants and Data Types for Synonyms

This section describes the data types and constants that you use for synonyms.

General-Purpose PostScript Operator Synonym

The `gxPostScriptTag` synonym ('psct') for the general-purpose PostScript operator is defined:

```
#define gxPostScriptTag    0x706f7374
```

PostScript Control Information Synonym

The `gxPostControlTag` synonym ('psct') for the PostScript control information is defined:

```
#define gxPostControlTag    0x70736374
```

The `gxPostControl` structure defines the contents of a `gxPostControlTag` synonym:

```
struct gxPostControl {
    long flags;
};
```

Field descriptions

<code>flags</code>	A flag that specifies how a shape is embedded in the PostScript data stream. If it is <code>gxNoSave</code> , the PostScript data should be encapsulated between a save and restore combination. If <code>gxNoSave</code> is not specified or the <code>gxPostControlTag</code> synonym is not present, the save and restore combination is used.
--------------------	---

QuickDraw GX defines PostScript state flag information in the `gxPsStateFlags` enumeration:

```
enum gxPsStateFlags{
    gxNoSave = 1        /* don't do save-restore around PostScript */
                        /* data */
};
```

Dash Synonym

The `gxDashSynonymTag` synonym ('sdsh') for dashes is defined:

```
#define gxDashSynonymTag 0x73647368
```

The `gxDashSynonym` structure defines the contents of a `gxDashSynonymTag` synonym:

```
struct gxDashSynonym {
    long size;
    fixed dashLength[gxAnyNumber]
};
```

Field descriptions

<code>size</code>	The number of elements in a dash array.
<code>dashLength</code>	The array of lengths for the dashes.

Halftone Synonym

The `gxFormatHalftoneInfo` structure defines the contents of a `gxFormatHalftoneTag` synonym:

```
struct gxFormatHalftoneInfo {
    long          numHalftones;
    gxHalftone    halftones[1];
};
```

Field descriptions

<code>numHalftones</code>	The number of halftones available for use.
<code>halftones</code>	The array of halftone specifications.

Halftones are specified in the `gxHalftone` structures, which are described completely in the view-related objects chapter of *Inside Macintosh: QuickDraw GX Objects*:

```
struct gxHalftone{
    fixed          angle;           /* direction of halftone */
    fixed          frequency;       /* cells per inch */
    gxDotType      method;         /* kind of pattern */
    gxTintType     tinting;         /* tint calculation method */
    gxColor        dotColor;       /* color of foreground */
    gxColor        backgroundColor; /* color of background */
    gxColorSpace   tintSpace;      /* color space for tint */
};
```

Line Cap Synonym

The `gxLineCapSynonymTag` synonym ('lcap') for dashes is defined:

```
#define gxLineCapSynonymTag 0x6C636170
```

QuickDraw GX defines line cap information in the line cap synonym enumeration:

```
enum gxLineCaps{
    gxButtCap =      0,
    gxRoundCap =     1,
    gxSquareCap =    2,
    gxTriangleCap =   3
};
```

```
typedef long  gxLineCapSynonym;
```

Constant descriptions

<code>gxButtCap</code>	Use a cap that does not look like a cap, such as the PostScript butt cap.
<code>gxRoundCap</code>	Use a round cap, such as the PostScript round cap.
<code>gxSquareCap</code>	Use a square cap, such as the PostScript projecting square cap.
<code>gxTriangleCap</code>	Use a triangle cap.

Pattern Synonym

The `gxPatternSynonymTag` synonym ('ptrn') for patterns is defined:

```
#define gxPatternSynonymTag 0x7074726E
```

The `gxPatternSynonym` structure defines the contents of a `gxPatternSynonymTag` synonym:

```
struct gxPatternSynonym {
    long    patternType;
    fixed   angle;
    fixed   spacing;
    fixed   thickness;
    gxPoint anchorPoint;
};
```

Advanced Printing Features

Field descriptions

<code>patternType</code>	The pattern type, either <code>gxHatch</code> or <code>gxCrossHatch</code> .
<code>angle</code>	The angle of the lines in the pattern.
<code>spacing</code>	The distance of the lines in the pattern.
<code>thickness</code>	A point that specifies the upper-left corner at which the pattern begins.

Patterns can be either hatch or crosshatch:

```
enum gxPatterns {
    gxHatch          = 0,
    gxCrossHatch     = 1
};
```

Constant descriptions

<code>gxHatch</code>	Use a hatch pattern.
<code>gxCrossHatch</code>	Use a crosshatch pattern.

Cubic Synonym

The `gxCubicSynonymTag` synonym ('cubx') for cubics is defined:

```
#define gxCubicSynonymTag 0x63756278
```

QuickDraw GX defines cubic synonym information in the cubic synonym enumeration:

```
enum gxCubicSynonym{
    gxIgnoreFlag = 0
    gxLineToFlag = 1
    gxCurveToFlag = 2
    gxMoveToFlag = 3
    gxClosePathFlag = 4
};

typedef short gxCubicSynonymFlags;
```

Constant descriptions

<code>gxIgnoreFlag</code>	Ignore this flag; get the next one.
<code>gxLineToFlag</code>	Draw a line from the current point to the point specified after this flag.
<code>gxCurveToFlag</code>	Draw a curve from the current point through the three points specified after this flag.
<code>gxMoveToFlag</code>	Move the start of a new contour, which becomes the current point, to the point specified after this flag.
<code>gxClosePathFlag</code>	Close the contour.

QuickDraw Picture Synonym

The `gxQuickDrawPictTag` tag object contains a `gxQuickDrawPict` structure:

```
struct gxQuickDrawPict {
    gxTranslationOptions      options;
    Rect                      srcRect;
    Point                     styleStretch;
    unsigned long             dataLength;
    struct gxBitmapDataSourceAlias alias;
};
```

Field descriptions

<code>options</code>	The translation options to be used by the QuickDraw GX Translator when converting the QuickDraw data.
<code>srcRect</code>	The source rectangle for the translation, in QuickDraw coordinates. It controls scaling of the image. This rectangle is the QuickDraw picture frame that bounds the QuickDraw data.
<code>styleStretch</code>	The scale factor (both horizontal and vertical) to apply to certain items, such as dashes, in QuickDraw picture comments.
<code>dataLength</code>	The length of the QuickDraw picture data, in bytes.
<code>alias</code>	A structure that defines the location of the file containing the QuickDraw data, and the offset within the file to that data.

Functions

This section describes functions that allow you to implement advanced features of QuickDraw GX printing. Many of these features are implemented by functions that manipulate

- job objects
- printer objects and associated view-device objects and color profiles
- print file objects
- paper-types objects

Included with each function description is a list of specific result codes returned by QuickDraw GX. In addition to these result codes, you may also receive file-system, memory, and resource errors. For a complete listing of specific file-system, memory, and resource errors, see *Inside Macintosh: C Summary* or *Inside Macintosh: Pascal Summary*.

Advanced Printing Features

You should note that not all possible result codes for a particular function are included in function descriptions within this section. For example, the Message Manager, described in *Inside Macintosh: QuickDraw GX Environment and Utilities*, allows QuickDraw GX functions to send specific messages to your application. These messages can also generate errors.

IMPORTANT

All printing functions in QuickDraw GX, with the exception of the `GXGetJobError` function, may move Macintosh memory. The `GXGetJobError` function, however, relies on data that may also move. Therefore, your application should never call a QuickDraw GX printing-related function at interrupt time. ▲

Advanced Job Object Functions

You use the `GXGetJobOutputPrinter` function to determine the output printer for a print job and use the `GXGetJobFormattingPrinter` function to determine the formatting printer for the print job. You use the `GXSelectJobFormattingPrinter` function to specify a formatting printer for a particular print job.

QuickDraw GX provides a place to store a reference constant in each job object for your application's use. A reference constant is accessible through the `GXGetJobRefCon` function. You use the `GXSetJobRefCon` function to set a reference constant.

You can duplicate a job object using the `GXCopyJob` function. This function allows you to take an existing job object and duplicate it for use with another document, causing the associated printer driver, formatting information, and other settings to be used by the other document.

GXSelectJobFormattingPrinter

You can use the `GXSelectJobFormattingPrinter` function to specify a formatting printer for a particular print job.

```
void GXSelectJobFormattingPrinter (gxJob aJob, Str31 printerName);
```

aJob A reference to the job object for which you are specifying a formatting printer.

printerName The name of the formatting printer.

DESCRIPTION

You call `GXSelectJobFormattingPrinter` when the user selects a formatting printer. You can obtain the name of the formatting printer from the Page Setup dialog box and place it in the `printerName` parameter before calling this function.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>fnfErr</code>	The printer driver cannot be located.

GXGetJobFormattingPrinter

You can use the `GXGetJobFormattingPrinter` function to obtain the formatting printer for a particular print job.

```
gxPrinter GXGetJobFormattingPrinter (gxJob aJob);
```

`aJob` A reference to the job object whose formatting printer you wish to obtain.

function result A reference to a printer object.

DESCRIPTION

The `GXGetJobFormattingPrinter` function returns a reference to the formatting printer associated with the job specified in the `aJob` parameter.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXGetJobOutputPrinter

You can use the `GXGetJobOutputPrinter` function to obtain the output printer for a particular job.

```
gxPrinter GXGetJobOutputPrinter (gxJob aJob);
```

`aJob` A reference to the job object whose output printer you wish to obtain.

function result A reference to a printer object.

DESCRIPTION

The `GXGetJobOutputPrinter` function returns a reference to the output printer associated with the job object specified in the `aJob` parameter.

Advanced Printing Features

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For an example that uses the `GXGetJobOutputPrinter` function, see “Obtaining Printer and Printer Driver Information for a Job” on page 4-22.

GXGetJobRefCon

You can use the `GXGetJobRefCon` function to obtain a reference constant associated with a particular job object.

```
void* GXGetJobRefcon (gxJob aJob);
```

<code>aJob</code>	A reference to the job object from which you wish to obtain a reference constant.
-------------------	---

DESCRIPTION

You can use the `GXGetJobRefCon` function to obtain application-defined data associated with a job object.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

To associate a reference constant with a job object, see the description of the `GXSetJobRefCon` function in the next section.

For an example that uses the `GXGetJobRefCon` function, see “Getting and Setting the Reference Constant for a Job Object” on page 4-23.

GXSetJobRefCon

You can use the `GXSetJobRefCon` function to associate a reference constant with a particular job object.

```
void GXSetJobRefcon (gxJob aJob, void *refCon);
```

`aJob` The job object in which to assign a reference constant.

`refCon` A pointer to the reference constant to assign.

DESCRIPTION

The `GXSetJobRefCon` function sets the reference constant for a job object. For example, the reference constant may point to the document data associated with the print job.

RESULT CODES

`gxSegmentLoadFailedErr` A required code segment could not be found, or there was not enough memory to load it.

SEE ALSO

To get the reference constant associated with a job object, see the description of the `GXGetJobRefCon` function in the previous section.

For an example that uses the `GXSetJobRefCon` function, see “Getting and Setting the Reference Constant for a Job Object” on page 4-23.

GXCopyJob

You can use the `GXCopyJob` function to copy job object data from one job object to another.

```
gxJob GXCopyJob (gxJob srcJob, gxJob dstJob);
```

`srcJob` A reference to the job object to copy.

`dstJob` A reference to the job object in which to receive the copied data.

function result A reference to a job object.

Advanced Printing Features

DESCRIPTION

The `GXCopyJob` function makes a copy of the job object specified by the `srcJob` parameter and stores a reference to it in the `dstJob` parameter. If you set the `dstJob` parameter to `nil`, QuickDraw GX allocates and returns a new job object with the properties of the `srcJob` parameter.

For example, you can use this function to copy a job object for use with another document. All information from the source job object is copied into the destination job object, including references to the output and formatting printers, formats, and paper types.

QuickDraw GX allocates appropriate space if the job object that you are copying (the source job object) contains more objects, such as formats, than the job object that you are copying into (the destination job object).

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For an example that uses the `GXCopyJob` function, see “Copying Job Object Information” on page 4-25.

Manipulating Printer Objects

You use the `GXGetJobPrinter` to obtain the printer used by a specific print job. You use the `GXGetPrinterJob` function to obtain the job object associated with a specific printer object.

You use the `GXForEachPrinterViewDeviceDo` function to loop through the view devices associated with a printer object.

You can use the `GXCountPrinterViewDevices` function to obtain the number of view devices associated with a particular printer object.

You use the `GXGetPrinterViewDevice` function to obtain a particular view device associated with a printer object. You use the `GXSelectPrinterViewDevice` function to select the view device to represent a printer’s resolution and color space.

You use the `GXGetPrinterDriverName` and `GXGetPrinterName` functions to obtain the names of a printer and driver, respectively, from a printer object.

You use the `GXGetPrinterDriverType` function to obtain the printer driver type (such as raster, vector, or PostScript) associated with a particular printer object. You use the `GXGetPrinterType` function to obtain the printer’s type.

GXGetJobPrinter

You can use the `GXGetJobPrinter` function to determine the printer object used by a specific job object.

```
gxPrinter GXGetJobPrinter (gxJob aJob);
```

`aJob` A reference to the job object from which you wish to obtain a printer object.

function result A reference to a printer object.

DESCRIPTION

Your application can use the printer object to determine information specific to a device and printer driver for use in formatting and optimizing the user's data.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXGetPrinterJob

You can use the `GXGetPrinterJob` function to obtain the job object associated with a particular printer object.

```
gxJob GXGetPrinterJob (gxPrinter aPrinter);
```

`aPrinter` A reference to the printer object from which you wish to obtain the job object.

function result A reference to the job object associated with the printer object.

DESCRIPTION

The `GXGetPrinterJob` function returns a reference to the job object that refers to the printer object specified in the `aPrinter` parameter.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXForEachPrinterViewDeviceDo

You can use the `GXForEachPrinterViewDeviceDo` function to execute an application-defined function on each view device associated with a particular printer object.

```
void GXForEachPrinterViewDeviceDo (gxPrinter aPrinter,
                                   gxViewDeviceProc aViewDeviceProc,
                                   void *refCon);
```

`aPrinter` A reference to the printer object whose view devices you want to manipulate.

`aViewDeviceProc` The function you want to execute for each view device.

`refCon` A pointer to the reference constant that is passed to the application-defined function.

DESCRIPTION

You can use the `GXForEachPrinterViewDeviceDo` function to perform the actions specified in an application-defined function, `aViewDeviceProc`, on all the view devices associated with a particular printer object.

The `GXForEachPrinterViewDeviceDo` function calls your application-defined function and terminates when the application-defined function returns `gxStopLooping` or when `GXForEachPrinterViewDeviceDo` has been called for each view device.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For information about declaring the application-defined function, see “Message Override Function for the Printing Status Dialog Box” on page 4-90.

GXCountPrinterViewDevices

You can use the `GXCountPrinterViewDevices` function to obtain the number of view devices associated with a particular printer object.

```
long GXCountPrinterViewDevices (gxPrinter aPrinter);
```

`aPrinter` A reference to the printer object whose view devices you want to count.

function result The number of view devices associated with the printer object specified by the `aPrinter` parameter.

DESCRIPTION

The `GXCountPrinterViewDevices` function returns the number of view devices associated with the specified printer object. A printer object can have multiple view devices, one for each possible combination of printer resolution and color space.

RESULT CODES

`gxSegmentLoadFailedErr` A required code segment could not be found, or there was not enough memory to load it.

SEE ALSO

For an example that uses the `GXCountPrinterViewDevices` function, see “Determining a Printer’s Resolution” on page 4-26.

GXGetPrinterViewDevice

You can use the `GXGetPrinterViewDevice` function to obtain a printer object’s view device, using an index value.

```
gxViewDevice GXGetPrinterViewDevice (gxPrinter aPrinter,
                                     long whichViewDevice);
```

`aPrinter` A reference to the printer object whose view device you wish to obtain.

`whichViewDevice`

An index value that specifies the position of the view device reference in the printer object’s view device list.

function result A reference to the specified view device.

Advanced Printing Features

DESCRIPTION

You specify an index value, starting with 1, in the `whichViewDevice` parameter. The parameter specifies a particular view device. You can specify 0 in the `whichViewDevice` parameter to obtain the view device that represents the current view device, which allows you to obtain the current resolution and color space for the printer.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For examples that use the `GXGetPrinterViewDevice` function, see “Determining a Printer’s Resolution” on page 4-26 and “Retrieving the Color Profile and Color Space for a Printer” on page 4-27.

GXSelectPrinterViewDevice

You can use the `GXSelectPrinterViewDevice` function to specify a view device for a printer object.

```
void GXSelectPrinterViewDevice (gxPrinter aPrinter,
                                long whichViewDevice);
```

`aPrinter` A reference to the printer object associated with a particular view device.

`whichViewDevice`

The index value of the view device you want to select.

DESCRIPTION

The `GXSelectPrinterViewDevice` function determines the printer resolution and color space of the printer referenced by the `aPrinter` parameter. A printer object refers to one or more view devices, each of which contains a combination of printer resolution and color space available for the specified printer. You specify an index value, starting with 1, in the `whichViewDevice` parameter. The parameter specifies a particular view device.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXGetPrinterDriverName

You can use the `GXGetPrinterDriverName` function to obtain the name of the printer driver associated with a particular printer object.

```
void GXGetPrinterDriverName (gxPrinter aPrinter, Str31 name);
```

<code>aPrinter</code>	A reference to the printer object associated with a particular formatting printer driver.
-----------------------	---

<code>name</code>	On return, the formatting printer driver's name.
-------------------	--

DESCRIPTION

The `GXGetPrinterDriverName` function retrieves the name of the printer driver to which the `aPrinter` parameter refers and places it in the `name` parameter.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For an example that uses the `GXGetPrinterDriverName` function, see "Obtaining Printer and Printer Driver Information for a Job" on page 4-22.

GXGetPrinterName

You can use the `GXGetPrinterName` function to obtain the name of the printer associated with a particular printer object.

```
void GXGetPrinterName (gxPrinter aPrinter, Str31 name);
```

<code>aPrinter</code>	A reference to the printer object associated with a printer.
-----------------------	--

<code>name</code>	On return, the printer's name.
-------------------	--------------------------------

Advanced Printing Features

DESCRIPTION

The `GXGetPrinterName` function retrieves the name of the printer to which the `aPrinter` parameter refers and places it in the `name` parameter.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For an example that uses the `GXGetPrinterName` function, see “Obtaining Printer and Printer Driver Information for a Job” on page 4-22.

GXGetPrinterDriverType

You can use the `GXGetPrinterDriverType` function to obtain the printer driver type associated with a particular printer object.

```
OSType GXGetPrinterDriverType (gxPrinter aPrinter);
```

<code>aPrinter</code>	A reference to the printer object associated with a particular printer driver type.
-----------------------	---

function result The printer driver type associated with the printer object.

DESCRIPTION

The `GXGetPrinterDriverType` function returns a printer type in the format of an `OSType`. Do not make assumptions about the services provided by driver based on its type.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For an example that uses the `GXGetPrinterDriverType` function, see “Obtaining Printer and Printer Driver Information for a Job” on page 4-22.

For possible values of printer driver types, see “Printer Driver Types” on page 4-7.

GXGetPrinterType

You can use the `GXGetPrinterType` function to obtain the printer type of the printer associated with a particular printer object.

```
OSType GXGetPrinterType (gxPrinter aPrinter);
```

`aPrinter` A reference to the printer object associated with a particular printer.

function result The printer type.

DESCRIPTION

The `GXGetPrinterType` function returns a printer type in the format of an `OSType`; for example, 'LWRW' for LaserWriter GX.

RESULT CODES

`gxSegmentLoadFailedErr` A required code segment could not be found, or there was not enough memory to load it.

SEE ALSO

For an example that uses the `GXGetPrinterType` function, see “Obtaining Printer and Printer Driver Information for a Job” on page 4-22.

Working With QuickDraw GX Print Files

You use the `GXOpenPrintFile` and `GXClosePrintFile` functions to open and close print files.

You use the `GXGetPrintFileJob` function to obtain the job object associated with a particular print file. This function is useful when you need to access or modify information of the job object associated with a print file.

You use the `GXCountPrintFilePages` function to count the number of pages in a print file.

You use the `GXReadPrintFilePage` function to retrieve a page or page format for a print file.

You use the `GXReplacePrintFilePage` function to replace a page or page format from a print file. To insert a new page in a print file, you use the `GXInsertPrintFilePage` function.

You use the `GXDeletePrintFilePageRange` function to delete a range of pages within a specified print file.

Advanced Printing Features

You use the `GXSavePrintFile` function to save a print file. You should save a print file object if you add, delete, or modify its pages, formats, or job object information.

GXOpenPrintFile

You can use the `GXOpenPrintFile` function to open a print file.

```
gxPrintFile GXOpenPrintFile (gxJob aPrintFileJob,
                             FSSpecPtr pFileSpec,
                             char permission);
```

`aPrintFileJob`

A reference to the job object to associate with a particular printer file.

`pFileSpec` A pointer to a file system specification.

`permission`

The access privileges to use when opening the print file object.

function result A reference to a print file object.

DESCRIPTION

The `GXOpenPrintFile` function attempts to open the print file specified by a pointer to a file system specification record, `pFileSpec`. If successful, the function returns a print file object that represents the file. The `permission` parameter specifies the access privileges, which can be read-only or read-and-write access.

The information for the print file's job object is unflattened into the job object you specify in the `aPrintFileJob` parameter. This job object specified in the parameter remains associated with the print file until you close the file by calling the `GXClosePrintFile` function.

To check for errors, you should call the `GXGetJobError` function with the specified job object following calls that operate on the print file.

SPECIAL CONSIDERATIONS

The `GXOpenPrintFile` function sets up a warning handler, which chains to the application's warning handler, if it exists. For more information about warning handlers, see the errors, warnings, and notices chapter of *Inside Macintosh: QuickDraw GX Environment and Utilities*.

Advanced Printing Features

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>gxIncompletePrintFileErr</code>	Contents of file are incomplete.
<code>gxCrashedPrintFileErr</code>	File is currently printing or crashed while printing.
<code>gxInvalidPrintFileVersion</code>	Cannot read file due to incompatible file version.
<code>gxFlattenVersionTooNew</code>	An attempt was made to unflatten a job object that was flattened using a later version of QuickDraw GX.
<code>collectionVersionErr</code>	The version of the collection object is not compatible with the current version of the Collection Manager.

SEE ALSO

For an example that uses the `GXOpenPrintFile` function, see “Opening and Closing a Print File” on page 4-29.

To close a print file object, you use the `GXClosePrintFile` function, which is described in the next section.

GXClosePrintFile

You can use the `GXClosePrintFile` function to close a print file and invalidate the reference to the print file object.

```
void GXClosePrintFile (gxPrintFile aPrintFile);
```

`aPrintFile`

A reference to the print file object for the file to close.

DESCRIPTION

The `GXClosePrintFile` function closes the specified file and invalidates the print file object’s association with a job object.

Advanced Printing Features

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For an example that uses the `GXClosePrintFile` function, see “Opening and Closing a Print File” on page 4-29.

GXGetPrintFileJob

You can use the `GXGetPrintFileJob` function to obtain the job object associated with a particular print file object.

```
gxJob GXGetPrintFileJob (gxPrintFile aPrintFile);
```

`aPrintFile`

A reference to the print file object whose job object you wish to obtain.

function result A reference to a job object.

DESCRIPTION

The `GXGetPrintFileJob` function returns a reference to the job object that was associated with the print file object when you called the `GXOpenPrintFile` function. If you save the reference when you call the `GXOpenPrintFile` function, you do not need to call this function.

This function is useful when you need to access or modify information in the job object associated with a print file object. For example, you can use this function to obtain the job object and then call `GXGetJobError` for the job object to test for an error condition associated with the print file.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXCountPrintFilePages

You can use the `GXCountPrintFilePages` function to count the number of pages in a print file.

```
long GXCountPrintFilePages (gxPrintFile aPrintFile);
```

`aPrintFile`

A reference to the print file object that represents the print file.

function result The number of pages in the file.

DESCRIPTION

The `GXCountPrintFilePages` function returns the number of pages in the file.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXReadPrintFilePage

You can use the `GXReadPrintFilePage` function to retrieve a page or page format for a print file object.

```
void GXReadPrintFilePage (gxPrintFile aPrintFile, long pageNumber,
                          long numViewPorts, gxViewport *viewPortList,
                          gxFormat *pageFormat, gxShape *pageShape);
```

`aPrintFile`

A reference to the print file object whose file you want to access.

`pageNumber`

The page you want to access.

`numViewPorts`

The number of view ports in the view port list.

`viewPortList`

A pointer to a list of references to view ports through which you want the page's picture shape to draw.

`pageFormat`

On return, a reference to the format object associated with the page.

`pageShape`

On return, a reference to the picture shape that contains the page's data.

Advanced Printing Features

DESCRIPTION

The `GXReadPrintFilePage` function retrieves the print file object's page that you specify in the `pageNumber` parameter. It returns the page format and a picture shape representing the contents of the page in the `pageFormat` and `pageShape` parameters, respectively. You can set one or both of these parameters to `nil` if you do not want them returned.

The page shape is associated with the view ports in the `viewPortList` list parameter, which is the list of view ports you want the shape to be drawn through when you call `GXDrawShape` for the shape in the `pageShape` parameter. The `numViewPorts` parameter specifies how many view ports are in the list.

SPECIAL CONSIDERATIONS

Do not change the page format or page shape, pointed to by the `pageFormat` and `pageShape` parameters, directly. If you want to change the format or shape, make a copy of the format or shape and modify the copy. After you make a change to the copy, you can replace the format or page in the print file with your copy or insert your copy into the print file.

For speed and memory efficiency, dispose of the references to the format and page shape objects as soon as they are no longer needed. For example, dispose of them as soon as you make a copy of them or draw a page with them.

The page number specified in the `pageNumber` parameter must be valid. Call the `GXCountPrintFilePages` function to ensure that the page number is valid.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For an example that uses the `GXReadPrintFilePage` function, see "Reading Print File Data" on page 4-30.

GXReplacePrintFilePage

You can use the `GXReplacePrintFilePage` function to replace a page in a print file object.

```
void GXReplacePrintFilePage (gxPrintFile aPrintFile,
                             long pageNumber, gxFormat pageFormat,
                             gxShape pageShape);
```

Advanced Printing Features

<code>aPrintFile</code>	A reference to the print file object in which you want to replace a page.
<code>pageNumber</code>	The page you want to replace.
<code>pageFormat</code>	A reference to the page's format object.
<code>pageShape</code>	A reference to the page's picture shape object.

DESCRIPTION

The `GXReplacePrintFilePage` function replaces in the page specified in the `pageNumber` parameter.

You specify a replacement page format and page shape in the `pageFormat` and `pageShape` parameters, respectively. You can specify `nil` for either of these parameters to ensure that the page format or the page shape remains unchanged.

Any changes you make to a print file are not permanent until you save the print file object with the `GXSavePrintFile` function.

SPECIAL CONSIDERATIONS

After you call the `GXReplacePrintFilePage` function, do not change the page format or page shape referenced by the `pageFormat` and `pageShape` parameters. For example, if you want to change the format or shape later, make a copy, and modify the copy. Dispose of the original page or format after you make the copy.

For speed and memory efficiency, dispose of the references to the format and page parameters immediately after you call `GXReplacePrintFilePage`.

If a format or page is to be duplicated, passing a clone of the object to the function is more efficient than passing a copy. For example, you can pass a clone of a page or format to replicate a page or format already in the file. The cloned object may be one that you have previously read from a print file or one that you created.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

To save a print file object, see the description of the `GXSavePrintFile` function on page 4-70.

GXInsertPrintFilePage

You can use the `GXInsertPrintFilePage` function to insert a new page in a print file.

```
void GXInsertPrintFilePage (gxPrintFile aPrintFile,
                           long atPageNumber, gxFormat pageFormat, gxShape pageShape);
```

`aPrintFile`

A reference to the print file object in whose file you want to insert a page.

`atPageNumber`

The page to insert.

`pageFormat`

A reference to a format object for the inserted page.

`pageShape`

A reference to a picture shape object for the inserted page.

DESCRIPTION

The `GXInsertPrintFilePage` function inserts a page in a print file before the page number that you specify in the `atPageNumber` parameter. You can pass a value of 1 in this parameter to insert the new page before all other pages in the print file. When you pass a value that is higher than the current page count, QuickDraw GX appends the page to the end of the print file.

Any changes you make to a print file are not permanent until you save the print file object by calling the `GXSavePrintFile` function.

SPECIAL CONSIDERATIONS

After you call the `GXInsertPrintFilePage` function, do not change the page format or page shape referenced by the `pageFormat` and `pageShape` parameters. For example, if you want to change the format or shape later, make a copy, and modify the copy. Dispose of the original page or format after you make the copy.

For speed and memory efficiency, dispose of the references to the format and page parameters immediately after you call `GXInsertPrintFilePage`.

If a format or page can be reused, passing a clone of the object to the function is more efficient than passing a copy. For example, you can pass a clone of a page or format to replicate a page or format already in the file. The cloned object may be one that you have previously read from a print file or one that you created.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

To save a print file object, see the description of the `GXSavePrintFile` function on page 4-70.

GXDeletePrintFilePageRange

You can use the `GXDeletePrintFilePageRange` function to delete a range of pages within a particular print file object.

```
void GXDeletePrintFilePageRange (gxPrintFile aPrintFile,
                                long fromPageNumber,
                                long toPageNumber);
```

`aPrintFile`

A reference to the print file object from whose file you want to delete pages.

`fromPageNumber`

The first page that you want to delete.

`toPageNumber`

The last page that you want to delete.

DESCRIPTION

The `GXDeletePrintFilePageRange` function deletes a page or pages in a print file object within the range that you specify in the `fromPageNumber` and `toPageNumber` parameters. The range of page numbers is inclusive. For example, deleting from page 2 to page 3 deletes both pages 2 and 3.

Any changes you make to a print file are not permanent until you save the print file object with the `GXSavePrintFile` function.

Advanced Printing Features

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

To save a print file object, see the description of the `GXSavePrintFile` function in the next section.

GXSavePrintFile

You can use the `GXSavePrintFile` function to save a print file object.

```
void GXSavePrintFile (gxPrintFile aPrintFile, FSSpec *pFileSpec);
```

`aPrintFile`

A reference to the print file object whose file you want to save.

`pFileSpec`

A pointer to a file system specification record.

DESCRIPTION

The `GXSavePrintFile` function writes an entire print file to disk. This file must previously have been opened with the `GXOpenPrintFile` function. To replace or update the print file, you can pass `nil` in the `pFileSpec` parameter. Otherwise, you can specify a name and location in the `pFileSpec` parameter to save the updated print file and leave the original print file intact.

This function compacts a print file by recovering any space no longer needed. Space becomes available when pages are removed or when a format no longer references any pages. This function also permanently saves any changes that you have made to the print file.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

Working With Paper Types

You use the `GXNewPaperType` function to create a new paper-type object, and you use the `GXDisposePaperType` function to dispose of a paper-type object.

You can use the `GXGetNewPaperType` function to retrieve a paper-type object from a resource or use the `GXGetJobPaperType` function to access a specific paper-type object. You use the `GXGetJobPaperType` function to obtain the indexed paper-type object from the total set of paper-type objects that are accessible to a particular job object.

You can use the `GXCountJobPaperTypes` function to obtain the total number of paper-type definitions that are accessible to a particular job object.

You use the `GXCopyPaperType` function to replace the contents of the destination paper-type object with that of the source paper-type object.

You use the `GXGetPaperTypeName` function to obtain the name of a paper-type object.

You use the `GXGetPaperTypeDimensions` function to obtain the page rectangle and the paper rectangle associated with a paper-type object.

You use the `GXGetPaperTypeJob` function to obtain the reference to the job object that owns the paper-type object.

You use the `GXForEachJobPaperTypeDo` function to call an application-defined function for each paper-type definition that is accessible to a particular job object.

GXNewPaperType

You can use the `GXNewPaperType` function to create a new paper-type object.

```
gxPaperType GXNewPaperType (gxJob aJob, Str31 name,
                             gxRectangle *pageSize, gxRectangle *paperSize);
```

<code>aJob</code>	A reference to the job object with which to associate the new paper-type object.
<code>name</code>	The name of the new paper type.
<code>pageSize</code>	A pointer to a rectangle that defines the page size, or imageable area of the paper.
<code>paperSize</code>	A pointer to a rectangle that defines the paper size.

function result A reference to the newly created paper-type object.

DESCRIPTION

The `GXNewPaperType` function creates a paper-type object with the title `name`, the imageable area defined by the `pageSize` rectangle, and the paper size defined by the `paperSize` rectangle. This function associates a paper type of these specifications with the specified job object.

Advanced Printing Features

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>collectionVersionErr</code>	The version of the collection object is not compatible with the current version of the Collection Manager.
<code>gxPaperTypeNotFound</code>	The paper-type object cannot be located.

SEE ALSO

For an example that uses the `GXNewPaperType` function, see “Creating a Paper-Type Object” on page 4-32.

GXDisposePaperType

You can use the `GXDisposePaperType` function to dispose of a paper-type object.

```
void GXDisposePaperType (gxPaperType aPaperType);
```

`aPaperType`

A reference to the paper-type object that you want to dispose of.

DESCRIPTION

The `GXDisposePaperType` function disposes of the paper-type object specified by the `aPaperType` parameter by decrementing its owner count. If the owner count falls to 0, QuickDraw GX may delete the paper-type object.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXGetNewPaperType

You can use the `GXGetNewPaperType` function to create a new paper-type object from a resource template.

```
gxPaperType GXGetNewPaperType (gxJob aJob, short resID);
```

`aJob` A reference to the job object associated with the new paper-type object.
`resID` The ID of the resource template.

function result A reference to a paper-type object.

DESCRIPTION

The `GXGetNewPaperType` function creates a paper-type object in the same way that the `GXNewPaperType` function does, except that the title, the imageable area, and the paper size are defined in the resource identified by `resID`. The `GXGetNewPaperType` function associates the returned paper-type object reference with the `aJob` job object.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>collectionVersionErr</code>	The version of the collection object is not compatible with the current version of the Collection Manager.
<code>gxPaperTypeNotFound</code>	The paper-type object cannot be located.

GXGetJobPaperType

You can use the `GXGetJobPaperType` function to access the specified paper-type object associated with a particular job object.

```
gxPaperType GXGetJobPaperType (gxJob aJob, long whichPaperType,
                               Boolean forFormatDevice,
                               gxPaperType aPaperType);
```

`aJob` A reference to the job object from which to obtain the paper-type object.

`whichPaperType` The index that specifies which paper-type object to obtain.

`forFormatDevice` A Boolean value that specifies whether the paper-type objects are associated with the formatting printer (`true`) or with the output printer (`false`).

`aPaperType` A valid paper-type object reference.

function result A reference to a paper-type object.

DESCRIPTION

The `GXGetJobPaperType` function retrieves the specified paper type from the job object based on the index value in the `whichPaperType` parameter. Index values begin at 1.

Set the `forFormatDevice` parameter to `true` to retrieve only the paper types associated with the formatting printer or to `false` to retrieve only paper types associated with the output printer.

If the desired paper-type object is found, based on its index value, this function replaces the contents of the `aPaperType` parameter with that of the retrieved paper-type object.

If the paper-type object is not located, the job object's error is set to `gxPaperTypeNotFound`. Any error generated by this function can be retrieved using the `GXGetJobError` function.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>gxPaperTypeNotFound</code>	The paper-type object cannot be located.

GXCountJobPaperTypes

You can use the `GXCountJobPaperTypes` function to obtain the total number of paper-type definitions that are accessible to a particular job object.

```
long GXCountJobPaperTypes (gxJob aJob, Boolean forFormatDevice);
```

`aJob` A reference to the job object from which to obtain the number of paper-type definitions.

`forFormatDevice` A Boolean value that specifies whether the paper-type objects are associated with the formatting printer (`true`) or with the output printer (`false`).

function result The number of paper-type objects that are associated with the print job.

DESCRIPTION

The `GXCountJobPaperTypes` function returns the number of paper types associated with either the print job's formatting printer or output printer.

Set the `forFormatDevice` parameter to `true` to count only the paper types associated with the formatting printer or to `false` to count only paper types associated with the output printer.

Depending on the format specification of the job object, the total number of paper types returned may include the total number of system paper types, user paper types, printer driver paper types, and printer-configuration-file paper types.

Use the `GXGetJobError` function to retrieve errors for this function.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXCopyPaperType

You can use the `GXCopyPaperType` function to copy paper-type object data from one paper-type object to another paper-type object.

```
gxPaperType GXCopyPaperType (gxPaperType srcPaperType,
                             gxPaperType dstPaperType);
```

`srcPaperType`

A reference to the paper-type object whose data you want to copy.

`dstPaperType`

A reference to the paper-type object in which to copy the data.

function result Reference to a paper-type object.

DESCRIPTION

The `GXCopyPaperType` function copies the contents of the paper-type object referred to in the `srcPaperType` parameter to the paper-type object referred to in the `dstPaperType` parameter. Each component of the paper-type object is copied. You must specify valid paper types in both the `srcPaperType` and `dstPaperType` parameters.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>gxPaperTypeNotFound</code>	The paper-type object cannot be located.

GXGetPaperTypeName

You can use the `GXGetPaperTypeName` function to obtain the name of a paper-type object.

```
void GXGetPaperTypeName (gxPaperType aPaperType,
                        Str31 name);
```

`aPaperType`

A reference to the paper-type object from which to obtain the name.

`name`

On return, the name of the paper type.

DESCRIPTION

The `GXGetPaperTypeName` function returns the name of the paper-type object specified by the `aPaperType` parameter. The `aPaperType` parameter must refer to a valid paper-type object. The name of the paper-type object is returned in the `name` parameter.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>gxPaperTypeNotFound</code>	The paper-type object cannot be located.

SEE ALSO

For an example that uses the `GXGetPaperTypeName` function, see “Obtaining the Name of a Paper Type” on page 4-32.

GXGetPaperTypeDimensions

You can use the `GXGetPaperTypeDimensions` function to obtain the page rectangle and the paper rectangle associated with a paper-type object.

```
void GXGetPaperTypeDimensions (gxPaperType aPaperType,
                               gxRectangle *aPageSize,
                               gxRectangle *aPaperSize);
```

`aPaperType`

A reference to the paper-type object from which to obtain page and paper sizes.

`aPageSize` A pointer to a rectangle that receives the page size of the paper type.

`aPaperSize`

A pointer to a rectangle that receives the paper size of the paper type.

DESCRIPTION

The `GXGetPaperTypeDimensions` function returns the page and paper size for the specified paper type in the geometry of rectangles. The page rectangle is the imageable portion of a page. The paper rectangle is the size of the paper. The geometry for each rectangle specifies the size in 72 dots-per-inch units. Passing a `nil` pointer for either the `aPageSize` or the `aPaperSize` parameters causes QuickDraw GX to ignore the parameter.

Advanced Printing Features

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>gxPaperTypeNotFound</code>	The paper-type object cannot be located.

SEE ALSO

For an example that uses the `GXGetPaperTypeDimensions` function, see “Obtaining the Dimensions of a Paper Type” on page 4-33.

GXGetPaperTypeJob

You can use the `GXGetPaperTypeJob` function to obtain a reference to the job object that owns a paper-type object.

```
gxJob GXGetPaperTypeJob (gxPaperType aPaperType);
```

`aPaperType`

A reference to the paper-type object for which you want to obtain the job object.

function result A reference to the job object that owns the paper type.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXForEachJobPaperTypeDo

You can use the `GXForEachJobPaperTypeDo` function to call an application-defined function for each paper-type definition that is accessible to a particular job object.

```
void GXForEachJobPaperTypeDo (gxJob aJob,
                             gxPaperTypeProc aPaperTypeProc,
                             void *refCon,
                             Boolean forFormattingPrinter);
```

`aJob` A reference to the job object from which to obtain the paper-type object.

`aPaperTypeProc`

An application-defined function to be called for each paper-type definition accessible to a job object.

Advanced Printing Features

`refCon` A pointer to a reference constant.

`forFormattingPrinter`
 A Boolean value that specifies whether the paper-type objects are associated with the formatting printer (`true`) or with the output printer (`false`).

DESCRIPTION

The `GXForEachJobPaperTypeDo` function loops over each of the paper-type objects for the specified print job, executing the application-supplied function on each one.

The application-defined function is called until either all the paper types have been processed or the function returns the `gxStopLooping` constant.

Set the `forFormattingPrinter` parameter to `true` to execute the application-defined function only on the paper types associated with the formatting printer or to `false` to execute the application-defined function only on paper types associated with the output printer.

RESULT CODES

`gxSegmentLoadFailedErr` A required code segment could not be found, or there was not enough memory to load it.

SEE ALSO

For an example that uses the `GXForEachJobPaperTypeDo` function, see “Scanning the Paper Types Available to a Job” on page 4-34.

For information about declaring the application-defined function, see “Looping Through a Job’s Paper Types” on page 4-92.

Formatting for Specific Devices

You use the `GXSetAvailableJobFormatModes` function to set your list of job format modes for a particular job object, and you use the `GXGetPreferredJobFormatMode` function to obtain the printer driver’s preferred mode.

You use the `GXGetJobFormatMode` function to obtain the current job format mode and the `GXSetJobFormatMode` function to set it.

You use the `GXJobFormatModeQuery` function to get or set additional information for the text job format mode.

GXSetAvailableJobFormatModes

You can use the `GXSetAvailableJobFormatModes` function to set the list of job format modes that your application supports.

```
void GXSetAvailableJobFormatModes (gxJob aJob,
                                   gxJobFormatModeTableHdl aJobFormatModeTableHdl);
```

`aJob` A reference to the job object to which the list of format modes applies.

`aJobFormatModeTableHdl`
 A handle that contains the list of supported modes.

DESCRIPTION

The `GXSetAvailableJobFormatModes` function provides the printer driver with the list of modes that the printer driver could return as its preferred mode.

RESULT CODES

`gxSegmentLoadFailedErr` A required code segment could not be found, or there was not enough memory to load it.

SEE ALSO

For more information about how to use this function, see “Implementing Direct-Mode Printing” on page 4-35.

GXGetPreferredJobFormatMode

You can use the `GXGetPreferredJobFormatMode` function to obtain the preferred mode of printing to the printer associated with a print job.

```
gxJobFormatMode GXGetPreferredJobFormatMode (gxJob aJob,
                                              Boolean *directOnly);
```

`aJob` A reference to the job whose format mode you wish to determine.

`directOnly`
 A pointer to a Boolean value returned by this function that specifies whether the preferred mode is the only mode.

function result The preferred mode.

DESCRIPTION

The `GXGetPreferredJobFormatMode` function returns the preferred mode of printing to the job's output printer. The preferred mode is one of the modes proposed by the application in a call to `GXSetAvailableJobFormatModes`. From that information, the printer driver can respond with its preferred mode.

The preferred mode is typically a mode supported directly by the driver's hardware. In the case of an ImageWriter II, the `GXGetPreferredJobFormatMode` function returns `gxTextJobFormatMode` because it can use fonts built into the printer itself for faster text printing. The preferred mode typically represents the job format mode with the fastest throughput; however, it may limit the quality or even the kind of output that may be printed.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For more information about how to use this function, see "Implementing Direct-Mode Printing" on page 4-35.

GXGetJobFormatMode

You can use the `GXGetJobFormatMode` function to obtain the current job format mode for a particular job object.

```
gxJobFormatMode GXGetJobFormatMode (gxJob aJob);
```

<code>aJob</code>	A reference to the job object whose current format mode you wish to obtain.
-------------------	---

function result The current job format mode.

Advanced Printing Features

DESCRIPTION

The `GXGetJobFormatMode` function returns the current job format mode specified in the `GXSetJobFormatMode` function. The modes defined by QuickDraw GX are:

Constant	Value	Explanation
<code>gxGraphicsJobFormatMode</code>	<code>'grph'</code>	QuickDraw GX default printing
<code>gxTextJobFormatMode</code>	<code>'text'</code>	Text-only output
<code>gxPostScriptJobFormatMode</code>	<code>'post'</code>	PostScript-only output

A printer driver may define additional modes.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXSetJobFormatMode

You can use the `GXSetJobFormatMode` to set the job format mode.

```
void GXSetJobFormatMode (gxJob aJob, gxJobFormatMode aMode);
```

<code>aJob</code>	A reference to the job object associated with the direct mode.
<code>aMode</code>	The direct mode to set.

DESCRIPTION

The `GXSetJobFormatMode` function activates the specified job format mode for a job object whether or not the mode is supported by the printer driver or the application. You might want to call `GXSetJobFormatMode` to set the mode when printing without dialog boxes, such as when the user prints from the Finder.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

GXJobFormatModeQuery

You can use the `GXJobFormatModeQuery` function to get or set additional information related to a job format mode.

```
void GXJobFormatModeQuery (gxJob aJob, gxQueryType aQueryType,
                           void *srcData, void *dstData);
```

`aJob` A reference to the job object for which information relating to the printer's format mode is being requested.

`aQueryType` The kind of query requested.

`srcData` A pointer to the source data.

`dstData` A pointer to the destination data.

DESCRIPTION

The `GXJobFormatModeQuery` function obtains information from a printer driver that relates to the printer driver's preferred mode. The kinds of queries that can be specified in the `aQueryType` parameter depend on the printer driver. The format and direction of the data transfer depends on the kind of query.

QuickDraw GX defines query types for use with printer drivers whose preferred mode is `gxTextJobFormatMode`:

Constant	Value	Explanation
<code>gxGetJobFormatLineConstraintQuery</code>	0	Used to determine line constraint characteristics
<code>gxGetJobFormatFontsQuery</code>	1	Used to determine font information
<code>gxGetJobFormatFontCommonStylesQuery</code>	2	Used to determine style names
<code>gxGetJobFormatFontConstraintQuery</code>	3	Used to determine font positioning constraints
<code>gxSetStyleJobFormatCommonStyleQuery</code>	4	Used to set style names

Advanced Printing Features

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

For more information the kinds of queries and the format of data returned, see “Formatting for Text Job Format Mode Printing” on page 4-36.

Color Profile Functions

QuickDraw GX allows you to find and set the color profiles that are used for color matching. Color matching and the ColorSync Manager are described in *Inside Macintosh: Advanced Color Imaging*.

GXFindPrinterProfile

You can use the `GXFindPrinterProfile` function to determine the color profile used by an output printer.

```
OSErr GXFindPrinterProfile (gxPrinter thePrinter,
                           void *searchData, long index,
                           gxColorProfile *returnedProfile, long *numProfiles);
```

`thePrinter`

A reference to the printer object.

`searchData`

A pointer to a block of data that is assumed to be a ColorSync searching block of type `CMProfileSearchRecord`. If this value is not `nil`, then the value of the `index` parameter must not be 0 if you want the search to take place.

If this value is `nil`, the value of the `index` parameter defines which profile is returned.

`index`

The index of the profile to return. If the value is 0, then the current profile is returned in the `returnedProfile` parameter.

If the value of this parameter is not 0, then the behavior this function depends on the value of the `searchData` parameter. If `index` is not 0 and `searchData` is `nil`, the indexed profile is returned in the `returnedProfile` parameter. If `index` is not 0 and `searchData` is not `nil`, then the printer profiles are searched.

Advanced Printing Features

`returnedProfile`

On return, a list of references to color profiles matching the criteria specified by the `searchData` and `index` parameters. If no color profiles are found, this parameter is `nil` upon return.

`numProfiles`

On return, the number of profiles that were found.

function result An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXFindPrinterProfile` function searches for a color profile that matches the specifications in the `searchData` and `index` parameters.

RESULT CODES

`gxSegmentLoadFailedErr` A required code segment could not be found, or there was not enough memory to load it.

SEE ALSO

The `gxFindPrinterProfile` message that determines which profiles are returned is described in *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers*.

Color matching, color profiles, the `CMProfileSearchRecord` structure, and color profile resources are described in *Inside Macintosh: Advanced Color Imaging*.

GXFindFormatProfile

You can use the `GXFindFormatProfile` function to determine color-matching information for a specific format object. This function is similar to the `GXFindPrinterProfile` function (described in the previous section), except that it finds a color profile that is associated with a format object rather than a printer object.

```
OSErr GXFindFormatProfile (gxFormat theFormat,
                           void *searchData, long index,
                           gxColorProfile *returnedProfile, long *numProfiles);
```

`theFormat` A reference to the format object.

Advanced Printing Features

`searchData`

A pointer to a block of data that is assumed to be a ColorSync searching block of type `CMProfileSearchRecord`. If this value is not `nil`, then the value of the `index` parameter must not be 0 if you want the search to take place.

If this value is `nil`, the value of the `index` parameter defines which profile is returned.

`index`

The index of the profile to return. If the value is 0, then the current profile is returned in the `returnedProfile` parameter.

If the value of this parameter is not 0, then the behavior this function depends on the value of the `searchData` parameter. If `index` is not 0 and `searchData` is `nil`, the indexed profile is returned in the `returnedProfile` parameter. If `index` is not 0 and `searchData` is not `nil`, then the printer profiles are searched.

`returnedProfile`

On return, a list of references to color profiles matching the criteria specified by the `searchData` and `index` parameters. If no color profiles are found, this parameter returns `nil`.

`numProfiles`

On return, the number of profiles that were found.

function result An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXFindFormatProfile` function searches for a color profile that matches the specifications in the `searchData` and `index` parameters.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

The `gxFindFormatProfile` message that determines which profiles are returned is described in *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers*.

Color matching, color profiles, the `CMProfileSearchRecord` structure, and color profile resources are described in *Inside Macintosh: Advanced Color Imaging*.

GXSetPrinterProfile

You can call the GXSetPrinterProfile function to change the current color profile for a printer.

```
OSErr GXSetPrinterProfile (gxPrinter thePrinter,
                           gxColorProfile oldProfile, gxColorProfile newProfile);
```

- thePrinter
- A reference to the printer object.
- oldProfile
- A reference to the profile that has been associated with the printer object.
- newProfile
- A reference to the profile to add to the list of profiles for a printer object.

function result An error code. The value noErr indicates that the operation was successful.

DESCRIPTION

You can call GXSetPrinterProfile to change the current profile for a printer, to replace an existing profile that is associated with the printer object, or to remove a profile from the list of color profiles that are associated with the printer object.

A printer driver or printing extension defines the values of the oldProfile and newProfile parameters that determine what happens in response to this message. Table 4-6 shows an example.

Table 4-6 The actions of the GXSetPrinterProfile function

Value of oldProfile	Value of newProfile	Action taken
nil	nil	None
Valid	nil	oldProfile is deleted from the list of profiles associated with the printer object.
nil	Valid	newProfile is added to the list of profiles for the printer object and becomes the current profile.
Valid	Valid	oldProfile is deleted from the list of profiles, newProfile is added, and newProfile becomes the current profile for the printer object.

Advanced Printing Features

RESULT CODES

`gxSegmentLoadFailedErr` A required code segment could not be found, or there was not enough memory to load it.

SEE ALSO

The `gxSetPrinterProfile` message is described in *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers*.

Color matching, color profiles, and color profile resources are described in *Inside Macintosh: Advanced Color Imaging*.

GXSetFormatProfile

You can use the `GXSetFormatProfile` function to change the current color profile for a format object.

```
OSErr GXSetFormatProfile (gxFormat theFormat,
                          gxColorProfile oldProfile, gxColorProfile newProfile);
```

`theFormat` A reference to the format object.

`oldProfile` A reference to the profile that has been associated with the format object.

`newProfile` A reference to the profile to add to the list of profiles for a format object.

function result An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

You can call the `GXSetFormatProfile` function to change the current profile for a format object, to replace an existing profile that is associated with the format object, or to remove a profile from the list of color profiles that are associated with the format object.

Advanced Printing Features

A printer driver or printing extension defines the values of the `oldProfile` and `newProfile` parameters that determine what happens in response to this message. Table 4-7 shows an example.

Table 4-7 The actions of the `GXSetFormatProfile` function

Value of <code>oldProfile</code>	Value of <code>newProfile</code>	Action taken
<code>nil</code>	<code>nil</code>	None
Valid	<code>nil</code>	<code>oldProfile</code> is deleted from the list of profiles associated with the format object.
<code>nil</code>	Valid	<code>newProfile</code> is added to the list of profiles for the format object and becomes the current profile.
Valid	Valid	<code>oldProfile</code> is deleted from the list of profiles, <code>newProfile</code> is added, and <code>newProfile</code> becomes the current profile for the format object.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
-------------------------------------	--

SEE ALSO

The `gxSetFormatProfile` message is described in *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers*.
Color matching, color profiles, and color profile resources are described in *Inside Macintosh: Advanced Color Imaging*.

Idle Job Function

You can call the `GXIdleJob` function to allow other applications time to execute while your application is spooling.

GXIdleJob

You can use the `GXIdleJob` function to release time to other processes while your application is performing a computationally intensive task.

```
void GXIdleJob (gxJob aJob);
```

`aJob` A reference to a job object.

DESCRIPTION

The `GXIdleJob` function tells QuickDraw GX to release time to other processes that are currently active. If your application is performing a computationally intensive process that can potentially lock other processes out for an extended period of time, you need to periodically call this function.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>gxPrUserAbortErr</code>	The user has canceled printing.

Application-Defined Functions

The following sections describe the application-defined functions for preventing the display of the Printing Status dialog box, for manipulating the view devices associated with a printer object, and for manipulating the paper types associated with a job object.

Message Override Function for the Printing Status Dialog Box

You can call the `GXInstallApplicationOverride` function to install an override function for the `gxJobStatus` message to prevent the Printing Status dialog box from being displayed while spooling.

GXJobStatus

QuickDraw GX sends the `gxJobStatus` message to display the current status of a print job during spooling and despooling. You can install an override function for the `gxJobStatus` message to prevent the display of status information during spooling. Your override function must match the following formal declaration:

```
OSErr GXJobStatus (gxStatusRecord *aStatusRecord);
```

`aStatusRecord`

A pointer to a status structure.

function result An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `gxJobStatus` message when a printing extension or printer driver calls the `GXReportStatus` function. This is not under the application's control.

The default implementation of this message displays the status in the desktop printer window. To prevent the display of the Printing Status dialog box, your override function should return `noErr` as its only action.

SPECIAL CONSIDERATIONS

You never send the `gxJobStatus` message yourself.

You must forward the `gxJobStatus` message to other message handlers.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>gxPrUserAbortErr</code>	The user has canceled printing.

SEE ALSO

The status structure is described in the section “The Status Structure” on page 4-42.

For more information about status information, see the printing functions chapter of *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers*.

Looping Through a Printer's View Devices

The application-defined function called by the `GXForEachPrinterViewDeviceDo` function takes two parameters: the view device object associated with a particular printer object, and a pointer to a reference constant in which you specify data passed into the application-defined function. For example, this is how you should declare the application-function if you were to name it `MyViewDeviceFunction`:

```
gxLoopStatus MyViewDeviceFunction(gxViewDevice aViewDevice,
                                   void *refCon);
```

`aViewDevice`

A reference to the current view device. This is provided by QuickDraw GX when the `MyViewDeviceFunction` function is called.

`refCon`

A pointer to a reference constant.

function result A Boolean to indicate whether looping should stop.

DESCRIPTION

When you use the `GXForEachPrinterViewDeviceDo` function, QuickDraw GX calls the application-defined function for each view device object referenced by the specified printer object until the application-defined function returns `gxStopLooping` or there are no more view devices in the list. If you want the `GXForEachPrinterViewDeviceDo` function to continue with the next view device, return `gxKeepLooping` from the application-defined function.

Looping Through a Job's Paper Types

The application-defined function called by the `GXForEachJobPaperTypeDo` function takes two parameters: the view device object associated with a particular printer object, and a pointer to a reference constant in which you specify data passed into the application-defined function. For example, this is how you should declare the application-function if you were to name it `MyPaperTypeFunction`:

```
gxLoopStatus MyPaperTypeFunction(gxPaperType aPaperType,
                                   void *refCon);
```

`aPaperType`

A reference to the current paper type. This is provided by QuickDraw GX when the `MyPaperTypeFunction` function is called.

`refCon`

A pointer to a reference constant.

function result A Boolean to indicate whether looping should stop.

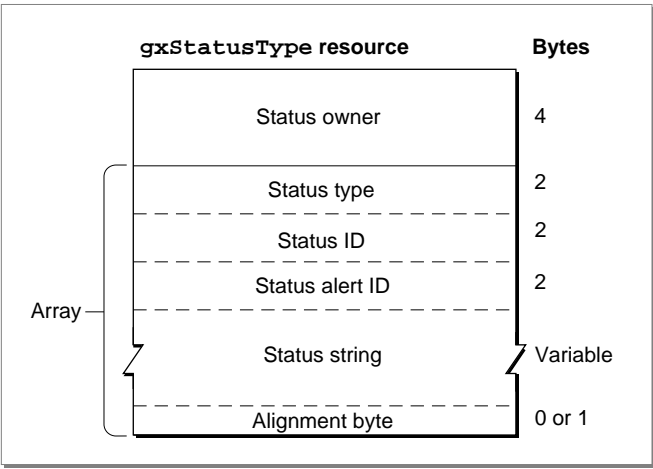
DESCRIPTION

When you use the `GXForEachJobPaperTypeDo` function, QuickDraw GX calls the application-defined function for each paper-type object referenced by the specified job object until the application-defined function returns `gxStopLooping` or there are no more paper types in the list. If you want the `GXForEachJobPaperTypeDo` function to continue with the next paper type, return `gxKeepLooping` from the application-defined function.

The Status Resource

You need to include a status resource, of type `gxStatusType`, to define the status messages that are displayed during the printing process. Figure 4-3 shows the structure of the status resource.

Figure 4-3 The status resource



The status resource contains a count of the status entries and an array of status definitions.

- Status owner. The signature of the printing extension or printer driver to which this status resource belongs.

Each status definition contains four values:

- Status type. The kind of status message that this is. The status type constants are shown in Table 4-8.
- Status ID. The ID of this status message within the status resource. You typically assign sequential numbers to the status messages within each status resource, as shown in the example at the end of this section.

Advanced Printing Features

- Status alert ID. The ID of the printing alert associated with this status message. Use the ID 0 to indicate that this status message does not require a printing alert.
- Status string. The status message string to display to the user.

Most of the status types produce side effects. For example, if you send a status message with status type `gxSpoolingPageStatus`, the page count is incremented in the spooling status that is displayed on the user's screen. Table 4-8 shows the status type constants and the side effects associated with each.

Table 4-8 Status types

Constant	Value	Explanation of side effects
<code>gxNonFatalError</code>	1	Affects the icon that is displayed during spooling
<code>gxFatalError</code>	2	Displays a printing alert during spooling
<code>gxPrinterReady</code>	3	Signals that alert mode is done
<code>gxUserAttention</code>	4	Signals initiation of a modal alert
<code>gxUserAlert</code>	5	Signals initiation of a printing alert
<code>gxPageTransmission</code>	6	Signals that a page has been sent to the printer and increments the printed page count
<code>gxOpenConnectionStatus</code>	7	Signals that animation of the printer icon is to begin
<code>gxInformationalStatus</code>	8	Displays an informational status message and continues
<code>gxSpoolingPageStatus</code>	9	Signals that a page has been spooled and increments the spooled page count
<code>gxEndStatus</code>	10	Signals the end of spooling
<code>gxPercentageStatus</code>	11	Signals the percentage of the current print job that is currently complete

Summary of Advanced Printing Features

Constants and Data Types for Advanced Printing Features

Job Format Modes

```
typedef OSType gxJobFormatMode;

enum {
    /* job format modes */
    gxGraphicsJobFormatMode    = (gxJobFormatMode) 'grph', /* graphics mode */
    gxTextJobFormatMode        = (gxJobFormatMode) 'text', /* text mode */
    gxPostscriptJobFormatMode  = (gxJobFormatMode) 'post'  /* format mode */
};

struct gxJobFormatModeTable {
    long          numModes;      /* number of modes */
    gxJobFormatMode modes[1];    /* any number of modes */
};
```

Text Job Format (Direct) Mode

```
typedef long gxQueryType; /* a query type */

enum {
    /* query types */
    gxGetJobFormatLineConstraintQuery    = (gxQueryType) 0, /* line */
                                                    /* constraint */
    gxGetJobFormatFontsQuery             = (gxQueryType) 1, /* fonts */
    gxGetJobFormatFontCommonStylesQuery  = (gxQueryType) 2, /* font common */
                                                    /* style */
    gxGetJobFormatFontConstraintQuery    = (gxQueryType) 3, /* font */
                                                    /* constraint */
    gxSetStyleJobFormatCommonStyleQuery  = (gxQueryType) 4 /* common */
                                                    /* style */
};

enum { gxConstraintRange = -1 };
```

Advanced Printing Features

```

struct gxPositionConstraintTable {
    gxPoint    phase;           /* the phase */
    gxPoint    offset;         /* the offset */
    long       numSizes;        /* the number of constraint sizes */
    Fixed      sizes[1];        /* any number of constraint sizes */
};

struct gxStyleNameTable {
    long       numStyleNames;   /* number of style names */
    Str255     styleNames[1];   /* any number of style names */
};

struct gxFontTable {
    long       numFonts;        /* number of font references */
    gxFont     fonts[1];        /* any number of font references */
};

```

The Status Structure

```

enum {
    /*status type IDs*/
    gxNonFatalError      = 1,    /* affects icon on spooling dialog box */
    gxFatalError         = 2,    /* sends up user alert on spooling */
                                /* dialog box */
    gxPrinterReady       = 3,    /* signals QuickDraw GX to leave alert */
                                /* mode */
    gxUserAttention       = 4,    /* signals initiation of a modal alert */
    gxUserAlert          = 5,    /* signals initiation of a movable */
                                /* modal alert */
    gxPageTransmission   = 6,    /* signals page sent to printer, */
                                /* increments page count in strings to */
                                /* user */
    gxOpenConnectionStatus = 7,  /* signals QuickDraw GX to begin */
                                /* animation of printer icon */
    gxInformationalStatus = 8,    /* default status type, no effects */
    gxSpoolingPageStatus  = 9,    /* signals page spooled, increments */
                                /* page count in spooling dialog box */
    gxEndStatus          = 10,    /* signals end of spooling */
    gxPercentageStatus    = 11,   /* signals to QuickDraw GX the amount */
                                /* of the print job which is currently */
                                /* complete */
};

```

Advanced Printing Features

```

/* status structure */
struct gxStatusRecord {
    unsigned short statusType; /* the type of status */
    unsigned short statusId; /* specific status ID */
    unsigned short statusAlertId; /* printing alert ID for status */
    Signature      statusOwner; /* status owner signature */
    short          statResId; /* resource ID for 'stat' resource */
    short          statResIndex; /* index into 'stat' resource */
    short          dialogResult; /* ID of button selected to
                                dismiss the printing alert box */
    unsigned short bufferLen; /* # of bytes in status buffer */
    char           statusBuffer[1]; /* user response from alert */
};

```

Pen Tables for Vector Devices

```

enum { gxPenTableTag = 'pent' };

enum {
    /* pen widths */
    gxDeviceUnits = 0, /* device-specific units */
    gxMmUnits = 1, /* millimeters */
    gxInchesUnits = 2 /* inches */
};

/* pen constants */
enum { gxPenNotLoaded = -1 };

/* pen table entry structure */
struct gxPenTableEntry {
    Str31      penName; /* name of the pen */
    gxColor    penColor; /* color that is part of the */
                                /* color set */
    fixed      penThickness; /* size of the pen */
    short      penUnits; /* specifies units in which pen */
                                /* thickness is defined */
    short      penPosition; /* pen position in the carousel */
};

```

```

struct gxPenTable {
    short          numPens;      /* number of pen entries in */
                                /* the following array */
    gxPenTableEntry pens[1];     /* array of pen entries */
};

```

Constants and Data Types for Synonyms

General-Purpose PostScript Operator Synonym

```

#define gxPostScriptTag    0x706f7374      /* 'post' synonym */

```

PostScript Control Information Synonym

```

#define gxPostControlTag    0x70736374      /* 'psct' synonym */

enum gxPsStateFlags {
    gxNoSave = 1      /* don't do save-restore around PostScript data */
};

struct gxPostControl {
    long  flags;      /* PostScript state flags */
};

```

Dash Synonym

```

#define gxDashSynonymTag    0x73647368 /* 'sdsh' synonym */

struct gxDashSynonym {
    long    size;      /* number of elements in array */
    fixed   dashLength[gxAnyNumber]; /* array of dash lengths */
};

```

Halftone Synonym

```

enum { gxFormatHalftoneTag = 'half' };

struct gxFormatHalftoneInfo{
    long numHalftones;      /* how many halftones */
    gxHalftone  halftones[1]; /* any number of halftones */
};

```

Advanced Printing Features

```

struct gxHalftone{
    fixed      angle;          /* direction of halftone */
    fixed      frequency;      /* cells per inch */
    gxDotType   method;        /* kind of pattern */
    gxTintType   tinting;      /* tint calculation method */
    gxColor     dotColor;      /* color of foreground */
    gxColor     backgroundColor; /* color of background */
    gxColorSpace tintSpace;    /* color space for tint */
};

```

Line Cap Synonym

```

#define gxLineCapSynonymTag 0x6c636170 /* 'lcap' synonym */

enum gxLineCaps {          /* possible line caps */
    gxButtCap      = 0,      /* square butt cap */
    gxRoundCap     = 1,      /* round cap */
    gxSquareCap    = 2,      /* square cap */
    gxTriangleCap  = 3       /* triangle cap */
};

typedef long  gxLineCapSynonym;      /* line cap type */

```

Pattern Synonym

```

#define gxPatternSynonymTag 0x70747265 /* 'ptrn' synonym */

enum gxPatterns {
    gxHatch        = 0,      /* hatch pattern */
    gxCrossHatch   = 1       /* crosshatch pattern */
;

struct gxPatternSynonym {
    long  patternType;      /* one of the patterns: hatch or crosshatch */
    fixed  angle;           /* angle at which pattern is drawn */
    fixed  spacing;         /* distance between two parallel pattern lines */
    fixed  thickness;       /* thickness of the pattern */
    gxPoint anchorPoint;    /* point with respect to the pattern position */
                                /* calculated */
};

```

Cubic Synonym

```
#define gxCubicSynonymTag 0x63756278    /* 'cubx' synonym */

enum gxCubicSynonym {
    gxIgnoreFlag = 0x0000,    /* ignore this word, get next one */
    gxLineToFlag = 0x0001,    /* draw a line to point following this */
                                /* flag */
    gxCurveToFlag = 0x0002,    /* draw a curve through the 3 points */
                                /* following this flag */
    gxMoveToFlag = 0x0003,    /* start a new contour at the point */
                                /* following this flag */
    gxClosePathFlag = 0x0004    /* close the contour */
};

#define gxCubicInstructionMask 0x000F    /* low 4 bits are point */
                                         /* instructions */

typedef short gxCubicSynonymFlags;
/* low 8 bits are instruction (moveto, lineto, curveto, closepath) */
```

QuickDraw Picture Synonym

```
struct gxQuickDrawPict {
    gxTranslationOptions    options;    /* translator options */
    Rect                    srcRect;    /* QuickDraw source Rect */
    Point                   styleStretch; /* the scale factor */
    unsigned long           dataLength; /* length of picture data */
    struct gxBitmapDataSourceAlias alias;    /* alias to QuickDraw data */
};
```

Functions

Working With Advanced Job Object Functions

```
void GXSelectJobFormattingPrinter
    (gxJob aJob, Str31 printerName);

gxPrinter GXGetJobFormattingPrinter
    (gxJob aJob);

gxPrinter GXGetJobOutputPrinter
    (gxJob aJob);

void* GXGetJobRefcon
    (gxJob aJob);
```


Advanced Printing Features

```
void GXSetJobRefcon      (gxJob aJob, void *refCon);
gxJob GXCopyJob          (gxJob srcJob, gxJob dstJob);
```

Manipulating Printer Objects

```
gxPrinter GXGetJobPrinter (gxJob aJob);
gxJob GXGetPrinterJob     (gxPrinter aPrinter);
void GXForEachPrinterViewDeviceDo
    (gxPrinter aPrinter,
     gxViewDeviceProc aViewDeviceProc,
     void *refCon);

long GXCountPrinterViewDevices
    (gxPrinter aPrinter);

gxViewDevice GXGetPrinterViewDevice
    (gxPrinter aPrinter, long whichViewDevice);

void GXSelectPrinterViewDevice
    (gxPrinter aPrinter, long whichViewDevice);

void GXGetPrinterDriverName (gxPrinter aPrinter, Str31 name);
void GXGetPrinterName       (gxPrinter aPrinter, Str31 name);
OSType GXGetPrinterDriverType
    (gxPrinter aPrinter);
OSType GXGetPrinterType     (gxPrinter aPrinter);
```

Working With QuickDraw GX Print Files

```
gxPrintFile GXOpenPrintFile (gxJob aPrintFileJob, FSSpecPtr pFileSpec,
                             char permission);

void GXClosePrintFile      (gxPrintFile aPrintFile);
gxJob GXGetPrintFileJob    (gxPrintFile aPrintFile);
long GXCountPrintFilePages (gxPrintFile aPrintFile);
void GXReadPrintFilePage   (gxPrintFile aPrintFile, long pageNumber,
                             long numViewPorts, gxViewPort *viewPortList,
                             gxFormat *pageFormat, gxShape *pageShape);

void GXReplacePrintFilePage (gxPrintFile aPrintFile,
                             long pageNumber, gxFormat pageFormat,
                             gxShape pageShape);

void GXInsertPrintFilePage (gxPrintFile aPrintFile, long atPageNumber,
                             gxFormat pageFormat, gxShape pageShape);

void GXDeletePrintFilePageRange
    (gxPrintFile aPrintFile,
     long fromPageNumber,
     long toPageNumber);

void GXSavePrintFile       (gxPrintFile aPrintFile, FSSpec *pFileSpec);
```

Working With Paper Types

```

gxPaperType GXNewPaperType    (gxJob aJob, Str31 name,
                                gxRectangle *pageSize, gxRectangle *paperSize);

void GXDisposePaperType      (gxPaperType aPaperType);

gxPaperType GXGetNewPaperType
                                (gxJob aJob, short resID);

gxPaperType GXGetJobPaperType
                                (gxJob aJob, long whichPaperType,
                                Boolean forFormatDevice,
                                gxPaperType aPaperType);

long GXCountJobPaperTypes    (gxJob aJob, Boolean forFormatDevice);

gxPaperType GXCopyPaperType  (gxPaperType srcPaperType,
                                gxPaperType dstPaperType);

void GXGetPaperTypeName      (gxPaperType aPaperType,
                                Str31 name);

void GXGetPaperTypeDimensions
                                (gxPaperType aPaperType,
                                gxRectangle *aPageSize,
                                gxRectangle *aPaperSize);

gxJob GXGetPaperTypeJob      (gxPaperType aPaperType);

void GXForEachJobPaperTypeDo
                                (gxJob aJob, gxPaperTypeProc aPaperTypeProc,
                                void *refCon, Boolean forFormattingPrinter);

```

Formatting for Specific Devices

```

void GXSetAvailableJobFormatModes
                                (gxJob aJob,
                                JobFormatModeTableHdl
                                aJobFormatModeTableHdl);

gxJobFormatMode GXGetPreferredJobFormatMode
                                (gxJob aJob,
                                Boolean *directOnly);

gxJobFormatMode GXGetJobFormatMode
                                (gxJob aJob);

void GXSetJobFormatMode      (gxJob aJob, gxJobFormatMode aMode);

void GXJobFormatModeQuery    (gxJob aJob, gxQueryType aQueryType,
                                void *srcData, void *dstData);

```

Color Profile Functions

```

OSErr GXFindPrinterProfile (gxPrinter thePrinter, void *searchData,
                           long index, gxColorProfile *returnedProfile,
                           long *numProfiles);

OSErr GXFindFormatProfile (gxFormat theFormat, void *searchData,
                           long index, gxColorProfile *returnedProfile,
                           long *numProfiles);

OSErr GXSetPrinterProfile (gxPrinter thePrinter,
                           gxColorProfile oldProfile,
                           gxColorProfile newProfile);

OSErr GXSetFormatProfile (gxFormat theFormat,
                           gxColorProfile oldProfile,
                           gxColorProfile newProfile);

```

Idle Job Function

```
void GXIdleJob (gxJob aJob);
```

Application-Defined Functions

```

OSErr GXJobStatus (gxStatusRecord *aStatusRecord);

gxLoopStatus MyViewDeviceFunction
    (gxViewDevice aViewDevice,
     void *refCon);

gxLoopStatus MyPaperTypeFunction
    (gxPaperType aPaperType,
     void *refCon);

```

